Package 'Distributacalcul'

January 10, 2024

Type Package

Title Probability Distribution Functions

Version 0.4.0

Date 2024-01-08

Description Calculates expected values, variance, different moments (kth moment, truncated mean), stop-loss, mean excess loss, Value-at-Risk (VaR) and Tail Value-at-Risk (TVaR) as well as some density and cumulative (survival) functions of continuous, discrete and compound distributions. This package also includes a visual 'Shiny' component to enable students to visualize distributions and understand the impact of their parameters. This package is intended to expand the 'stats' package so as to enable students to develop an intuition for probability.

License MIT + file LICENSE

URL https://alec42.github.io/Distributacalcul_Package/

BugReports https://github.com/alec42/Distributacalcul_Package/issues

Encoding UTF-8

Imports statmod, stats, dplyr, rlang

RoxygenNote 7.2.3

Suggests learnr, knitr, rmarkdown, spelling, testthat

Language en-CA

NeedsCompilation no

Author Alec James van Rassel [aut, cre, cph],
Gabriel Crépeault-Cauchon [aut, ccp],
Étienne Marceau [tch, sad],
Hélène Cossette [tch, sad],
Laboratoire Act & Risk [fnd, sht],
École d'actuariat de l'Université Laval [fnd, his, uvp],
Natural Sciences and Engineering Research Council of Canada [fnd],
Marc-André Devost [ccp]

Maintainer Alec James van Rassel <alec.van-rassel.1@ulaval.ca>

Repository CRAN

Date/Publication 2024-01-10 14:03:16 UTC

R topics documented:

`Beta` *Beta Distribution*

Description

Beta distribution with shape parameters α and β.

Usage

```
expValBeta(shape1, shape2)

varBeta(shape1, shape2)

kthMomentBeta(k, shape1, shape2)

expValLimBeta(d, shape1, shape2)

expValTruncBeta(d, shape1, shape2, less.than.d = TRUE)

stopLossBeta(d, shape1, shape2)

meanExcessBeta(d, shape1, shape2)

VatRBeta(kap, shape1, shape2)

TVatRBeta(kap, shape1, shape2)

mgfBeta(t, shape1, shape2, k0)
```

Arguments

`shape1`	shape parameter α, must be positive.
`shape2`	shape parameter β, must be positive.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.
`k0`	point up to which to sum the distribution for the approximation.

Details

The Beta distribution with shape parameters α and β has density:

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{(}\beta - 1)$$

for $x \in [0, 1]$, $\alpha, \beta > 0$.

Value

Function :

- `expValBeta` gives the expected value.

- varBeta gives the variance.
- kthMomentBeta gives the kth moment.
- expValLimBeta gives the limited mean.
- expValTruncBeta gives the truncated mean.
- stopLossBeta gives the stop-loss.
- meanExcessBeta gives the mean excess loss.
- VatRBeta gives the Value-at-Risk.
- TVatRBeta gives the Tail Value-at-Risk.
- mgfBeta gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRBeta is a wrapper for the qbeta function from the stats package.

Examples

```
expValBeta(shape1 = 3, shape2 = 5)

varBeta(shape1 = 4, shape2 = 5)

kthMomentBeta(k = 3, shape1 = 4, shape2 = 5)

expValLimBeta(d = 0.3, shape1 = 4, shape2 = 5)

expValTruncBeta(d = 0.4, shape1 = 4, shape2 = 5)

# Values less than d
expValTruncBeta(d = 0.4, shape1 = 4, shape2 = 5, less.than.d = FALSE)

stopLossBeta(d = 0.3, shape1 = 4, shape2 = 5)

meanExcessBeta(d = .3, shape1 = 4, shape2 = 5)

VatRBeta(kap = .99, shape1 = 4, shape2 = 5)

TVatRBeta(kap = .99, shape1 = 4, shape2 = 5)

mgfBeta(t = 1, shape1 = 3, shape2 = 5, k0 = 1E2)
```

`binom` *Binomial Distribution*

Description

Binomial distribution with size n and probability of success p.

Usage

```
expValBinom(size, prob)

varBinom(size, prob)

expValTruncBinom(d, size, prob, less.than.d = TRUE)

VatRBinom(kap, size, prob)

TVatRBinom(kap, size, prob)

pgfBinom(t, size, prob)

mgfBinom(t, size, prob)
```

Arguments

`size`	Number of trials (0 or more).
`prob`	Probability of success in each trial.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The binomial distribution with probability of success p for n trials has probability mass function :

$$Pr(X = k) = \binom{n}{k} p^n (1-p)^{n-k}$$

for $k = 0, 1, 2, \ldots, n$, $p \in [0, 1]$, and $n > 0$

Value

Function :

- `mgfBinom` gives the moment generating function (MGF).

- pgfBinom gives the probability generating function (PGF).
- expValBinom gives the expected value.
- varBinom gives the variance.
- expValTruncBinom gives the truncated mean.
- TVatRBinom gives the Tail Value-at-Risk.
- VatRBinom gives the Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRBinom is a wrapper of the qbinom function from the stats package.

Examples

```
expValBinom(size = 3, prob = 0.5)

varBinom(size = 3, prob = 0.5)


expValTruncBinom(d = 2, size = 3, prob = 0.5)
expValTruncBinom(d = 0, size = 3, prob = 0.5, less.than.d = FALSE)


VatRBinom(kap = 0.8, size = 5, prob = 0.2)

TVatRBinom(kap = 0.8, size = 5, prob = 0.2)


pgfBinom(t = 1, size = 3, prob = 0.5)

mgfBinom(t = 1, size = 3, prob = 0.5)
```

---

bivariateAMH *Bivariate Ali-Mikhail-Haq Copula*

Description

Computes CDF, PDF and simulations of of the bivariate Ali-Mikhail-Haq copula.

Usage

```
cBivariateAMH(u1, u2, dependencyParameter, ...)

cdBivariateAMH(u1, u2, dependencyParameter, ...)

crBivariateAMH(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`dependencyParameter`	correlation parameter.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The bivariate Ali-Mikhail-Haq copula has CDF :

Value

Function :

- `cBivariateAMH` returns the value of the copula.
- `cdBivariateAMH` returns the value of the density copula.
- `crBivariateAMH` returns simulated values of the copula.

Examples

```
cBivariateAMH(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

cdBivariateAMH(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

crBivariateAMH(numberSimulations = 10, seed = 42, dependencyParameter = 0.2)
```

`bivariateCA` *Bivariate Cuadras-Augé Copula*

Description

Computes CDF and simulations of the bivariate Cuadras-Augé copula.

Usage

```
cBivariateCA(u1, u2, dependencyParameter, ...)

crBivariateCA(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`dependencyParameter`	correlation parameter.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The bivariate Cuadras-Augé copula has CDF :

$$C(u_1, u_2) = u_1 u_2^{1-\alpha} \times \mathbf{1}_{\{u_1 \leq u_2\}} + u_1^{1-\alpha} u_2 \times \mathbf{1}_{\{u_1 \geq u_2\}}$$

for $u_1, u_2, \alpha \in [0, 1]$. It is the geometric mean of the independance and upper Fréchet bound copulas.

Value

Function :

- `cBivariateCA` returns the value of the copula.
- `crBivariateCA` returns simulated values of the copula.

Examples

```
cBivariateCA(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

crBivariateCA(numberSimulations = 10, seed = 42, dependencyParameter = 0.2)
```

`bivariateClayton` *Bivariate Clayton Copula*

Description

Computes CDF, PDF and simulations of the bivariate Clayton copula.

Usage

```
cBivariateClayton(u1, u2, dependencyParameter, ...)

cdBivariateClayton(u1, u2, dependencyParameter, ...)

crBivariateClayton(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

- `u1, u2` points at which to evaluate the copula.
- `dependencyParameter` correlation parameter.
- `...` other parameters.
- `numberSimulations` Number of simulations.
- `seed` Simulation seed, 42 by default.

Details

The bivariate Clayton copula has CDF :

Value

Function :

- `cBivariateAMH` returns the value of the copula.
- `cdBivariateAMH` returns the value of the density function associated to the copula.

Examples

```
cBivariateClayton(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

cdBivariateClayton(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

crBivariateClayton(numberSimulations = 10, seed = 42, dependencyParameter = 0.2)
```

`bivariateEFGM` *Bivariate Eyraud-Farlie-Gumbel-Morgenstern (EFGM) Copula*

Description

Computes CDF, PDF and simulations of the EFGM copula.

Usage

```
cBivariateEFGM(u1, u2, dependencyParameter)

cdBivariateEFGM(u1, u2, dependencyParameter)

crBivariateEFGM(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

- `u1, u2`: points at which to evaluate the copula.
- `dependencyParameter`: correlation parameter.
- `numberSimulations`: Number of simulations.
- `seed`: Simulation seed, 42 by default.

Details

The EFGM copula has CDF :

Value

Function :

- `cBivariateEFGM` returns the value of the copula.
- `cdBivariateEFGM` returns the value of the density function associated to the copula.
- `crBivariateEFGM` returns simulated values of the copula.

Examples

```
cBivariateEFGM(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

cdBivariateEFGM(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

crBivariateEFGM(numberSimulations = 10, seed = 42, dependencyParameter = 0.2)
```

---

bivariateFrank *Bivariate Frank Copula*

Description

Computes CDF, PDF and simulations of the bivariate Frank copula.

Usage

```
cBivariateFrank(u1, u2, dependencyParameter, ...)

cdBivariateFrank(u1, u2, dependencyParameter, ...)

crBivariateFrank(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

u1, u2	points at which to evaluate the copula.
dependencyParameter	correlation parameter.
...	other parameters.
numberSimulations	Number of simulations.
seed	Simulation seed, 42 by default.

Details

The bivariate Frank copula has CDF :

Value

Function :

- cBivariateFrank returns the value of the copula.
- cdBivariateFrank returns the value of the density function associated to the copula.
- crBivariateFrank returns simulated values of the copula.

Examples

```
cBivariateFrank(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

cdBivariateFrank(u1 = .76, u2 = 0.4, dependencyParameter = 0.4)

crBivariateFrank(numberSimulations = 10, seed = 42, dependencyParameter = 0.2)
```

---

bivariateGumbel *Bivariate Gumbel Copula*

Description

Computes CDF, PDF and simulations of the bivariate Gumbel copula.

Usage

```
cBivariateGumbel(u1, u2, dependencyParameter, ...)

cdBivariateGumbel(u1, u2, dependencyParameter, ...)

crBivariateGumbel(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`dependencyParameter`	correlation parameter.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The bivariate Gumbel copula has CDF :

Value

Function :

- `cBivariateGumbel` returns the value of the copula.
- `cdBivariateGumbel` returns the value of the density function associated to the copula.
- `crBivariateGumbel` returns simulated values of the copula.

Examples

```
cBivariateGumbel(u1 = .76, u2 = 0.4, dependencyParameter = 1.4)

cdBivariateGumbel(u1 = .76, u2 = 0.4, dependencyParameter = 1.4)

crBivariateGumbel(numberSimulations = 10, seed = 42, dependencyParameter = 1.2)
```

`bivariateMO` *Bivariate Marshall-Olkin Copula*

Description

Computes CDF and simulations of the bivariate Marshall-Olkin copula.

Usage

```
cBivariateMO(u1, u2, dependencyParameter, ...)

crBivariateMO(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`dependencyParameter`	correlation parameters, must be vector of length 2.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The bivariate Marshall-Olkin copula has CDF :

$$C(u_1, u_2) = u_1 u_2^{1-\beta} \times \mathbf{1}_{\{u_1^\alpha \leq u_2^\beta\}} + u_1^{1-\alpha} u_2 \times \mathbf{1}_{\{u_1^\alpha \geq u_2^\beta\}}$$

for $u_1, u_2, \alpha, \beta \in [0, 1]$. It is the geometric mean of the independance and upper Fréchet bound copulas.

Value

Function :

- `cBivariateMO` returns the value of the copula.
- `crBivariateMO` returns simulated values of the copula.

Examples

```
cBivariateMO(u1 = .76, u2 = 0.4, dependencyParameter = c(0.4, 0.3))

crBivariateMO(numberSimulations = 10, seed = 42, dependencyParameter = c(0.2, 0.5))
```

CompBinom *Compound Binomial Distribution*

Description

Computes various risk measures (mean, variance, Value-at-Risk (VaR), and Tail Value-at-Risk (TVaR)) for the compound Binomial distribution.

Usage

```
pCompBinom(
  x,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  k0,
  distr_severity = "Gamma"
)

expValCompBinom(
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  distr_severity = "Gamma"
)

varCompBinom(
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  distr_severity = "Gamma"
)

VatRCompBinom(
  kap,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  k0,
  distr_severity = "Gamma"
)

TVatRCompBinom(
  kap,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
```

```
  vark,
  k0,
  distr_severity = "Gamma"
)
```

Arguments

x	vector of quantiles
size	Number of trials (0 or more).
prob	Probability of success in each trial.
shape	shape parameter α, must be positive.
rate	rate parameter β, must be positive.
scale	alternative parameterization to the rate parameter, scale = 1 / rate.
k0	point up to which to sum the distribution for the approximation.
distr_severity	Choice of severity distribution. • "gamma" (default) • "lognormal" only for the expected value and variance.
kap	probability.
vark	Value-at-Risk (VaR) calculated at the given probability kap.

Details

The compound binomial distribution has density ....

Value

Function :

- pCompBinom gives the cumulative density function.
- expValCompBinom gives the expected value.
- varCompBinom gives the variance.
- TVatRCompBinom gives the Tail Value-at-Risk.
- VatRCompBinom gives the Value-at-Risk.

Returned values are approximations for the cumulative density function, TVaR, and VaR.

Examples

```
pCompBinom(x = 2, size = 1, prob = 0.2, shape = log(1000) - 0.405,
          rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")

expValCompBinom(size = 1, prob = 0.2, shape = log(1000) - 0.405, rate = 0.9^2,
          distr_severity = "Lognormale")

varCompBinom(size = 1, prob = 0.2, shape = log(1000) - 0.405, rate = 0.9^2,
          distr_severity = "Lognormale")
```

```
VatRCompBinom(kap = 0.9, size = 1, prob = 0.2, shape = log(1000) - 0.405,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")

vark_calc <- VatRCompBinom(kap = 0.9, size = 1, prob = 0.2, shape = 0.59,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")
TVatRCompBinom(kap = 0.9, size = 1, prob = 0.2, shape = 0.59, rate = 0.9^2,
            vark = vark_calc, k0 = 1E2, distr_severity = "Gamma")
```

CompNBinom *Compound Negative Binomial Distribution*

Description

Computes various risk measures (mean, variance, Value-at-Risk (VatR), and Tail Value-at-Risk (TVatR)) for the compound Negative Binomial distribution.

Usage

```
pCompNBinom(
  x,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  k0,
  distr_severity = "Gamma"
)

expValCompNBinom(
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  distr_severity = "Gamma"
)

varCompNBinom(
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  distr_severity = "Gamma"
)
```

```
VatRCompNBinom(
  kap,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  k0,
  distr_severity = "Gamma"
)

TVatRCompNBinom(
  kap,
  vark,
  size,
  prob,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  k0,
  distr_severity = "Gamma"
)
```

Arguments

`x`	vector of quantiles
`size`	Number of successful trials.
`prob`	Probability of success in each trial.
`shape`	shape parameter α, must be positive.
`rate`	rate parameter β, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.
`k0`	point up to which to sum the distribution for the approximation.
`distr_severity`	Choice of severity distribution. • "gamma" (default) • "lognormal" only for the expected value and variance.
`kap`	probability.
`vark`	Value-at-Risk (VaR) calculated at the given probability kap.

Details

The compound negative binomial distribution has density ....

Value

Function :

- pCompNBinom gives the cumulative density function.
- expValCompNBinom gives the expected value.
- varCompNBinom gives the variance.
- TVatRCompNBinom gives the Tail Value-at-Risk.
- VatRCompNBinom gives the Value-at-Risk.

Returned values are approximations for the cumulative density function, TVatR, and VatR.

Examples

```
pCompNBinom(x = 2, size = 1, prob = 0.2, shape = log(1000) - 0.405,
          rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")


expValCompNBinom(size = 4, prob = 0.2, shape = 0, scale = 1,
         distr_severity = "Lognormal")

varCompNBinom(size = 1, prob = 0.2, shape = log(1000) - 0.405, rate = 0.9^2,
          distr_severity = "Lognormale")

VatRCompNBinom(kap = 0.9, size = 1, prob = 0.2, shape = 0.59,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")

vark_calc <- VatRCompNBinom(kap = 0.9, size = 1, prob = 0.2, shape = 0.59,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")
TVatRCompNBinom(kap = 0.9, size = 1, prob = 0.2, shape = 0.59, rate = 0.9^2,
            vark = vark_calc, k0 = 1E2, distr_severity = "Gamma")
```

CompPois *Compound Poisson Distribution*

Description

Computes various risk measures (mean, variance, Value-at-Risk (VaR), and Tail Value-at-Risk (TVaR)) for the compound Poisson distribution.

Usage

```
pCompPois(
  x,
  lambda,
  shape,
  rate = 1/scale,
  scale = 1/rate,
```

```
    k0,
    distr_severity = "Gamma"
  )

  expValCompPois(
    lambda,
    shape,
    rate = 1/scale,
    scale = 1/rate,
    distr_severity = "Gamma"
  )

  varCompPois(
    lambda,
    shape,
    rate = 1/scale,
    scale = 1/rate,
    distr_severity = "Gamma"
  )

  VatRCompPois(
    kap,
    lambda,
    shape,
    rate = 1/scale,
    scale = 1/rate,
    k0,
    distr_severity = "Gamma"
  )

  TVatRCompPois(
    kap,
    lambda,
    shape,
    rate = 1/scale,
    scale = 1/rate,
    vark,
    k0,
    distr_severity = "Gamma"
  )
```

Arguments

`x`	vector of quantiles
`lambda`	Rate parameter λ.
`shape`	shape parameter α, must be positive.
`rate`	rate parameter β, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.

`k0`	point up to which to sum the distribution for the approximation.
`distr_severity`	Choice of severity distribution. • "gamma" (default) • "lognormal" only for the expected value and variance.
`kap`	probability.
`vark`	Value-at-Risk (VaR) calculated at the given probability kap.

Details

The compound Poisson distribution with parameters ... has density ....

Value

Function :

- `pCompPois` gives the cumulative density function.
- `expValCompPois` gives the expected value.
- `varCompPois` gives the variance.
- `TVatRCompPois` gives the Tail Value-at-Risk.
- `VatRCompPois` gives the Value-at-Risk.

Returned values are approximations for the cumulative density function, TVaR, and VaR.

Examples

```
pCompPois(x = 2, lambda = 2, shape = log(1000) - 0.405,
          rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")

expValCompPois(lambda = 2, shape = log(1000) - 0.405, rate = 0.9^2,
          distr_severity = "Lognormale")

varCompPois(lambda = 2, shape = log(1000) - 0.405, rate = 0.9^2,
          distr_severity = "Lognormale")

VatRCompPois(kap = 0.9, lambda = 2, shape = log(1000) - 0.405,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")

vark_calc <- VatRCompPois(kap = 0.9, lambda = 2, shape = 0.59,
            rate = 0.9^2, k0 = 1E2, distr_severity = "Gamma")
TVatRCompPois(kap = 0.9, lambda = 2, shape = 0.59, rate = 0.9^2,
            vark = vark_calc, k0 = 1E2, distr_severity = "Gamma")
```

`Erl` *Hypergeometric Distribution*

Description

Hypergeometric distribution where we have a sample of k balls from an urn containing N, of which m are white and n are black.

Usage

```
expValErl(N = n + m, m, n = N - m, k)

varErl(N = n + m, m, n = N - m, k)
```

Arguments

`N`	Total number of balls (white and black) in the urn. $N = n + m$
`m`	Number of white balls in the urn.
`n`	Number of black balls in the urn. Can specify n instead of N.
`k`	Number of balls drawn from the urn, k = 0, 1, ..., m + n.

Details

The Hypergeometric distribution for N total items of which m are of one type and n of the other and from which k items are picked has probability mass function :

$$Pr(X = x) = \frac{\binom{m}{k}\binom{n}{k-x}}{\binom{N}{k}}$$

for $x = 0, 1, \ldots, \min(k, m)$.

Value

Function :

- `expValErl` gives the expected value.
- `varErl` gives the variance.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
# With total balls specified
expValErl(N = 5, m = 2, k = 2)

# With number of each colour of balls specified
expValErl(m = 2, n = 3, k = 2)


# With total balls specified
varErl(N = 5, m = 2, k = 2)

# With number of each colour of balls specified
varErl(m = 2, n = 3, k = 2)
```

`erlang` *Erlang Distribution*

Description

Erlang distribution with shape parameter n and rate parameter β.

Usage

```
dErlang(x, shape, rate = 1/scale, scale = 1/rate)

pErlang(q, shape, rate = 1/scale, scale = 1/rate, lower.tail = TRUE)

expValErlang(shape, rate = 1/scale, scale = 1/rate)

varErlang(shape, rate = 1/scale, scale = 1/rate)

kthMomentErlang(k, shape, rate = 1/scale, scale = 1/rate)

expValLimErlang(d, shape, rate = 1/scale, scale = 1/rate)

expValTruncErlang(d, shape, rate = 1/scale, scale = 1/rate, less.than.d = TRUE)

stopLossErlang(d, shape, rate = 1/scale, scale = 1/rate)

meanExcessErlang(d, shape, rate = 1/scale, scale = 1/rate)

VatRErlang(kap, shape, rate = 1/scale, scale = 1/rate)

TVatRErlang(kap, shape, rate = 1/scale, scale = 1/rate)

mgfErlang(t, shape, rate = 1/scale, scale = 1/rate)
```

Arguments

`x, q`	vector of quantiles.
`shape`	shape parameter n, must be a positive integer.
`rate`	rate parameter β, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The Erlang distribution with shape parameter n and rate parameter β has density:

$$f\left(x\right) = \frac{\beta^n}{\Gamma(n)} x^{n-1} \mathrm{e}^{-\beta x}$$

for $x \in \mathcal{R}^+$, $\beta > 0$, $n \in \mathcal{N}^+$.

Value

Function :

- `dErlang` gives the probability density function (PDF).
- `pErlang` gives the cumulative density function (CDF).
- `expValErlang` gives the expected value.
- `varErlang` gives the variance.
- `kthMomentErlang` gives the kth moment.
- `expValLimErlang` gives the limited mean.
- `expValTruncErlang` gives the truncated mean.
- `stopLossErlang` gives the stop-loss.
- `meanExcessErlang` gives the mean excess loss.
- `VatRErlang` gives the Value-at-Risk.
- `TVatRErlang` gives the Tail Value-at-Risk.
- `mgfErlang` gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRErlang is a wrapper of the `qgamma` function from the stats package.

Examples

```
dErlang(x = 2, shape = 2, scale = 4)

pErlang(q = 2, shape = 2, scale = 4)

expValErlang(shape = 2, scale = 4)

varErlang(shape = 2, scale = 4)

kthMomentErlang(k = 3, shape = 2, scale = 4)

expValLimErlang(d = 2, shape = 2, scale = 4)

# With rate parameter
expValTruncErlang(d = 2, shape = 2, scale = 4)

# Values greater than d
expValTruncErlang(d = 2, shape = 2, scale = 4, less.than.d = FALSE)

stopLossErlang(d = 2, shape = 2, scale = 4)

meanExcessErlang(d = 3, shape = 2, scale = 4)

# With scale parameter
VatRErlang(kap = .2, shape = 2, scale = 4)

# With rate parameter
VatRErlang(kap = .2, shape = 2, rate = 0.25)

# With scale parameter
TVatRErlang(kap = .2, shape = 3, scale = 4)

# With rate parameter
TVatRErlang(kap = .2, shape = 3, rate = 0.25)

mgfErlang(t = 2, shape = 2, scale = .25)
```

Exp *Exponential Distribution*

Description

Exponential distribution with rate parameter β.

Usage

```
expValExp(rate = 1/scale, scale = 1/rate)
```

```
varExp(rate = 1/scale, scale = 1/rate)

kthMomentExp(k, rate = 1/scale, scale = 1/rate)

expValLimExp(d, rate = 1/scale, scale = 1/rate)

expValTruncExp(d, rate = 1/scale, scale = 1/rate, less.than.d = TRUE)

stopLossExp(d, rate = 1/scale, scale = 1/rate)

meanExcessExp(d, rate = 1/scale, scale = 1/rate)

VatRExp(kap, rate = 1/scale, scale = 1/rate)

TVatRExp(kap, rate = 1/scale, scale = 1/rate)

mgfExp(t, rate = 1/scale, scale = 1/rate)
```

Arguments

`rate`	rate parameter β, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The Exponential distribution with rate parameter β has density:

$$f(x) = \frac{1}{\beta} e^{-\beta x}$$

for $x \in \mathcal{R}^+$, $\beta > 0$.

Value

Function :

- `expValExp` gives the expected value.
- `varExp` gives the variance.
- `kthMomentExp` gives the kth moment.
- `expValLimExp` gives the limited mean.
- `expValTruncExp` gives the truncated mean.

- stopLossExp gives the stop-loss.
- meanExcessExp gives the mean excess loss.
- VatRExp gives the Value-at-Risk.
- TVatRExp gives the Tail Value-at-Risk.
- mgfExp gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRExp is a wrapper of the qexp function from the stats package.

Examples

```
# With scale parameter
expValExp(scale = 4)

# With rate parameter
expValExp(rate = 0.25)

# With scale parameter
varExp(scale = 4)

# With rate parameter
varExp(rate = 0.25)

# With scale parameter
kthMomentExp(k = 2, scale = 4)

# With rate parameter
kthMomentExp(k = 2, rate = 0.25)

# With scale parameter
expValLimExp(d = 2, scale = 4)

# With rate parameter
expValLimExp(d = 2, rate = 0.25)

# With scale parameter
expValTruncExp(d = 2, scale = 4)

# With rate parameter, values greater than d
expValTruncExp(d = 2, rate = 0.25, less.than.d = FALSE)

# With scale parameter
stopLossExp(d = 2, scale = 4)

# With rate parameter
stopLossExp(d = 2, rate = 0.25)

# With scale parameter
```

```
meanExcessExp(d = 2, scale = 4)

# With rate parameter
meanExcessExp(d = 5, rate = 0.25)

# With scale parameter
VatRExp(kap = .99, scale = 4)

# With rate parameter
VatRExp(kap = .99, rate = 0.25)

# With scale parameter
TVatRExp(kap = .99, scale = 4)

# With rate parameter
TVatRExp(kap = .99, rate = 0.25)

mgfExp(t = 1, rate = 5)
```

`frechet` *Fréchet Copula*

Description

Computes CDF and simulations of the Fréchet copula.

Usage

```
cFrechet(u1, u2, dependencyParameter, ...)

crFrechet(numberSimulations = 10000, seed = 42, dependencyParameter)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`dependencyParameter`	correlation parameters, must be vector of length 2.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The Fréchet copula has CDF :

$$C(u_1, u_2) = (1 - \alpha - \beta)(u_1 \times u_2) + \alpha \min(u_1, u_2) + \beta \max(u_1 + u_2 - 1, 0)$$

for $u_1, u_2, \alpha, \beta \in [0, 1]$ and $\alpha + \beta \leq 1$.

Value

Function :

- cFrechet returns the value of the copula.
- crFrechet returns simulated values of the copula.

Examples

```
cFrechet(u1 = .76, u2 = 0.4, dependencyParameter = c(0.2, 0.3))

crFrechet(numberSimulations = 10, seed = 42, dependencyParameter = c(0.2, 0.3))
```

`frechetLowerBound` *Fréchet Lower Bound Copula*

Description

Computes CDF and simulations of the Fréchet lower bound copula.

Usage

```
cFrechetLowerBound(u1, u2, ...)

crFrechetLowerBound(numberSimulations = 10000, seed = 42)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The Fréchet lower bound copula has CDF :

$$C(u_1, u_2) = \max(u_1 + u_2 - 1, 0)$$

for $u_1, u_2 \in [0, 1]$.

Value

Function :

- cFrechetLowerBound returns the value of the copula.
- crFrechetLowerBound returns simulated values of the copula.

Examples

```
cFrechetLowerBound(u1 = .76, u2 = 0.4)

crFrechetLowerBound(numberSimulations = 10, seed = 42)
```

`frechetUpperBound` *Fréchet Upper Bound Copula*

Description

Computes CDF and simulations of the Fréchet upper bound copula.

Usage

```
cFrechetUpperBound(u1, u2, ...)

crFrechetUpperBound(numberSimulations = 10000, seed = 42)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The Fréchet upper bound copula has CDF :

$$C(u_1, u_2) = \min(u_1, u_2)$$

for $u_1, u_2 \in [0, 1]$.

Value

Function :

- `cFrechetUpperBound` returns the value of the copula.
- `crFrechetUpperBound` returns simulated values of the copula.

Examples

```
cFrechetUpperBound(u1 = .56, u2 = 0.4)

crFrechetUpperBound(numberSimulations = 10, seed = 42)
```

Gamma *Gamma Distribution*

Description

Gamma distribution with shape parameter α and rate parameter β.

Usage

```
expValGamma(shape, rate = 1/scale, scale = 1/rate)

varGamma(shape, rate = 1/scale, scale = 1/rate)

kthMomentGamma(k, shape, rate = 1/scale, scale = 1/rate)

expValLimGamma(d, shape, rate = 1/scale, scale = 1/rate)

expValTruncGamma(d, shape, rate = 1/scale, scale = 1/rate, less.than.d = TRUE)

stopLossGamma(d, shape, rate = 1/scale, scale = 1/rate)

meanExcessGamma(d, shape, rate = 1/scale, scale = 1/rate)

VatRGamma(kap, shape, rate = 1/scale, scale = 1/rate)

TVatRGamma(kap, shape, rate = 1/scale, scale = 1/rate)

mgfGamma(t, shape, rate = 1/scale, scale = 1/rate)
```

Arguments

shape	shape parameter α, must be positive.
rate	rate parameter β, must be positive.
scale	alternative parameterization to the rate parameter, scale = 1 / rate.
k	kth-moment.
d	cut-off value.
less.than.d	logical; if TRUE (default) truncated mean for values <= d, otherwise, for values > d.
kap	probability.
t	t.

Details

The Gamma distribution with shape parameter α and rate parameter β has density:

$$f(x) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} \mathrm{e}^{-\beta x}$$

for $x \in \mathcal{R}^+$, $\beta, \alpha > 0$.

Value

Function :

- expValGamma gives the expected value.
- varGamma gives the variance.
- kthMomentGamma gives the kth moment.
- expValLimGamma gives the limited mean.
- expValTruncGamma gives the truncated mean.
- stopLossGamma gives the stop-loss.
- meanExcessGamma gives the mean excess loss.
- VatRGamma gives the Value-at-Risk.
- TVatRGamma gives the Tail Value-at-Risk.
- mgfGamma gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRGamma is a wrapper for the qgamma function stats package.

Examples

```
# With scale parameter
expValGamma(shape = 3, scale = 4)

# With rate parameter
expValGamma(shape = 3, rate = 0.25)


# With scale parameter
varGamma(shape = 3, scale = 4)

# With rate parameter
varGamma(shape = 3, rate = 0.25)


# With scale parameter
kthMomentGamma(k = 2, shape = 3, scale = 4)
```

```
# With rate parameter
kthMomentGamma(k = 2, shape = 3, rate = 0.25)


# With scale parameter
expValLimGamma(d = 2, shape = 3, scale = 4)

# With rate parameter
expValLimGamma(d = 2, shape = 3, rate = 0.25)


# With scale parameter
expValTruncGamma(d = 2, shape = 3, scale = 4)

# With rate parameter
expValTruncGamma(d = 2, shape = 3, rate = 0.25)

# values greather than d
expValTruncGamma(d = 2, shape = 3, rate = 0.25, less.than.d = FALSE)


# With scale parameter
stopLossGamma(d = 2, shape = 3, scale = 4)

# With rate parameter
stopLossGamma(d = 2, shape = 3, rate = 0.25)


# With scale parameter
meanExcessGamma(d = 2, shape = 3, scale = 4)

# With rate parameter
meanExcessGamma(d = 2, shape = 3, rate = 0.25)


# With scale parameter
VatRGamma(kap = .2, shape = 3, scale = 4)

# With rate parameter
VatRGamma(kap = .2, shape = 3, rate = 0.25)


# With scale parameter
TVatRGamma(kap = .2, shape = 3, scale = 4)

# With rate parameter
TVatRGamma(kap = .2, shape = 3, rate = 0.25)


mgfGamma(t = 1, shape = 3, rate = 5)
```

IG *Inverse Gaussian Distribution*

Description

Inverse Gaussian distribution with mean μ and shape parameter β.

Usage

```
expValIG(mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

varIG(mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

expValLimIG(d, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

expValTruncIG(
  d,
  mean,
  shape = dispersion * mean^2,
  dispersion = shape/mean^2,
  less.than.d = TRUE
)

stopLossIG(d, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

meanExcessIG(d, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

VatRIG(kap, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

TVatRIG(kap, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)

mgfIG(t, mean, shape = dispersion * mean^2, dispersion = shape/mean^2)
```

Arguments

`mean`	mean (location) parameter μ, must be positive.
`shape`	shape parameter β, must be positive
`dispersion`	alternative parameterization to the shape parameter, dispersion = 1 / rate.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The Inverse Gaussian distribution with

Value

Function :

- expValIG gives the expected value.
- varIG gives the variance.
- expValLimIG gives the limited mean.
- expValTruncIG gives the truncated mean.
- stopLossIG gives the stop-loss.
- meanExcessIG gives the mean excess loss.
- VatRIG gives the Value-at-Risk.
- TVatRIG gives the Tail Value-at-Risk.
- mgfIG gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRIG is a wrapper for the qinvgauss function from the statmod package.

Examples

```
expValIG(mean = 2, shape = 5)

varIG(mean = 2, shape = 5)

expValLimIG(d = 2, mean = 2, shape = 5)

expValTruncIG(d = 2, mean = 2, shape = 5)

stopLossIG(d = 2, mean = 2, shape = 5)

meanExcessIG(d = 2, mean = 2, shape = 5)

VatRIG(kap = 0.99, mean = 2, shape = 5)

TVatRIG(kap = 0.99, mean = 2, shape = 5)

mgfIG(t = 1, mean = 2, shape = .5)
```

`independent` *Independence Copula*

Description

Computes CDF and simulations of the independence copula.

Usage

```
cIndependent(u1, u2, ...)

crIndependent(numberSimulations = 10000, seed = 42)
```

Arguments

`u1, u2`	points at which to evaluate the copula.
`...`	other parameters.
`numberSimulations`	Number of simulations.
`seed`	Simulation seed, 42 by default.

Details

The independence copula has CDF :

$$C(u_1, u_2) = u_1 \times u_2$$

for $u_1, u_2 \in [0, 1]$.

Value

Function :

- `cIndependent` returns the value of the copula.
- `crIndependent` returns simulated values of the copula.

Examples

```
cIndependent(u1 = .76, u2 = 0.4)

crIndependent(numberSimulations = 10, seed = 42)
```

`llogis` *Loglogistic Distribution*

Description

Loglogistic distribution with shape parameter τ and scale parameter λ.

Usage

```
dLlogis(x, shape, rate = 1/scale, scale = 1/rate)

pLlogis(q, shape, rate = 1/scale, scale = 1/rate, lower.tail = TRUE)

expValLlogis(shape, rate = 1/scale, scale = 1/rate)

varLlogis(shape, rate = 1/scale, scale = 1/rate)

kthMomentLlogis(k, shape, rate = 1/scale, scale = 1/rate)

expValLimLlogis(d, shape, rate = 1/scale, scale = 1/rate)

expValTruncLlogis(d, shape, rate = 1/scale, scale = 1/rate, less.than.d = TRUE)

stopLossLlogis(d, shape, rate = 1/scale, scale = 1/rate)

meanExcessLlogis(d, shape, rate = 1/scale, scale = 1/rate)

VatRLlogis(kap, shape, rate = 1/scale, scale = 1/rate)

TVatRLlogis(kap, shape, rate = 1/scale, scale = 1/rate)
```

Arguments

`x, q`	vector of quantiles.
`shape`	shape parameter τ, must be positive
`rate`	rate parameter β, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.

Details

The loglogistic distribution with shape parameter τ and scale parameter λ has density:

$$\frac{\tau \lambda^\tau x^{\tau-1}}{(\lambda^\tau + x^\tau)^2}$$

for $x \in \mathcal{R}^+$, $\lambda, \tau > 0$.

Value

Function :

- `dLlogis` gives the probability density function (PDF).
- `pLlogis` gives the cumulative density function (CDF).
- `expValLlogis` gives the expected value.
- `varLlogis` gives the variance.
- `kthMomentLlogis` gives the kth moment.
- `expValLimLlogis` gives the limited mean.
- `expValTruncLlogis` gives the truncated mean.
- `stopLossLlogis` gives the stop-loss.
- `meanExcessLlogis` gives the mean excess loss.
- `VatRLlogis` gives the Value-at-Risk.
- `TVatRLlogis` gives the Tail Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
dLlogis(x = 2, shape = 2, scale = 4)

# With scale parameter
pLlogis(q = 3, shape = 3, scale = 5)

# With rate parameter
pLlogis(q = 3, shape = 3, rate = 0.2)

# Survival function
pLlogis(q = 3, shape = 3, rate = 0.2, lower.tail = FALSE)

expValLlogis(shape = 2, scale = 4)

varLlogis(shape = 3, scale = 4)

kthMomentLlogis(k = 3, shape = 5, scale = 4)

expValLimLlogis(d = 2, shape = 2, scale = 4)

# With rate parameter
```

```
expValTruncLlogis(d = 2, shape = 2, scale = 4)

# Values greater than d
expValTruncLlogis(d = 2, shape = 2, scale = 4, less.than.d = FALSE)

stopLossLlogis(d = 2, shape = 2, scale = 4)

meanExcessLlogis(d = 3, shape = 2, scale = 4)

# With scale parameter
VatRLlogis(kap = .2, shape = 2, scale = 4)

# With rate parameter
VatRLlogis(kap = .2, shape = 2, rate = 0.25)

# With scale parameter
TVatRLlogis(kap = .2, shape = 3, scale = 4)

# With rate parameter
TVatRLlogis(kap = .2, shape = 3, rate = 0.25)
```

Lnorm *Lognormal Distribution*

Description

Lognormal distribution with mean μ and variance σ.

Usage

```
expValLnorm(meanlog, sdlog)

varLnorm(meanlog, sdlog)

kthMomentLnorm(k, meanlog, sdlog)

expValLimLnorm(d, meanlog, sdlog)

expValTruncLnorm(d, meanlog, sdlog, less.than.d = TRUE)

stopLossLnorm(d, meanlog, sdlog)

meanExcessLnorm(d, meanlog, sdlog)

VatRLnorm(kap, meanlog, sdlog)

TVatRLnorm(kap, meanlog, sdlog)
```

Arguments

`meanlog`	location parameter μ.
`sdlog`	standard deviation σ, must be positive.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.

Details

The Log-normal distribution with mean μ and standard deviation σ has density:

$$\frac{1}{\sqrt{2\pi}\sigma x}\mathrm{e}^{-\frac{1}{2}\left(\frac{\ln(x)-\mu}{\sigma}\right)^2}$$

for $x \in \mathcal{R}^+$, $\mu \in \mathcal{R}, \sigma > 0$.

Value

Function :

- `expValLnorm` gives the expected value.
- `varLnorm` gives the variance.
- `kthMomentLnorm` gives the kth moment.
- `expValLimLnorm` gives the limited mean.
- `expValTruncLnorm` gives the truncated mean.
- `stopLossLnorm` gives the stop-loss.
- `meanExcessLnorm` gives the mean excess loss.
- `VatRLnorm` gives the Value-at-Risk.
- `TVatRLnorm` gives the Tail Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRLnorm is a wrapper of the `qlnorm` function from the stats package.

Examples

```
expValLnorm(meanlog = 3, sdlog = 5)

varLnorm(meanlog = 3, sdlog = 5)

kthMomentLnorm(k = 2, meanlog = 3, sdlog = 5)

expValLimLnorm(d = 2, meanlog = 2, sdlog = 5)
```

```
expValTruncLnorm(d = 2, meanlog = 2, sdlog = 5)

# Values greater than d
expValTruncLnorm(d = 2, meanlog = 2, sdlog = 5, less.than.d = FALSE)

stopLossLnorm(d = 2, meanlog = 2, sdlog = 5)

meanExcessLnorm(d = 2, meanlog = 2, sdlog = 5)

VatRLnorm(kap = 0.8, meanlog = 3, sdlog = 5)

TVatRLnorm(kap = 0.8, meanlog = 2, sdlog = 5)
```

`Logarithmic` *Logarithmic Distribution*

Description

Logarithmic distribution with probability parameter γ.

Usage

```
dLogarithmic(x, prob)

pLogarithmic(q, prob, lower.tail = TRUE)

expValLogarithmic(prob)

varLogarithmic(prob)

VatRLogarithmic(kap, prob)

mgfLogarithmic(t, prob)

pgfLogarithmic(t, prob)
```

Arguments

`x, q`	vector of quantiles.
`prob`	probability parameter γ.
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`kap`	probability.
`t`	t.

Details

The Logarithmic distribution with probability parameter γ has probability mass function :

$$Pr(X = k) = \frac{-\gamma^k}{\ln(1-\gamma)k}$$

, for $k = 0, 1, 2, \ldots$, and $\gamma \in (0, 1)]$.

Value

Function :

- `dLogarithmic` gives the probability density function (PDF).
- `pLogarithmic` gives the cumulative density function (CDF).
- `expValLogarithmic` gives the expected value.
- `varLogarithmic` gives the variance.
- `VatRLogarithmic` gives the Value-at-Risk.
- `mgfLogarithmic` gives the moment generating function (MGF).
- `pgfLogarithmic` gives the probability generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
dLogarithmic(x = 3, prob = 0.2)

pLogarithmic(q = 3, prob = 0.2)

expValLogarithmic(prob = 0.50)

varLogarithmic(prob = 0.50)

VatRLogarithmic(kap = 0.99, prob = 0.2)

mgfLogarithmic(t = .2, prob = 0.50)

pgfLogarithmic(t = .2, prob = 0.50)
```

`NBinom` *Negative Binomial Distribution*

Description

Negative binomial distribution with parameters r (number of successful trials) and p (probability of success).

Usage

```
expValNBinom(
  size,
  prob = (1/(1 + beta)),
  beta = ((1 - prob)/prob),
  nb_tries = FALSE
)

varNBinom(
  size,
  prob = (1/(1 + beta)),
  beta = ((1 - prob)/prob),
  nb_tries = FALSE
)

mgfNBinom(
  t,
  size,
  prob = (1/(1 + beta)),
  beta = ((1 - prob)/prob),
  nb_tries = FALSE
)

pgfNBinom(
  t,
  size,
  prob = (1/(1 + beta)),
  beta = ((1 - prob)/prob),
  nb_tries = FALSE
)
```

Arguments

size	Number of successful trials.
prob	Probability of success in each trial.
beta	Alternative parameterization of the negative binomial distribution where beta = (1 - p) / p.
nb_tries	logical; if FALSE (default) number of trials until the rth success, otherwise, number of failures until the rth success.
t	t.

Details

When k is the number of failures until the rth success, with a probability p of a success, the negative binomial has density:

$$\left(\frac{r+k-1}{k}\right)(p)^r(1-p)^k$$

for $k \in \{0, 1, \dots\}$

When k is the number of trials until the rth success, with a probability p of a success, the negative binomial has density:

$$\binom{k-1}{r-1} (p)^r (1-p)^{k-r}$$

for $k \in \{r, r+1, r+2, \dots\}$

The alternative parameterization of the negative binomial with parameter β, and k being the number of trials, has density:

$$\frac{\Gamma(r+k)}{\Gamma(r)k!} \left(\frac{1}{1+\beta}\right)^r \left(\frac{\beta}{1+\beta}\right)^{k-r}$$

for $k \in \{0, 1, \dots\}$

Value

Function :

- `expValNBinom` gives the expected value.
- `varNBinom` gives the variance.
- `mgfNBinom` gives the moment generating function (MGF).
- `pgfNBinom` gives the probability generating function (PGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
# Where k is the number of trials for a rth success
expValNBinom(size = 2, prob = .4)

# Where k is the number of failures before a rth success
expValNBinom(size = 2, prob = .4, nb_tries = TRUE)

# With alternative parameterization where k is the number of trials
expValNBinom(size = 2, beta = 1.5)


# Where k is the number of trials for a rth success
varNBinom(size = 2, prob = .4)

# Where k is the number of failures before a rth success
varNBinom(size = 2, prob = .4, nb_tries = TRUE)

# With alternative parameterization where k is the number of trials
varNBinom(size = 2, beta = 1.5)

mgfNBinom(t = 1, size = 4, prob = 0.5)

pgfNBinom(t = 5, size = 3, prob = 0.3)
```

Norm *Normal Distribution*

Description

Normal distribution

Usage

```
expValNorm(mean, sd)

varNorm(mean, sd)

expValLimNorm(d, mean = 0, sd = 1)

expValTruncNorm(d, mean = 0, sd = 1, less.than.d = TRUE)

stopLossNorm(d, mean = 0, sd = 1)

meanExcessNorm(d, mean = 0, sd = 1)

VatRNorm(kap, mean = 0, sd = 1)

TVatRNorm(kap, mean = 0, sd = 1)

mgfNorm(t, mean = 0, sd = 1)
```

Arguments

`mean`	mean (location) parameter μ.
`sd`	standard deviation σ, must be positive.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The Normal distribution with mean μ and standard deviation σ has density:

$$\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

for $x \in \mathcal{R}$, $\mu \in \mathcal{R}, \sigma > 0$.

Value

Function :

- expValNorm gives the expected value.
- varNorm gives the variance.
- expValLimNorm gives the limited mean.
- expValTruncNorm gives the truncated mean.
- stopLossNorm gives the stop-loss.
- meanExcessNorm gives the mean excess loss.
- VatRNorm gives the Value-at-Risk.
- TVatRNorm gives the Tail Value-at-Risk.
- mgfNorm gives the moment generating function (MGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Note

Function VatRNorm is a wrapper of the qnorm function from the stats package.

Examples

```
expValNorm(mean = 3, sd = 5)

varNorm(mean = 3, sd = 5)

expValLimNorm(d = 2, mean = 2, sd = 5)

expValTruncNorm(d = 2, mean = 2, sd = 5)

stopLossNorm(d = 2, mean = 2, sd = 5)

meanExcessNorm(d = 2, mean = 2, sd = 5)

VatRNorm(kap = 0.8, mean = 3, sd = 5)

TVatRNorm(kap = 0.8, mean = 2, sd = 5)

mgfNorm(t = 1, mean = 3, sd = 5)
```

Pareto *Pareto Distribution*

Description

Pareto distribution with shape parameter α and rate parameter λ.

Usage

```
dPareto(x, shape, rate = 1/scale, scale = 1/rate)

pPareto(q, shape, rate = 1/scale, scale = 1/rate, lower.tail = TRUE)

expValPareto(shape, rate = 1/scale, scale = 1/rate)

varPareto(shape, rate = 1/scale, scale = 1/rate)

kthMomentPareto(k, shape, rate = 1/scale, scale = 1/rate)

expValLimPareto(d, shape, rate = 1/scale, scale = 1/rate)

expValTruncPareto(d, shape, rate = 1/scale, scale = 1/rate, less.than.d = TRUE)

stopLossPareto(d, shape, rate = 1/scale, scale = 1/rate)

meanExcessPareto(d, shape, rate = 1/scale, scale = 1/rate)

VatRPareto(kap, shape, rate = 1/scale, scale = 1/rate)

TVatRPareto(kap, shape, rate = 1/scale, scale = 1/rate)
```

Arguments

`x, q`	vector of quantiles.
`shape`	shape parameter α, must be positive.
`rate`	rate parameter λ, must be positive.
`scale`	alternative parameterization to the rate parameter, scale = 1 / rate.
`lower.tail`	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`k`	kth-moment.
`d`	cut-off value.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.

Details

The Pareto distribution with rate parameter λ as well as shape parameter α has density:

$$f\left(x\right) = \frac{\alpha \lambda^{\alpha}}{(\lambda + x)^{\alpha+1}}$$

for $x \in \mathcal{R}^{+}$, $\alpha, \lambda > 0$.

Value

Function :

- dPareto gives the probability density function (PDF).
- pPareto gives the cumulative density function (CDF).
- expValPareto gives the expected value.
- varPareto gives the variance.
- kthMomentPareto gives the kth moment.
- expValLimPareto gives the limited mean.
- expValTruncPareto gives the truncated mean.
- stopLossPareto gives the stop-loss.
- meanExcessPareto gives the mean excess loss.
- VatRPareto gives the Value-at-Risk.
- TVatRPareto gives the Tail Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
# With scale parameter
dPareto(x = 2, shape = 2, scale = 5)

# With rate parameter
dPareto(x = 2, shape = 2, rate = .20)

# With scale parameter
pPareto(q = 2, shape = 2, scale = 5)

# With rate parameter
pPareto(q = 2, shape = 2, rate = 0.20)

# Survival function
pPareto(q = 2, shape = 2, rate = 0.20, lower.tail = FALSE)

# With scale parameter
expValPareto(shape = 5, scale = 0.5)

# With rate parameter
expValPareto(shape = 5, rate = 2)
```

```
# With scale parameter
varPareto(shape = 5, scale = 0.5)

# With rate parameter
varPareto(shape = 5, rate = 2)

# With scale parameter
kthMomentPareto(k = 4, shape = 5, scale = 0.5)

# With rate parameter
kthMomentPareto(k = 4, shape = 5, rate = 2)

# With scale parameter
expValLimPareto(d = 4, shape = 5, scale = 0.5)

# With rate parameter
expValLimPareto(d = 4, shape = 5, rate = 2)

# With scale parameter
expValTruncPareto(d = 4, shape = 5, scale = 0.5)

# With rate parameter
expValTruncPareto(d = 4, shape = 5, rate = 2)

# With scale parameter
stopLossPareto(d = 2, shape = 5, scale = 0.5)

# With rate parameter
stopLossPareto(d = 2, shape = 5, rate = 2)

# With scale parameter
meanExcessPareto(d = 6, shape = 5, scale = 0.5)

# With rate parameter
meanExcessPareto(d = 6, shape = 5, rate = 2)

# With scale parameter
VatRPareto(kap = .99, shape = 5, scale = 0.5)

# With rate parameter
VatRPareto(kap = .99, shape = 5, rate = 2)

# With scale parameter
TVatRPareto(kap = .99, shape = 5, scale = 0.5)

# With rate parameter
TVatRPareto(kap = .99, shape = 5, rate = 2)
```

Pois *Poisson Distribution*

Description

Poisson distribution with rate parameter λ.

Usage

```
expValPois(lambda)

varPois(lambda)

expValTruncPois(d, lambda, k0, less.than.d = TRUE)

TVatRPois(kap, lambda, k0)

mgfPois(t, lambda)

pgfPois(t, lambda)
```

Arguments

`lambda`	Rate parameter λ.
`d`	cut-off value.
`k0`	point up to which to sum the distribution for the approximation.
`less.than.d`	logical; if `TRUE` (default) truncated mean for values <= d, otherwise, for values > d.
`kap`	probability.
`t`	t.

Details

The Poisson distribution with rate parameter λ has probability mass function :

$$Pr(X = k) = \frac{\lambda^k \mathrm{e}^{-\lambda}}{k!}$$

for $k = 0, 1, 2, \ldots$, and $\lambda > 0$

Value

Function :

- `expValPois` gives the expected value.
- `varPois` gives the variance.
- `expValTruncPois` gives the truncated mean.
- `TVatRPois` gives the Tail Value-at-Risk.
- `mgfPois` gives the moment generating function (MGF).
- `pgfPois` gives the probability generating function (PGF).

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
expValPois(lambda = 3)

varPois(lambda = 3)

expValTruncPois(d = 0, lambda = 2, k0 = 2E2, less.than.d = FALSE)
expValTruncPois(d = 2, lambda = 2, k0 = 2E2, less.than.d = TRUE)

TVatRPois(kap = 0.8, lambda = 3, k0 = 2E2)

mgfPois(t = 1, lambda = 3)

pgfPois(t = 1, lambda = 3)
```

`Unif` *Uniform Distribution*

Description

Uniform distribution with min a and max b.

Usage

```
expValUnif(min = 0, max = 1)

varUnif(min = 0, max = 1)

kthMomentUnif(k, min = 0, max = 1)

expValLimUnif(d, min = 0, max = 1)

expValTruncUnif(d, min = 0, max = 1, less.than.d = TRUE)

stopLossUnif(d, min = 0, max = 1)

meanExcessUnif(d, min = 0, max = 1)

VatRUnif(kap, min = 0, max = 1)

TVatRUnif(kap, min = 0, max = 1)

mgfUnif(t, min = 0, max = 1)
```

Arguments

`min, max`	lower and upper limits of the distribution. Must be finite.

k	kth-moment.
d	cut-off value.
less.than.d	logical; if TRUE (default) truncated mean for values <= d, otherwise, for values > d.
kap	probability.
t	t.

Details

The (continuous) uniform distribution with min and max parameters a and b respectively has density:

$$f(x) = \frac{1}{b-a} \times \mathbf{1}_{\{x \in [a,b]\}}$$

for $x \in [a, b]$.

Value

Function :

- expValUnif gives the expected value.
- varUnif gives the variance.
- kthMomentUnif gives the kth moment.
- expValLimUnif gives the limited mean.
- expValTruncUnif gives the truncated mean.
- stopLossUnif gives the stop-loss.
- meanExcessUnif gives the mean excess loss.
- VatRUnif gives the Value-at-Risk.
- TVatRUnif gives the Tail Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
expValUnif(min = 3, max = 4)

varUnif(min = 3, max = 4)

kthMomentUnif(k = 2, min = 3, max = 4)

expValLimUnif(d = 3, min = 2, max = 4)

expValTruncUnif(d = 3, min = 2, max = 4)

# Values greather than d
expValTruncUnif(d = 3, min = 2, max = 4, less.than.d = FALSE)

stopLossUnif(d = 3, min = 2, max = 4)
```

```
meanExcessUnif(d = 2, min = 2, max = 4)

VatRUnif(kap = .99, min = 3, max = 4)

TVatRUnif(kap = .99, min = 3, max = 4)

mgfUnif(t = 2, min = 0, max = 1)
```

---

unifDiscr *Discrete Uniform Distribution*

Description

Discrete uniform distribution with min a and max b.

Usage

```
pUnifD(q, min = 0, max = 1)

dUnifD(x, min = 0, max = 1)

varUnifD(min = 0, max = 1)

expValUnifD(min = 0, max = 1)
```

Arguments

min, max	lower and upper limits of the distribution. Must be finite.
x, q	vector of quantiles.

Details

The (discrete) uniform distribution with min and max parameters a and b respectively has density:

$$\Pr(X = x) = \frac{1}{b - a + 1}$$

for $x \in \{a, a + 1, \ldots, b - 1, b\}$.

Value

Function :

- dUnifD gives the probability density function (PDF).
- pUnifD gives the cumulative density function (CDF).
- expValUnifD gives the expected value.
- varUnifD gives the variance.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
pUnifD(q = 0.2, min = 0, max = 1)

dUnifD(min = 0, max = 1)

varUnifD(min = 0, max = 1)

expValUnifD(min = 0, max = 1)
```

Weibull *Weibull Distribution*

Description

Weibull distribution with shape parameter τ and rate parameter β.

Usage

```
expValWeibull(shape, rate = 1/scale, scale = 1/rate)

varWeibull(shape, rate = 1/scale, scale = 1/rate)

kthMomentWeibull(k, shape, rate = 1/scale, scale = 1/rate)

expValLimWeibull(d, shape, rate = 1/scale, scale = 1/rate)

expValTruncWeibull(
  d,
  shape,
  rate = 1/scale,
  scale = 1/rate,
  less.than.d = TRUE
)

stopLossWeibull(d, shape, rate = 1/scale, scale = 1/rate)

meanExcessWeibull(d, shape, rate = 1/scale, scale = 1/rate)

VatRWeibull(kap, shape, rate = 1/scale, scale = 1/rate)

TVatRWeibull(kap, shape, rate = 1/scale, scale = 1/rate)
```

Arguments

shape	shape parameter τ, must be positive
rate	rate parameter β, must be positive.

scale	alternative parameterization to the rate parameter, scale = 1 / rate.
k	kth-moment.
d	cut-off value.
less.than.d	logical; if TRUE (default) truncated mean for values <= d, otherwise, for values > d.
kap	probability.

Details

The Weibull distribution with shape parameter τ and rate parameter β has density:

$$f(x) = \beta\tau\left(\beta x\right)^{\tau-1}\mathrm{e}^{-(\beta x)^{\tau}}$$

for $x \in \mathcal{R}^+$, $\beta > 0$, $\tau > 0$

Value

Function :

- expValWeibull gives the expected value.
- varWeibull gives the variance.
- kthMomentWeibull gives the kth moment.
- expValLimWeibull gives the limited mean.
- expValTruncWeibull gives the truncated mean.
- stopLossWeibull gives the stop-loss.
- meanExcessWeibull gives the mean excess loss.
- VatRWeibull gives the Value-at-Risk.
- TVatRWeibull gives the Tail Value-at-Risk.

Invalid parameter values will return an error detailing which parameter is problematic.

Examples

```
# With scale parameter
expValWeibull(shape = 2, scale = 5)

# With rate parameter
expValWeibull(shape = 2, rate = 0.2)

# With scale parameter
varWeibull(shape = 2, scale = 5)

# With rate parameter
varWeibull(shape = 2, rate = 0.2)

# With scale parameter
kthMomentWeibull(k = 2, shape = 2, scale = 5)
```

```
# With rate parameter
kthMomentWeibull(k = 2, shape = 2, rate = 0.2)

# With scale parameter
expValLimWeibull(d = 2, shape = 2, scale = 5)

# With rate parameter
expValLimWeibull(d = 2, shape = 2, rate = 0.2)

# With scale parameter
expValTruncWeibull(d = 2, shape = 2, scale = 5)

# With rate parameter
expValTruncWeibull(d = 2, shape = 2, rate = 0.2)

# Mean of values greater than d
expValTruncWeibull(d = 2, shape = 2, rate = 0.2, less.than.d = FALSE)

# With scale parameter
stopLossWeibull(d = 2, shape = 3, scale = 4)

# With rate parameter
stopLossWeibull(d = 2, shape = 3, rate = 0.25)

# With scale parameter
meanExcessWeibull(d = 2, shape = 3, scale = 4)

# With rate parameter
meanExcessWeibull(d = 2, shape = 3, rate = 0.25)

# With scale parameter
VatRWeibull(kap = .2, shape = 3, scale = 4)

# With rate parameter
VatRWeibull(kap = .2, shape = 3, rate = 0.25)

# With scale parameter
TVatRWeibull(kap = .2, shape = 3, scale = 4)

# With rate parameter
TVatRWeibull(kap = .2, shape = 3, rate = 0.25)
```

Index