Package ‘actuar’

November 8, 2023

Type Package
Title Actuarial Functions and Heavy Tailed Distributions
Version 3.3-4
Date 2023-11-07
Description Functions and data sets for actuarial science:
modeling of loss distributions; risk theory and ruin theory;
simulation of compound models, discrete mixtures and compound
hierarchical models; credibility theory. Support for many additional
probability distributions to model insurance loss size and
frequency: 23 continuous heavy tailed distributions; the
Poisson-inverse Gaussian discrete distribution; zero-truncated and
zero-modified extensions of the standard discrete distributions.
Support for phase-type distributions commonly used to compute ruin
Implementation of the Feller-Pareto family of distributions:
<doi:10.18637/jss.v103.i06>.
Depends R (>= 4.1.0)
Imports stats, graphics, expint
LinkingTo expint
Suggests MASS
License GPL (>= 2)
URL https://gitlab.com/vigou3/actuar
BugReports https://gitlab.com/vigou3/actuar/-/issues
Encoding UTF-8
LazyData yes
Classification/MSC-2010 62P05, 91B30, 62G32
NeedsCompilation yes
Author Vincent Goulet [cre, aut],
Sébastien Auclair [ctb],
Christophe Dutang [aut],
Walter Garcia-Fontes [ctb],
Nicholas Langevin [ctb],
Xavier Milhaud [ctb],
Tommy Ouellet [ctb],
Alexandre Parent [ctb],
Mathieu Pigeon [aut],
Louis-Philippe Pouliot [ctb],
Jeffrey A. Ryan [aut] (Package API),
Robert Gentleman [aut] (Parts of the R to C interface),
Ross Ihaka [aut] (Parts of the R to C interface),
R Core Team [aut] (Parts of the R to C interface),
R Foundation [aut] (Parts of the R to C interface)

Maintainer Vincent Goulet <vincent.goulet@act.ulaval.ca>

Repository CRAN

Date/Publication 2023-11-08 05:00:02 UTC

R topics documented:

- actuar-package ................................................. 4
- adjCoef .................................................... 5
- aggregateDist .............................................. 8
- betaInt ..................................................... 13
- BetaMoments ............................................... 15
- Burr ........................................................ 16
- ChisqSupp ................................................. 18
- cm .......................................................... 19
- coverage .................................................... 25
- CTE .......................................................... 27
- dental ....................................................... 29
- discretize .................................................. 29
- elev ......................................................... 32
- emm .......................................................... 33
- ExponentialSupp .......................................... 35
- Extract.grouped.data ...................................... 36
- FellerPareto .............................................. 37
- GammaSupp ............................................... 40
- gdental ..................................................... 41
- GeneralizedBeta .......................................... 41
- GeneralizedPareto ....................................... 43
- grouped.data ............................................ 46
- Gumbel ..................................................... 48
- hachemeister .............................................. 50
- hist.grouped.data ......................................... 50
- InverseBurr ............................................... 52
- InverseExponential ....................................... 54
- InverseGamma ............................................. 55
- InverseGaussian ......................................... 57
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>InverseParalogistic</td>
<td>60</td>
</tr>
<tr>
<td>InversePareto</td>
<td>61</td>
</tr>
<tr>
<td>InverseTransformedGamma</td>
<td>63</td>
</tr>
<tr>
<td>InverseWeibull</td>
<td>65</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>67</td>
</tr>
<tr>
<td>Loggamma</td>
<td>69</td>
</tr>
<tr>
<td>Loglogistic</td>
<td>70</td>
</tr>
<tr>
<td>LognormalMoments</td>
<td>72</td>
</tr>
<tr>
<td>mde</td>
<td>73</td>
</tr>
<tr>
<td>mean.grouped.data</td>
<td>75</td>
</tr>
<tr>
<td>NormalSupp</td>
<td>76</td>
</tr>
<tr>
<td>ogive</td>
<td>77</td>
</tr>
<tr>
<td>Paralogistic</td>
<td>79</td>
</tr>
<tr>
<td>Pareto</td>
<td>81</td>
</tr>
<tr>
<td>Pareto2</td>
<td>83</td>
</tr>
<tr>
<td>Pareto3</td>
<td>85</td>
</tr>
<tr>
<td>Pareto4</td>
<td>87</td>
</tr>
<tr>
<td>PhaseType</td>
<td>89</td>
</tr>
<tr>
<td>PoissonInverseGaussian</td>
<td>91</td>
</tr>
<tr>
<td>quantile.aggregateDist</td>
<td>94</td>
</tr>
<tr>
<td>quantile.grouped.data</td>
<td>95</td>
</tr>
<tr>
<td>rcomphierarc</td>
<td>96</td>
</tr>
<tr>
<td>rcomphierarc.summaries</td>
<td>99</td>
</tr>
<tr>
<td>rcompound</td>
<td>101</td>
</tr>
<tr>
<td>rmixture</td>
<td>103</td>
</tr>
<tr>
<td>ruin</td>
<td>105</td>
</tr>
<tr>
<td>severity</td>
<td>107</td>
</tr>
<tr>
<td>SingleParameterPareto</td>
<td>108</td>
</tr>
<tr>
<td>TransformedBeta</td>
<td>110</td>
</tr>
<tr>
<td>TransformedGamma</td>
<td>112</td>
</tr>
<tr>
<td>UniformSupp</td>
<td>114</td>
</tr>
<tr>
<td>unroll</td>
<td>115</td>
</tr>
<tr>
<td>VaR</td>
<td>116</td>
</tr>
<tr>
<td>var</td>
<td>117</td>
</tr>
<tr>
<td>WeibullMoments</td>
<td>118</td>
</tr>
<tr>
<td>ZeroModifiedBinomial</td>
<td>120</td>
</tr>
<tr>
<td>ZeroModifiedGeometric</td>
<td>122</td>
</tr>
<tr>
<td>ZeroModifiedLogarithmic</td>
<td>124</td>
</tr>
<tr>
<td>ZeroModifiedNegativeBinomial</td>
<td>126</td>
</tr>
<tr>
<td>ZeroModifiedPoisson</td>
<td>128</td>
</tr>
<tr>
<td>ZeroTruncatedBinomial</td>
<td>130</td>
</tr>
<tr>
<td>ZeroTruncatedGeometric</td>
<td>132</td>
</tr>
<tr>
<td>ZeroTruncatedNegativeBinomial</td>
<td>134</td>
</tr>
<tr>
<td>ZeroTruncatedPoisson</td>
<td>136</td>
</tr>
</tbody>
</table>

#### Index

| Index | 139 |
Actuarial Functions and Heavy Tailed Distributions

Description


Details

`actuar` provides additional actuarial science functionality and support for heavy tailed distributions to the R statistical system.

The current feature set of the package can be split into five main categories.

1. Additional probability distributions: 23 continuous heavy tailed distributions from the Feller-Pareto and Transformed Gamma families, the loggamma, the Gumbel, the inverse Gaussian and the generalized beta; phase-type distributions; the Poisson-inverse Gaussian discrete distribution; zero-truncated and zero-modified extensions of the standard discrete distributions; computation of raw moments, limited moments and the moment generating function (when it exists) of continuous distributions. See the “distributions” package vignette for details.

2. Loss distributions modeling: extensive support of grouped data; functions to compute empirical raw and limited moments; support for minimum distance estimation using three different measures; treatment of coverage modifications (deductibles, limits, inflation, coinsurance). See the “modeling” and “coverage” package vignettes for details.

3. Risk and ruin theory: discretization of the claim amount distribution; calculation of the aggregate claim amount distribution; calculation of the adjustment coefficient; calculation of the probability of ruin, including using phase-type distributions. See the “risk” package vignette for details.

4. Simulation of discrete mixtures, compound models (including the compound Poisson), and compound hierarchical models. See the “simulation” package vignette for details.

5. Credibility theory: function `cm` fits hierarchical (including Bühlmann, Bühlmann-Straub), regression and linear Bayes credibility models. See the “credibility” package vignette for details.

Author(s)

Christophe Dutang, Vincent Goulet, Mathieu Pigeon and many other contributors; use `packageDescription("actuar")` for the complete list.

Maintainer: Vincent Goulet.
adjCoef

References


See Also

For probability distributions support functions, use as starting points: FellerPareto, TransformedGamma, Loggamma, Gumbel, InverseGaussian, PhaseType, PoissonInverseGaussian and, e.g., ZeroTruncatedPoisson, ZeroModifiedPoisson.

For loss modeling support functions: grouped.data, ogive, emm, elev, mde, coverage.

For risk and ruin theory functions: discretize, aggregateDist, adjCoef, ruin.

For credibility theory functions and datasets: cm, hachemeister.

Examples

## The package comes with extensive demonstration scripts;
## use the following command to obtain the list.
## Not run: demo(package = "actuar")

adjCoef

| adjCoef | Adjustment Coefficient |

Description

Compute the adjustment coefficient in ruin theory, or return a function to compute the adjustment coefficient for various reinsurance retentions.

Usage

adjCoef(mgf.claim, mgf.wait = mgfexp, premium.rate, upper.bound, h, reinsurance = c("none", "proportional", "excess-of-loss"), from, to, n = 101)

plot(x, xlab = "x", ylab = "R(x)", main = "Adjustment Coefficient", sub = comment(x), type = "l", add = FALSE, ...)

adjCoef

adjCoef

adjCoef

adjCoef

adjCoef

adjCoef
Arguments

- `mgf.claim`: an expression written as a function of `x` or of `x` and `y`, or alternatively the name of a function, giving the moment generating function (mgf) of the claim severity distribution.
- `mgf.wait`: an expression written as a function of `x`, or alternatively the name of a function, giving the mgf of the claims interarrival time distribution. Defaults to an exponential distribution with mean 1.
- `premium.rate`: if `reinsurance = "none"`, a numeric value of the premium rate; otherwise, an expression written as a function of `y`, or alternatively the name of a function, giving the premium rate function.
- `upper.bound`: numeric; an upper bound for the coefficient, usually the upper bound of the support of the claim severity mgf.
- `h`: an expression written as a function of `x` or of `x` and `y`, or alternatively the name of a function, giving function `h` in the Lundberg equation (see below); ignored if `mgf.claim` is provided.
- `reinsurance`: the type of reinsurance for the portfolio; can be abbreviated.
- `from, to`: the range over which the adjustment coefficient will be calculated.
- `n`: integer; the number of values at which to evaluate the adjustment coefficient.
- `x`: an object of class "adjCoef".
- `xlab, ylab`: label of the x and y axes, respectively.
- `main`: main title.
- `sub`: subtitle, defaulting to the type of reinsurance.
- `type`: 1-character string giving the type of plot desired; see `plot` for details.
- `add`: logical; if `TRUE` add to already existing plot.
- `...`: further graphical parameters accepted by `plot` or `lines`.

Details

In the typical case `reinsurance = "none"`, the coefficient of determination is the smallest (strictly) positive root of the Lundberg equation

$$h(x) = E[e^{xB-xcW}] = 1$$
on $[0, m]$, where $m = upper.bound$, $B$ is the claim severity random variable, $W$ is the claim interarrival (or wait) time random variable and $c = premium.rate$. The premium rate must satisfy the positive safety loading constraint $E[B - cW] < 0$.

With `reinsurance = "proportional"`, the equation becomes

$$h(x, y) = E[e^{xyB-xc(y)W}] = 1,$$

where $y$ is the retention rate and $c(y)$ is the premium rate function.

With `reinsurance = "excess-of-loss"`, the equation becomes

$$h(x, y) = E[e^{x \min(B,y)-xc(y)W}] = 1,$$
adjCoef

where $y$ is the retention limit and $c(y)$ is the premium rate function.
One can use argument $h$ as an alternative way to provide function $h(x)$ or $h(x, y)$. This is necessary
in cases where random variables $B$ and $W$ are not independent.
The root of $h(x) = 1$ is found by minimizing $(h(x) - 1)^2$.

Value

If reinsurance = "none", a numeric vector of length one. Otherwise, a function of class "adjCoef"
inherting from the "function" class.

Author(s)

Christophe Dutang, Vincent Goulet <vincent.goulet@act.ulaval.ca>

References

Society of Actuaries.
Centeno, M. d. L. (2002), Measuring the effects of reinsurance by the adjustment coefficient in the
Klugman, S. A., Panjer, H. H. and Willmot, G. E. (2008), Loss Models, From Data to Decisions,

Examples

## Basic example: no reinsurance, exponential claim severity and wait
times, premium rate computed with expected value principle and
# safety loading of 20%.
adjCoef(mgfexp, premium = 1.2, upper = 1)

## Same thing, giving function $h$.
h <- function(x) 1/((1 - x) * (1 + 1.2 * x))
adjCoef(h = h, upper = 1)

## Example 11.4 of Klugman et al. (2008)
mgfx <- function(x) 0.6 * exp(x) + 0.4 * exp(2 * x)
adjCoef(mgfx(x), mgfexp(x, 4), prem = 7, upper = 0.3182)

## Proportional reinsurance, same assumptions as above, reinsurer's
# safety loading of 30%.
mgfx <- function(x, y) mgfexp(x * y)
p <- function(x) 1.3 * x - 0.1
h <- function(x, a) 1/((1 - a * x) * (1 + x * p(a)))
R1 <- adjCoef(mgfx, premium = p, upper = 1, reins = "proportional",
          from = 0, to = 1, n = 11)
R2 <- adjCoef(h = h, upper = 1, reins = "p",
          from = 0, to = 1, n = 101)
R1(seq(0, 1, length = 10)) # evaluation for various retention rates
R2(seq(0, 1, length = 10)) # same
## Excess-of-loss reinsurance

```r
p <- function(x) 1.3 * levgamma(x, 2, 2) - 0.1
mgfx <- function(x, l)
  mgfgamma(x, 2, 2) * pgamma(l, 2, 2 - x) +
  exp(x * l) * pgamma(l, 2, 2, lower = FALSE)

h <- function(x, l) mgfx(x, l) * mgfexp(-x * p(l))
R1 <- adjCoef(mgfx, upper = 1, premium = p, reins = "excess-of-loss", 
  from = 0, to = 10, n = 11)
R2 <- adjCoef(h = h, upper = 1, reins = "e", 
  from = 0, to = 10, n = 101)
plot(R1)
plot(R2, col = "green", add = TRUE)
```

### aggregateDist

#### Aggregate Claim Amount Distribution

**Description**

Compute the aggregate claim amount cumulative distribution function of a portfolio over a period using one of five methods.

**Usage**

```r
aggregateDist(method = c("recursive", "convolution", "normal", 
  "npower", "simulation"), 
  model.freq = NULL, model.sev = NULL, p0 = NULL, 
  x.scale = 1, convolve = 0, moments, nb.simul, ..., 
  tol = 1e-06, maxit = 500, echo = FALSE)
```

### S3 methods

- **print(x, ...)**
- **plot(x, xlim, ylab = expression(F[S](x)),**
  ```r
  main = "Aggregate Claim Amount Distribution", 
  sub = comment(x), ...)
  ```
- **summary(object, ...)**
- **mean(x, ...)**
- **diff(x, ...)**
Arguments

- **method**: method to be used
- **model.freq**: for "recursive" method: a character string giving the name of a distribution in the \((a, b, 0)\) or \((a, b, 1)\) families of distributions. For "convolution" method: a vector of claim number probabilities. For "simulation" method: a frequency simulation model (see `rcomphierarc` for details) or NULL. Ignored with normal and npower methods.
- **model.sev**: for "recursive" and "convolution" methods: a vector of claim amount probabilities. For "simulation" method: a severity simulation model (see `rcomphierarc` for details) or NULL. Ignored with normal and npower methods.
- **p0**: arbitrary probability at zero for the frequency distribution. Creates a zero-modified or zero-truncated distribution if not NULL. Used only with "recursive" method.
- **x.scale**: value of an amount of 1 in the severity model (monetary unit). Used only with "recursive" and "convolution" methods.
- **convolve**: number of times to convolve the resulting distribution with itself. Used only with "recursive" method.
- **moments**: vector of the true moments of the aggregate claim amount distribution; required only by the "normal" or "npower" methods.
- **nb.simul**: number of simulations for the "simulation" method.
- **...**: parameters of the frequency distribution for the "recursive" method; further arguments to be passed to or from other methods otherwise.
- **tol**: the resulting cumulative distribution in the "recursive" method will get less than tol away from 1.
- **maxit**: maximum number of recursions in the "recursive" method.
- **echo**: logical; echo the recursions to screen in the "recursive" method.
- **x, object**: an object of class "aggregateDist".
- **xlim**: numeric of length 2; the \(x\) limits of the plot.
- **ylab**: label of the y axis.
- **main**: main title.
- **sub**: subtitle, defaulting to the calculation method.

Details

`aggregateDist` returns a function to compute the cumulative distribution function (cdf) of the aggregate claim amount distribution in any point.

The "recursive" method computes the cdf using the Panjer algorithm; the "convolution" method using convolutions; the "normal" method using a normal approximation; the "npower" method using the Normal Power 2 approximation; the "simulation" method using simulations. More details follow.
A function of class "aggregateDist", inheriting from the "function" class when using normal and Normal Power approximations and additionally inheriting from the "ecdf" and "stepfun" classes when other methods are used.

There are methods available to summarize (summary), represent (print), plot (plot), compute quantiles (quantile) and compute the mean (mean) of "aggregateDist" objects.

For the diff method: a numeric vector of probabilities corresponding to the probability mass function evaluated at the knots of the distribution.

**Recursive method**

The frequency distribution must be a member of the \((a,b,0)\) or \((a,b,1)\) families of discrete distributions.

To use a distribution from the \((a,b,0)\) family, `model.freq` must be one of "binomial", "geometric", "negative binomial" or "poisson", and `p0` must be NULL.

To use a zero-truncated distribution from the \((a,b,1)\) family, `model.freq` may be one of the strings above together with `p0 = 0`. As a shortcut, `model.freq` may also be one of "zero-truncated binomial", "zero-truncated geometric", "zero-truncated negative binomial", "zero-truncated poisson" or "logarithmic", and `p0` is then ignored (with a warning if non NULL).

(Note: since the logarithmic distribution is always zero-truncated. `model.freq = "logarithmic"` may be used with either `p0 = NULL` or `p0 = 0`.)

To use a zero-modified distribution from the \((a,b,1)\) family, `model.freq` may be one of standard frequency distributions mentioned above with `p0` set to some probability that the distribution takes the value 0. It is equivalent, but more explicit, to set `model.freq` to one of "zero-modified binomial", "zero-modified geometric", "zero-modified negative binomial", "zero-modified poisson" or "zero-modified logarithmic".

The parameters of the frequency distribution must be specified using names identical to the arguments of the appropriate function `dbinom`, `dgeom`, `dnbinom`, `dpois` or `dlogarithmic`. In the latter case, do take note that the parametrization of `dlogarithmic` is different from Appendix B of Klugman et al. (2012).

If the length of `p0` is greater than one, only the first element is used, with a warning.

`model.sev` is a vector of the (discretized) claim amount distribution \(X\); the first element **must** be \(f_X(0) = \Pr[X = 0]\).

The recursion will fail to start if the expected number of claims is too large. One may divide the appropriate parameter of the frequency distribution by 2\(^n\) and convolve the resulting distribution \(n = \text{convolve times}\).

Failure to obtain a cumulative distribution function less than `tol` away from 1 within `maxit` iterations is often due to too coarse a discretization of the severity distribution.

**Convolution method**

The cumulative distribution function (cdf) \(F_S(x)\) of the aggregate claim amount of a portfolio in the collective risk model is

\[
F_S(x) = \sum_{n=0}^{\infty} F_X^n(x)p_n,
\]
for $x = 0, 1, \ldots; p_n = \Pr[N = n]$ is the frequency probability mass function and $F_{X^n}^*(x)$ is the cdf of the $n$th convolution of the (discrete) claim amount random variable.

`model.freq` is vector $p_n$ of the number of claims probabilities; the first element must be $\Pr[N = 0]$.

`model.sev` is vector $f_X(x)$ of the (discretized) claim amount distribution; the first element must be $f_X(0)$.

**Normal and Normal Power 2 methods**

The Normal approximation of a cumulative distribution function (cdf) $F(x)$ with mean $\mu$ and standard deviation $\sigma$ is

$$F(x) \approx \Phi\left(\frac{x - \mu}{\sigma}\right).$$

The Normal Power 2 approximation of a cumulative distribution function (cdf) $F(x)$ with mean $\mu$, standard deviation $\sigma$ and skewness $\gamma$ is

$$F(x) \approx \Phi\left(-\frac{3}{\gamma} + \sqrt{\frac{9}{\gamma^2} + 1 + \frac{6(x - \mu)}{\gamma\sigma}}\right).$$

This formula is valid only for the right-hand tail of the distribution and skewness should not exceed unity.

**Simulation method**

This methods returns the empirical distribution function of a sample of size `nb.simul` of the aggregate claim amount distribution specified by `model.freq` and `model.sev`. `rcomphierarc` is used for the simulation of claim amounts, hence both the frequency and severity models can be mixtures of distributions.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Louis-Philippe Pouliot

**References**


**See Also**

discretize to discretize a severity distribution; `mean.aggregateDist` to compute the mean of the distribution; `quantile.aggregateDist` to compute the quantiles or the Value-at-Risk; `CTE.aggregateDist` to compute the Conditional Tail Expectation (or Tail Value-at-Risk); `rcomphierarc`. 
Examples

```r
## Convolution method (example 9.5 of Klugman et al. (2012))
fx <- c(0, 0.15, 0.2, 0.25, 0.125, 0.075, 0.05, 0.05, 0.05, 0.025, 0.025)
pn <- c(0.05, 0.1, 0.15, 0.2, 0.25, 0.15, 0.06, 0.03, 0.01)
Fs <- aggregateDist("convolution", model.freq = pn,
                   model.sev = fx, x.scale = 25)
summary(Fs)
c(Fs(0), diff(Fs(25 * 0:21))) # probability mass function
plot(Fs)

## Recursive method (example 9.10 of Klugman et al. (2012))
fx <- c(0, crossprod(c(2, 1)/3,
                     matrix(c(0.6, 0.7, 0.4, 0, 0, 0.3), 2, 3)))
Fs <- aggregateDist("recursive", model.freq = "poisson",
                   model.sev = fx, lambda = 3)
plot(Fs)
Fs(knots(Fs)) # cdf evaluated at its knots
diff(Fs) # probability mass function

## Recursive method (high frequency)
fx <- c(0, 0.15, 0.2, 0.25, 0.125, 0.075, 0.05, 0.05, 0.05, 0.025, 0.025)
## Not run: Fs <- aggregateDist("recursive", model.freq = "poisson",
##                            model.sev = fx, lambda = 1000)
## End(Not run)
Fs <- aggregateDist("recursive", model.freq = "poisson",
                   model.sev = fx, lambda = 250, convolve = 2, maxit = 1500)
plot(Fs)

## Recursive method (zero-modified distribution; example 9.11 of Klugman et al. (2012))
Fn <- aggregateDist("recursive", model.freq = "binomial",
                   model.sev = c(0.3, 0.5, 0.2), x.scale = 50,
                   p0 = 0.4, size = 3, prob = 0.3)
diff(Fn)

## Equivalent but more explicit call
aggregateDist("recursive", model.freq = "zero-modified binomial",
              model.sev = c(0.3, 0.5, 0.2), x.scale = 50,
              p0 = 0.4, size = 3, prob = 0.3)

## Recursive method (zero-truncated distribution). Using 'fx' above
## would mean that both Pr[N = 0] = 0 and Pr[X = 0] = 0, therefore
## Pr[S = 0] = 0 and recursions would not start.
fx <- discretize(pexp(x, 1), from = 0, to = 100, method = "upper")
fx[1L] # non zero
aggregateDist("recursive", model.freq = "zero-truncated poisson",
              model.sev = fx, lambda = 3, x.scale = 25, echo=TRUE)

## Normal Power approximation
Fs <- aggregateDist("npower", moments = c(200, 200, 0.5))
```
```r
## Simulation method
model.freq <- expression(data = rpois(3))
model.sev <- expression(data = rgamma(100, 2))
Fs <- aggregateDist("simulation", nb.simul = 1000,
                   model.freq, model.sev)
mean(Fs)
plot(Fs)

## Evaluation of ruin probabilities using Beekman's formula with
## Exponential(1) claim severity, Poisson(1) frequency and premium rate
## c = 1.2.
fx <- discretize(pexp(x, 1), from = 0, to = 100, method = "lower")
phi0 <- 0.2/1.2
Fs <- aggregateDist(method = "recursive", model.freq = "geometric",
                   model.sev = fx, prob = phi0)
1 - Fs(400) # approximate ruin probability
u <- 0:100
plot(u, 1 - Fs(u), type = "l", main = "Ruin probability")
```

---

**betaint**  
*The “Beta Integral”*

**Description**

The “beta integral” which is just a multiple of the non regularized incomplete beta function. This function merely provides an R interface to the C level routine. It is not exported by the package.

**Usage**

`betaint(x, a, b)`

**Arguments**

- `x`  
  vector of quantiles.
- `a, b`  
  parameters. See Details for admissible values.

**Details**

Function `betaint` computes the “beta integral”

\[
B(a, b; x) = \Gamma(a + b) \int_0^x t^{a-1} (1 - t)^{b-1} dt
\]

for \(a > 0, b \neq -1, -2, \ldots\) and \(0 < x < 1\). (Here \(\Gamma(a)\) is the function implemented by R’s `gamma()` and defined in its help.) When \(b > 0\),

\[
B(a, b; x) = \Gamma(a) \Gamma(b) I_x(a, b),
\]
where $I_x(a, b)$ is $\text{pbeta}(x, a, b)$. When $b < 0$, $b \neq -1, -2, \ldots$, and $a > 1 + [-b],$

\[
B(a, b; x) = -\Gamma(a + b) \left[ \sum_{r=0}^{\infty} \frac{(a - 1) \cdots (a - r - 1) \Gamma(a - r - 1)}{b(b + 1) \cdots (b + r) \Gamma(b + r + 1)} \cdot \frac{a^{a-r-1} (1-x)^{b+r}}{\Gamma(a-r) \Gamma(b+r+1)} \right]
\]

where $r = [-b]$.

This function is used (at the C level) to compute the limited expected value for distributions of the transformed beta family; see, for example, `levtrbeta`.

**Value**

The value of the integral.

Invalid arguments will result in return value `NaN`, with a warning.

**Note**

The need for this function in the package is well explained in the introduction of Appendix A of Klugman et al. (2012). See also chapter 6 and 15 of Abramowitz and Stegun (1972) for definitions and relations to the hypergeometric series.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**References**


**Examples**

```r
x <- 0.3
a <- 7
## case with b > 0
b <- 2
actuar::betaint(x, a, b)
gamma(a) * gamma(b) * pbeta(x, a, b) # same

## case with b < 0
b <- -2.2
r <- floor(-b) # r = 2
actuar::betaint(x, a, b)
```
BetaMoments

### "manual" calculation

\[
s <- \frac{(x^{a-1}) \cdot (1-x)^b}{b} + \frac{((a-1) \cdot x^{a-2}) \cdot (1-x)^{b+1}}{(b \cdot (b+1))} + \frac{((a-1) \cdot (a-2) \cdot x^{a-3}) \cdot (1-x)^{b+2}}{(b \cdot (b+1) \cdot (b+2))} - \gamma(a+b) \cdot s + \frac{(a-1) \cdot (a-2) \cdot (a-3) \cdot \gamma(a-r-1)}{(b \cdot (b+1) \cdot (b+2))} \cdot \gamma(b+r+1) \cdot \text{pbeta}(x, a-r-1, b+r+1)
\]

---

**BetaMoments**

**Raw and Limited Moments of the Beta Distribution**

### Description

Raw moments and limited moments for the (central) Beta distribution with parameters `shape1` and `shape2`.

### Usage

- `mbeta(order, shape1, shape2)`
- `levbeta(limit, shape1, shape2, order = 1)`

### Arguments

- `order` order of the moment.
- `limit` limit of the loss variable.
- `shape1, shape2` positive parameters of the Beta distribution.

### Details

The \( k \)th raw moment of the random variable \( X \) is \( \mathbb{E}[X^k] \) and the \( k \)th limited moment at some limit \( d \) is \( \mathbb{E}[\min(X, d)^k] \), \( k > -\alpha \).

The noncentral beta distribution is not supported.

### Value

`mbeta` gives the \( k \)th raw moment and `levbeta` gives the \( k \)th moment of the limited loss variable. Invalid arguments will result in return value `NaN`, with a warning.

### Author(s)

Vincent Goulet &lt;vincent.goulet@act.ulaval.ca&gt; and Mathieu Pigeon

### References

See Also

**Beta** for details on the beta distribution and functions [dpqr]beta.

Examples

```r
mbeta(2, 3, 4) - mbeta(1, 3, 4)^2
levbeta(10, 3, 4, order = 2)
```

---

**Burr**

*The Burr Distribution*

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Burr distribution with parameters shape1, shape2 and scale.

Usage

```r
dburr(x, shape1, shape2, rate = 1, scale = 1/rate, log = FALSE)
pburr(q, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qburr(p, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rburr(n, shape1, shape2, rate = 1, scale = 1/rate)
mburr(order, shape1, shape2, rate = 1, scale = 1/rate)
levburr(limit, shape1, shape2, rate = 1, scale = 1/rate, order = 1)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `shape1, shape2, scale` parameters. Must be strictly positive.
- `rate` an alternative way to specify the scale.
- `log, log.p` logical; if TRUE, probabilities/densities `p` are returned as `log(p)`.
- `lower.tail` logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- `order` order of the moment.
- `limit` limit of the loss variable.
Details

The Burr distribution with parameters \( \text{shape1} = \alpha \), \( \text{shape2} = \gamma \) and \( \text{scale} = \theta \) has density:

\[
f(x) = \frac{\alpha \gamma (x/\theta)^{\gamma}}{x[1 + (x/\theta)^{\gamma}]^{\alpha+1}}
\]

for \( x > 0 \), \( \alpha > 0 \), \( \gamma > 0 \) and \( \theta > 0 \).

The Burr is the distribution of the random variable

\[\theta \left( \frac{X}{1 - X} \right)^{1/\gamma},\]

where \( X \) has a beta distribution with parameters 1 and \( \alpha \).

The Burr distribution has the following special cases:

- A Loglogistic distribution when \( \text{shape1} == 1 \);
- A Paralogistic distribution when \( \text{shape2} == \text{shape1} \);
- A Pareto distribution when \( \text{shape2} == 1 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], -\gamma < k < \alpha \gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k], k > -\gamma \) and \( \alpha - k/\gamma \) not a negative integer.

Value

dburr gives the density, pburr gives the distribution function, qburr gives the quantile function, rburr generates random deviates, mburr gives the \( k \)th moment, and levburr gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levburr computes the limited expected value using betaint.

Distribution also known as the Burr Type XII or Singh-Maddala distribution. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also
dpareto4 for an equivalent distribution with a location parameter.

Examples

\[
\exp(dburr(1, 2, 3, \log = \text{TRUE}))
\]
\[
p <- (1:10)/10
pburr(qburr(p, 2, 3, 2), 2, 3, 2)
\]

## variance
\[
\text{mburr}(2, 2, 3, 1) - \text{mburr}(1, 2, 3, 1)^2
\]

## case with shape1 - order/shape2 > 0
\[
\text{levburr}(10, 2, 3, 1, \text{order} = 2)
\]

## case with shape1 - order/shape2 < 0
\[
\text{levburr}(10, 1.5, 0.5, 1, \text{order} = 2)
\]
cm

Value
mchisq gives the \(k\)th raw moment, levchisq gives the \(k\)th moment of the limited loss variable, and mgfchisq gives the moment generating function in \(t\).
Invalid arguments will result in return value NaN, with a warning.

Author(s)
Christophe Dutang, Vincent Goulet <vincent.goulet@act.ulaval.ca>

References

See Also
Chisquare

Examples
mchisq(2, 3, 4)
levchisq(10, 3, order = 2)
mgfchisq(0.25, 3, 2)

cm

Credibility Models

Description
Fit the following credibility models: Bühlmann, Bühlmann-Straub, hierarchical, regression (Hachemeister) or linear Bayes.

Usage

```r
cm(formula, data, ratios, weights, subset,
   regformula = NULL, regdata, adj.intercept = FALSE,
   method = c("Buhlmann-Gisler", "Ohlsson", "iterative"),
   likelihood, ...,
   tol = sqrt(.Machine$double.eps), maxit = 100, echo = FALSE)
```

```r
## S3 method for class 'cm'
print(x, ...)

## S3 method for class 'cm'
predict(object, levels = NULL, newdata, ...)
```
## S3 method for class 'cm'
summary(object, levels = NULL, newdata, ...)

## S3 method for class 'summary.cm'
print(x, ...)

### Arguments

- **formula**: character string "bayes" or an object of class "formula": a symbolic description of the model to be fit. The details of model specification are given below.
- **data**: a matrix or a data frame containing the portfolio structure, the ratios or claim amounts and their associated weights, if any.
- **ratios**: expression indicating the columns of data containing the ratios or claim amounts.
- **weights**: expression indicating the columns of data containing the weights associated with ratios.
- **subset**: an optional logical expression indicating a subset of observations to be used in the modeling process. All observations are included by default.
- **regformula**: an object of class "formula": symbolic description of the regression component (see `lm` for details). No left hand side is needed in the formula; if present it is ignored. If NULL, no regression is done on the data.
- **regdata**: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the regression model.
- **adj.intercept**: if TRUE, the intercept of the regression model is located at the barycenter of the regressor instead of the origin.
- **method**: estimation method for the variance components of the model; see Details.
- **likelihood**: a character string giving the name of the likelihood function in one of the supported linear Bayes cases; see Details.
- **tol**: tolerance level for the stopping criteria for iterative estimation method.
- **maxit**: maximum number of iterations in iterative estimation method.
- **echo**: logical; whether to echo the iterative procedure or not.
- **x, object**: an object of class "cm".
- **levels**: character vector indicating the levels to predict or to include in the summary; if NULL all levels are included.
- **newdata**: data frame containing the variables used to predict credibility regression models.
- **...**: parameters of the prior distribution for cm; additional attributes to attach to the result for the predict and summary methods; further arguments to `format` for the `print.summary` method; unused for the `print` method.

### Details

cm is the unified front end for credibility models fitting. The function supports hierarchical models with any number of levels (with Bühlmann and Bühlmann-Straub models as special cases) and the regression model of Hachemeister. Usage of cm is similar to `lm` for these cases. cm can also fit linear Bayes models, in which case usage is much simplified; see the section on linear Bayes below.
When not "bayes", the formula argument symbolically describes the structure of the portfolio in the form \textit{terms}. Each term is an interaction between risk factors contributing to the total variance of the portfolio data. Terms are separated by + operators and interactions within each term by :: For a portfolio divided first into sectors, then units and finally contracts, formula would be \(~\text{sector} + \text{sector}:\text{unit} + \text{sector}:\text{unit}:\text{contract},\) where sector, unit and contract are column names in data. In general, the formula should be of the form \(~\text{a} + \text{a:b} + \text{a:b:c} + \text{a:b:c:d} + \ldots\).

If argument \text{regformula} is not NULL, the regression model of Hachemeister is fit to the data. The response is usually time. By default, the intercept of the model is located at time origin. If argument \text{adj.intercept} is TRUE, the intercept is moved to the (collective) barycenter of time, by orthogonalization of the design matrix. Note that the regression coefficients may be difficult to interpret in this case.

Arguments \text{ratios}, \text{weights} and \text{subset} are used like arguments \text{select}, \text{select} and \text{subset}, respectively, of function \text{subset}.

Data does not have to be sorted by level. Nodes with no data (complete lines of NA except for the portfolio structure) are allowed, with the restriction mentioned above.

**Value**

Function \text{cm} computes the structure parameters estimators of the model specified in \text{formula}. The value returned is an object of class \text{cm}.

An object of class "cm" is a list with at least the following components:

- \text{means} a list containing, for each level, the vector of linearly sufficient statistics.
- \text{weights} a list containing, for each level, the vector of total weights.
- \text{unbiased} a vector containing the unbiased variance components estimators, or NULL.
- \text{iterative} a vector containing the iterative variance components estimators, or NULL.
- \text{cred} for multi-level hierarchical models: a list containing, the vector of credibility factors for each level. For one-level models: an array or vector of credibility factors.
- \text{nodes} a list containing, for each level, the vector of the number of nodes in the level.
- \text{classification} the columns of data containing the portfolio classification structure.
- \text{ordering} a list containing, for each level, the affiliation of a node to the node of the level above.

Regression fits have in addition the following components:

- \text{adj.models} a list containing, for each node, the credibility adjusted regression model as obtained with \text{lm.fit} or \text{lm.wfit}.
- \text{transition} if \text{adj.intercept} is TRUE, a transition matrix from the basis of the orthogonal design matrix to the basis of the original design matrix.
- \text{terms} the \text{terms} object used.

The method of \text{predict} for objects of class "cm" computes the credibility premiums for the nodes of every level included in argument \text{levels} (all by default). Result is a list the same length as \text{levels} or the number of levels in \text{formula}, or an atomic vector for one-level models.
Hierarchical models

The credibility premium at one level is a convex combination between the linearly sufficient statistic of a node and the credibility premium of the level above. (For the first level, the complement of credibility is given to the collective premium.) The linearly sufficient statistic of a node is the credibility weighted average of the data of the node, except at the last level, where natural weights are used. The credibility factor of node $i$ is equal to

$$\frac{w_i}{w_i + \alpha/b},$$

where $w_i$ is the weight of the node used in the linearly sufficient statistic, $\alpha$ is the average within node variance and $b$ is the average between node variance.

Regression models

The credibility premium of node $i$ is equal to

$$y' b_i^a,$$

where $y$ is a matrix created from newdata and $b_i^a$ is the vector of credibility adjusted regression coefficients of node $i$. The latter is given by

$$b_i^a = Z_i b_i + (I - Z_i)m,$$

where $b_i$ is the vector of regression coefficients based on data of node $i$ only, $m$ is the vector of collective regression coefficients, $Z_i$ is the credibility matrix and $I$ is the identity matrix. The credibility matrix of node $i$ is equal to

$$A^{-1}(A + s^2 S_i),$$

where $S_i$ is the unscaled regression covariance matrix of the node, $s^2$ is the average within node variance and $A$ is the within node covariance matrix.

If the intercept is positioned at the barycenter of time, matrices $S_i$ and $A$ (and hence $Z_i$) are diagonal. This amounts to use Bühlmann-Straub models for each regression coefficient.

Argument newdata provides the “future” value of the regressors for prediction purposes. It should be given as specified in predict.lm.

Variance components estimation

For hierarchical models, two sets of estimators of the variance components (other than the within node variance) are available: unbiased estimators and iterative estimators.

Unbiased estimators are based on sums of squares of the form

$$B_i = \sum_j w_{ij} (X_{ij} - \bar{X}_i)^2 - (J - 1)\alpha$$

and constants of the form

$$c_i = w_i - \sum_j \frac{w_{ij}^2}{w_i},$$
where $X_{ij}$ is the linearly sufficient statistic of level $(ij)$; $\bar{X}_i$ is the weighted average of the latter using weights $w_{ij}$; $\bar{X}_i = \sum_j w_{ij}$; $J$ is the effective number of nodes at level $(ij)$; $\alpha$ is the within variance of this level. Weights $w_{ij}$ are the natural weights at the lowest level, the sum of the natural weights the next level and the sum of the credibility factors for all upper levels.

The Bühlmann-Gisler estimators (method = "Buhlmann-Gisler") are given by

$$b = \frac{1}{J} \sum_i \max \left( \frac{B_i}{c_i}, 0 \right),$$

that is the average of the per node variance estimators truncated at 0.

The Ohlsson estimators (method = "Ohlsson") are given by

$$b = \frac{\sum_i B_i}{\sum_i c_i},$$

that is the weighted average of the per node variance estimators without any truncation. Note that negative estimates will be truncated to zero for credibility factor calculations.

In the Bühlmann-Straub model, these estimators are equivalent.

Iterative estimators method = "iterative" are pseudo-estimators of the form

$$b = \frac{1}{d} \sum_i w_i (X_i - \bar{X})^2,$$

where $X_i$ is the linearly sufficient statistic of one level, $\bar{X}$ is the linearly sufficient statistic of the level above and $d$ is the effective number of nodes at one level minus the effective number of nodes of the level above. The Ohlsson estimators are used as starting values.

For regression models, with the intercept at time origin, only iterative estimators are available. If method is different from "iterative", a warning is issued. With the intercept at the barycenter of time, the choice of estimators is the same as in the Bühlmann-Straub model.

**Linear Bayes**

When formula is "bayes", the function computes pure Bayesian premiums for the following combinations of distributions where they are linear credibility premiums:

- $X|\Theta = \theta \sim \text{Poisson}(\theta)$ and $\Theta \sim \text{Gamma}(\alpha, \lambda)$;
- $X|\Theta = \theta \sim \text{Exponential}(\theta)$ and $\Theta \sim \text{Gamma}(\alpha, \lambda)$;
- $X|\Theta = \theta \sim \text{Gamma}(\tau, \theta)$ and $\Theta \sim \text{Gamma}(\alpha, \lambda)$;
- $X|\Theta = \theta \sim \text{Normal}(\theta, \sigma_2^2)$ and $\Theta \sim \text{Normal}(\mu, \sigma_1^2)$;
- $X|\Theta = \theta \sim \text{Bernoulli}(\theta)$ and $\Theta \sim \text{Beta}(a, b)$;
- $X|\Theta = \theta \sim \text{Binomial}(\nu, \theta)$ and $\Theta \sim \text{Beta}(a, b)$;
- $X|\Theta = \theta \sim \text{Geometric}(\theta)$ and $\Theta \sim \text{Beta}(a, b)$;
- $X|\Theta = \theta \sim \text{Negative Binomial}(r, \theta)$ and $\Theta \sim \text{Beta}(a, b)$. 
The following combination is also supported: $X|\Theta = \theta \sim \text{Single Parameter Pareto}(\theta)$ and $\Theta \sim \Gamma(\alpha, \lambda)$. In this case, the Bayesian estimator not of the risk premium, but rather of parameter $\theta$ is linear with a “credibility” factor that is not restricted to $(0, 1)$.

Argument likelihood identifies the distribution of $X|\Theta = \theta$ as one of "poisson", "exponential", "gamma", "normal", "bernoulli", "binomial", "geometric", "negative binomial" or "pareto".

The parameters of the distributions of $X|\Theta = \theta$ (when needed) and $\Theta$ are set in ... using the argument names (and default values) of dgamma, dnorm, dbeta, dbinom, dnbinom or dpareto1, as appropriate. For the Gamma/Gamma case, use shape.lik for the shape parameter $\tau$ of the Gamma likelihood. For the Normal/Normal case, use sd.lik for the standard error $\sigma_2$ of the Normal likelihood.

Data for the linear Bayes case may be a matrix or data frame as usual; an atomic vector to fit the model to a single contract; missing or NULL to fit the prior model. Arguments ratios, weights and subset are ignored.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Xavier Milhaud, Tommy Ouellet, Louis-Philippe Pouliot

References


See Also

subset, formula, lm, predict.lm.

Examples

data(hachemeister)

## Buhlmann-Straub model
fit <- cm(~state, hachemeister, 
    ratios = ratio.1:ratio.12, weights = weight.1:weight.12)
fit # print method
predict(fit) # credibility premiums
summary(fit) # more details

## Two-level hierarchical model. Notice that data does not have
## to be sorted by level
X <- data.frame(unit = c("A", "B", "A", "B", "B"), hachemeister)
fit <- cm(~factor(unit) + unit:state, X, ratio.1:ratio.12, weight.1:weight.12)
predict(fit)
## Regression model with intercept at time origin
fit <- cm(~state, hachemeister, 
  regformula = ~time, regdata = data.frame(time = 12:1), 
  ratios = ratio.1:ratio.12, weights = weight.1:weight.12) 
fit
predict(fit, newdata = data.frame(time = 0))
summary(fit, newdata = data.frame(time = 0))

## Same regression model, with intercept at barycenter of time
fit <- cm(~state, hachemeister, adj.intercept = TRUE, 
  regformula = ~time, regdata = data.frame(time = 12:1), 
  ratios = ratio.1:ratio.12, weights = weight.1:weight.12) 
fit
predict(fit, newdata = data.frame(time = 0))
summary(fit, newdata = data.frame(time = 0))

## Poisson/Gamma pure Bayesian model
fit <- cm("bayes", data = c(5, 3, 0, 1, 1), 
  likelihood = "poisson", shape = 3, rate = 3) 
fit
predict(fit)
summary(fit)

## Normal/Normal pure Bayesian model
cm("bayes", data = c(5, 3, 0, 1, 1), 
  likelihood = "normal", sd.lik = 2, 
  mean = 2, sd = 1)

---

### coverage

#### Density and Cumulative Distribution Function for Modified Data

**Description**

Compute probability density function or cumulative distribution function of the payment per payment or payment per loss random variable under any combination of the following coverage modifications: deductible, limit, coinsurance, inflation.

**Usage**

coverage(pdf, cdf, deductible = 0, franchise = FALSE, 
  limit = Inf, coinsurance = 1, inflation = 0, 
  per.loss = FALSE)
coverage

Arguments

pdf, cdf  function object or character string naming a function to compute, respectively, the probability density function and cumulative distribution function of a probability law.
deductible  a unique positive numeric value.
franchise  logical; TRUE for a franchise deductible, FALSE (default) for an ordinary deductible.
limit  a unique positive numeric value larger than deductible.
coinsurance  a unique value between 0 and 1; the proportion of coinsurance.
inflation  a unique value between 0 and 1; the rate of inflation.
per.loss  logical; TRUE for the per loss distribution, FALSE (default) for the per payment distribution.

Details

coverage returns a function to compute the probability density function (pdf) or the cumulative distribution function (cdf) of the distribution of losses under coverage modifications. The pdf and cdf of unmodified losses are pdf and cdf, respectively.

If pdf is specified, the pdf is returned; if pdf is missing or NULL, the cdf is returned. Note that cdf is needed if there is a deductible or a limit.

Value

An object of mode "function" with the same arguments as pdf or cdf, except "lower.tail", "log.p" and "log", which are not supported.

Note

Setting arguments of the function returned by coverage using formals may very well not work as expected.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

vignette("coverage") for the exact definitions of the per payment and per loss random variables under an ordinary or franchise deductible.
Examples

```r
## Default case: pdf of the per payment random variable with
## an ordinary deductible
coverage(dgamma, pgamma, deductible = 1)

## Add a limit
f <- coverage(dgamma, pgamma, deductible = 1, limit = 7)
f <- coverage("dgamma", "pgamma", deductible = 1, limit = 7) # same
f(0, shape = 3, rate = 1)
f(2, shape = 3, rate = 1)
f(6, shape = 3, rate = 1)
f(8, shape = 3, rate = 1)
curve(dgamma(x, 3, 1), xlim = c(0, 10), ylim = c(0, 0.3)) # original
curve(f(x, 3, 1), xlim = c(0.01, 5.99), col = 4, add = TRUE) # modified
points(6, f(6, 3, 1), pch = 21, bg = 4)

## Cumulative distribution function
F <- coverage(cdf = pgamma, deductible = 1, limit = 7)
F(0, shape = 3, rate = 1)
F(2, shape = 3, rate = 1)
F(6, shape = 3, rate = 1)
F(8, shape = 3, rate = 1)
curve(pgamma(x, 3, 1), xlim = c(0, 10), ylim = c(0, 1)) # original
curve(F(x, 3, 1), xlim = c(0, 5.99), col = 4, add = TRUE) # modified
curve(F(x, 3, 1), xlim = c(6, 10), col = 4, add = TRUE) # modified

## With no deductible, all distributions below are identical
coverage(dweibull, pweibull, limit = 5)
coverage(dweibull, pweibull, per.loss = TRUE, limit = 5)
coverage(dweibull, pweibull, franchise = TRUE, limit = 5)
coverage(dweibull, pweibull, per.loss = TRUE, franchise = TRUE,
         limit = 5)

## Coinsurance alone; only case that does not require the cdf
coverage(dgamma, coinsurance = 0.8)
```

---

CTE  

Conditional Tail Expectation

Description

Conditional Tail Expectation, also called Tail Value-at-Risk.

TVaR is an alias for CTE.

Usage

```r
CTE(x, 
```

## S3 method for class 'aggregateDist'
CTE(x, conf.level = c(0.9, 0.95, 0.99),
   names = TRUE, ...)

TVaR(x, ...)

Arguments

- **x**: an R object.
- **conf.level**: numeric vector of probabilities with values in [0, 1).
- **names**: logical; if true, the result has a names attribute. Set to FALSE for speedup with many probs.
- **...**: further arguments passed to or from other methods.

Details

The Conditional Tail Expectation (or Tail Value-at-Risk) measures the average of losses above the Value at Risk for some given confidence level, that is \( E[X|X > \text{VaR}(X)] \) where \( X \) is the loss random variable.

CTE is a generic function with, currently, only a method for objects of class "aggregateDist".

For the recursive, convolution and simulation methods of aggregateDist, the CTE is computed from the definition using the empirical cdf.

For the normal approximation method, an explicit formula exists:

\[
\mu + \frac{\sigma}{(1 - \alpha)\sqrt{2\pi}} e^{-\text{VaR}(X)^2/2},
\]

where \( \mu \) is the mean, \( \sigma \) the standard deviation and \( \alpha \) the confidence level.

For the Normal Power approximation, the explicit formula given in Castañer et al. (2013) is

\[
\mu + \frac{\sigma}{(1 - \alpha)\sqrt{2\pi}} e^{-\text{VaR}(X)^2/2} \left(1 + \frac{\gamma}{6} \text{VaR}(X)\right),
\]

where, as above, \( \mu \) is the mean, \( \sigma \) the standard deviation, \( \alpha \) the confidence level and \( \gamma \) is the skewness.

Value

A numeric vector, named if names is TRUE.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Tommy Ouellet

References

See Also

aggregateDist, VaR

Examples

```r
model.freq <- expression(data = rpois(7))
model.sev <- expression(data = rnorm(9, 2))
Fs <- aggregateDist("simulation", model.freq, model.sev, nb.simul = 1000)
CTE(Fs)
```

---

### dental

**Individual Dental Claims Data Set**

**Description**

Basic dental claims on a policy with a deductible of 50.

**Usage**

dental

**Format**

A vector containing 10 observations

**Source**


---

### discretize

**Discretization of a Continuous Distribution**

**Description**

Compute a discrete probability mass function from a continuous cumulative distribution function (cdf) with various methods.

discretise is an alias for discretize.

**Usage**

```r
discretize(cdf, from, to, step = 1, method = c("upper", "lower", "rounding", "unbiased"), lev, by = step, xlim = NULL)
discretise(cdf, from, to, step = 1, method = c("upper", "lower", "rounding", "unbiased"), lev, by = step, xlim = NULL)
```
Arguments

cdf: an expression written as a function of \( x \), or alternatively the name of a function, giving the cdf to discretize.

from, to: the range over which the function will be discretized.

step: numeric; the discretization step (or span, or lag).

method: discretization method to use.

lev: an expression written as a function of \( x \), or alternatively the name of a function, to compute the limited expected value of the distribution corresponding to cdf. Used only with the "unbiased" method.

by: an alias for step.

xlim: numeric of length 2; if specified, it serves as default for c(from, to).

Details

Usage is similar to curve.

discretize returns the probability mass function (pmf) of the random variable obtained by discretization of the cdf specified in cdf.

Let \( F(x) \) denote the cdf, \( E[\min(X, x)] \) the limited expected value at \( x \), \( h \) the step, \( p_x \) the probability mass at \( x \) in the discretized distribution and set \( a = \text{from} \) and \( b = \text{to} \).

Method "upper" is the forward difference of the cdf \( F \):

\[ p_x = F(x + h) - F(x) \]

for \( x = a, a + h, \ldots, b - \text{step} \).

Method "lower" is the backward difference of the cdf \( F \):

\[ p_x = F(x) - F(x - h) \]

for \( x = a + h, \ldots, b \) and \( p_a = F(a) \).

Method "rounding" has the true cdf pass through the midpoints of the intervals \([x-h/2, x+h/2]\):

\[ p_x = F(x + h/2) - F(x - h/2) \]

for \( x = a + h, \ldots, b - \text{step} \) and \( p_a = F(a + h/2) \). The function assumes the cdf is continuous. Any adjustment necessary for discrete distributions can be done via cdf.

Method "unbiased" matches the first moment of the discretized and the true distributions. The probabilities are as follows:

\[ p_a = \frac{E[\min(X, a)] - E[\min(X, a + h)]}{h} + 1 - F(a) \]

\[ p_x = \frac{2E[\min(X, x)] - E[\min(X, x - h)] - E[\min(X, x + h)]}{h}, \quad a < x < b \]

\[ p_b = \frac{E[\min(X, b)] - E[\min(X, b - h)]}{h} - 1 + F(b) \]
discretize

Value

A numeric vector of probabilities suitable for use in `aggregateDist`.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

`aggregateDist`

Examples

```r
x <- seq(0, 5, 0.5)

op <- par(mfrow = c(1, 1), col = "black")

## Upper and lower discretization
fu <- discretize(pgamma(x, 1), method = "upper",
  from = 0, to = 5, step = 0.5)
fl <- discretize(pgamma(x, 1), method = "lower",
  from = 0, to = 5, step = 0.5)
curve(pgamma(x, 1), xlim = c(0, 5))
par(col = "blue")
plot(stepfun(head(x, -1), diffinv(fu)), pch = 19, add = TRUE)
par(col = "green")
plot(stepfun(x, diffinv(fl)), pch = 19, add = TRUE)
par(col = "black")

## Rounding (or midpoint) discretization
fr <- discretize(pgamma(x, 1), method = "rounding",
  from = 0, to = 5, step = 0.5)
curve(pgamma(x, 1), xlim = c(0, 5))
par(col = "blue")
plot(stepfun(head(x, -1), diffinv(fr)), pch = 19, add = TRUE)
par(col = "black")

## First moment matching
fb <- discretize(pgamma(x, 1), method = "unbiased",
  lev = levgamma(x, 1), from = 0, to = 5, step = 0.5)
curve(pgamma(x, 1), xlim = c(0, 5))
par(col = "blue")
plot(stepfun(x, diffinv(fb)), pch = 19, add = TRUE)

par(op)
```
Description

Compute the empirical limited expected value for individual or grouped data.

Usage

elev(x, ...)

## Default S3 method:
elev(x, ...)

## S3 method for class 'grouped.data'
elev(x, ...)

## S3 method for class 'elev'
print(x, digits =getOption("digits") - 2, ...)

## S3 method for class 'elev'
summary(object, ...)

## S3 method for class 'elev'
knots(Fn, ...)

## S3 method for class 'elev'
plot(x, ..., main = NULL, xlab = "x", ylab = "Empirical LEV")

Arguments

x
a vector or an object of class "grouped.data" (in which case only the first
column of frequencies is used); for the methods, an object of class "elev".
typically.
digits
number of significant digits to use, see print.
Fn, object
an R object inheriting from "ogive".
main
main title.
xlab, ylab
labels of x and y axis.
... arguments to be passed to subsequent methods.

Details

The limited expected value (LEV) at $u$ of a random variable $X$ is $E[X \wedge u] = E[\min(X, u)]$. For
individual data $x_1, \ldots, x_n$, the empirical LEV $E_n[X \wedge u]$ is thus

$$E_n[X \wedge u] = \frac{1}{n} \left( \sum_{x_j < u} x_j + \sum_{x_j \geq u} u \right).$$
Methods of elev exist for individual data or for grouped data created with grouped.data. The formula in this case is too long to show here. See the reference for details.

Value
For elev, a function of class "elev", inheriting from the "function" class.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References

See Also
grouped.data to create grouped data objects; stepfun for related documentation (even though the empirical LEV is not a step function).

Examples
data(gdental)
lev <- elev(gdental)
lev
summary(lev)
knots(lev) # the group boundaries
lev(knots(lev)) # empirical lev at boundaries
lev(c(80, 200, 2000)) # and at other limits
plot(lev, type = "o", pch = 16)

emn

Empirical Moments

Description
Raw empirical moments for individual and grouped data.

Usage
emn(x, order = 1, ...)

## Default S3 method:
emn(x, order = 1, ...)

## S3 method for class 'grouped.data'
emn(x, order = 1, ...)

Arguments

- **x**: a vector or matrix of individual data, or an object of class “grouped data”.
- **order**: order of the moment. Must be positive.
- **...**: further arguments passed to or from other methods.

Details

Arguments ... are passed to `colMeans`; `na.rm = TRUE` may be useful for individual data with missing values.

For individual data, the $k$th empirical moment is $\sum_{j=1}^{n} x_j^k$.

For grouped data with group boundaries $c_0, c_1, \ldots, c_r$ and group frequencies $n_1, \ldots, n_r$, the $k$th empirical moment is

$$\frac{1}{n} \sum_{j=1}^{r} n_j (c_j^{k+1} - c_{j-1}^{k+1}) (k+1)(c_j - c_{j-1})$$

where $n = \sum_{j=1}^{r} n_j$.

Value

A named vector or matrix of moments.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

`mean` and `mean.grouped.data` for simpler access to the first moment.

Examples

```r
## Individual data
data(dental)
emm(dental, order = 1:3)

## Grouped data
data(gdental)
emm(gdental)
x <- grouped.data(cj = gdental[, 1],
                  nj1 = sample(1:100, nrow(gdental)),
                  nj2 = sample(1:100, nrow(gdental)))
emm(x) # same as mean(x)
```
**Description**

Raw moments, limited moments and moment generating function for the exponential distribution with rate rate (i.e., mean 1/rate).

**Usage**

- `mexp(order, rate = 1)`
- `levexp(limit, rate = 1, order = 1)`
- `mgfexp(t, rate = 1, log = FALSE)`

**Arguments**

- `order` order of the moment.
- `limit` limit of the loss variable.
- `rate` vector of rates.
- `t` numeric vector.
- `log` logical; if TRUE, the cumulant generating function is returned.

**Details**

The $k$th raw moment of the random variable $X$ is $E[X^k]$, the $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$ and the moment generating function is $E[e^{tX}]$, $k > -1$.

**Value**

- `mexp` gives the $k$th raw moment,
- `levexp` gives the $k$th moment of the limited loss variable, and
- `mgfexp` gives the moment generating function in t.

Invalid arguments will result in return value NaN, with a warning.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Christophe Dutang and Mathieu Pigeon.

**References**


**See Also**

*Exponential*
Extract.grouped.data

Examples

\[ mexp(2, 3) - mexp(1, 3)^2 \]
\[ levexp(10, 3, \text{order} = 2) \]
\[ mgfexp(1, 2) \]

Description

Extract or replace subsets of grouped data objects.

Usage

```r
## S3 method for class 'grouped.data'
x[i, j]
## S3 replacement method for class 'grouped.data'
x[i, j] <- value
```

Arguments

- `x` an object of class `grouped.data`.
- `i, j` elements to extract or replace. `i, j` are numeric or character or, for `[` only, empty. Numeric values are coerced to integer as if by `as.integer`. For replacement by `[`, a logical matrix is allowed, but not replacement in the group boundaries and group frequencies simultaneously.
- `value` a suitable replacement value.

Details

Objects of class "grouped.data" can mostly be indexed like data frames, with the following restrictions:

1. For `[`, the extracted object must keep a group boundaries column and at least one group frequencies column to remain of class "grouped.data";
2. For `[<`, it is not possible to replace group boundaries and group frequencies simultaneously;
3. When replacing group boundaries, `length(value) == length(i) + 1`.

`x[, 1]` will return the plain vector of group boundaries.

Replacement of non adjacent group boundaries is not possible for obvious reasons.

Otherwise, extraction and replacement should work just like for data frames.

Value

For `[` an object of class "grouped.data", a data frame or a vector.
For `[<` an object of class "grouped.data".
FellerPareto

Note

Currently [[, [[<-, $ and $<- are not specifically supported, but should work as usual on group
frequency columns.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

See Also

[.data.frame for extraction and replacement methods of data frames, grouped.data to create
grouped data objects.

Examples

data(gdental)

(x <- gdental[1]) # select column 1
class(x) # no longer a grouped.data object
class(gdental[2]) # same
gdental[, 1] # group boundaries
gdental[, 2] # group frequencies
gdental[1:4,] # a subset
gdental[c(1, 3, 5),] # avoid this
gdental[1:2, 1] <- c(0, 30, 60) # modified boundaries
gdental[, 2] <- 10 # modified frequencies
## Not run: gdental[1, ] <- 2 # not allowed

FellerPareto

The Feller Pareto Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and
limited moments for the Feller Pareto distribution with parameters min, shape1, shape2, shape3 and scale.

Usage

dfpareto(x, min, shape1, shape2, shape3, rate = 1, scale = 1/rate, log = FALSE)
pfpareto(q, min, shape1, shape2, shape3, rate = 1, scale = 1/rate,
lower.tail = TRUE, log.p = FALSE)
qfpareto(p, min, shape1, shape2, shape3, rate = 1, scale = 1/rate,
lower.tail = TRUE, log.p = FALSE)
rfpareto(n, min, shape1, shape2, shape3, rate = 1, scale = 1/rate)
mfpareto(order, min, shape1, shape2, shape3, rate = 1, scale = 1/rate)
levfpareto(limit, min, shape1, shape2, shape3, rate = 1, scale = 1/rate, order = 1)

Arguments

- **x, q**: vector of quantiles.
- **p**: vector of probabilities.
- **n**: number of observations. If length(n) > 1, the length is taken to be the number required.
- **min**: lower bound of the support of the distribution.
- **shape1, shape2, shape3, scale**: parameters. Must be strictly positive.
- **rate**: an alternative way to specify the scale.
- **log, log.p**: logical; if TRUE, probabilities/densities p are returned as log(p).
- **lower.tail**: logical; if TRUE (default), probabilities are P[X \leq x], otherwise, P[X > x].
- **order**: order of the moment.
- **limit**: limit of the loss variable.

Details

The Feller-Pareto distribution with parameters \( \min = \mu, \text{shape1} = \alpha, \text{shape2} = \gamma, \text{shape3} = \tau \) and scale = \( \theta \), has density:

\[
f(x) = \frac{\Gamma(\alpha + \tau)}{\Gamma(\alpha)\Gamma(\tau)} \frac{\gamma((x - \mu)/\theta)^{\gamma-1}}{\theta[1 + ((x - \mu)/\theta)^\gamma]^{\alpha+\tau}}
\]

for \( x > \mu, -\infty < \mu < \infty, \alpha > 0, \gamma > 0, \tau > 0 \) and \( \theta > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by R’s \texttt{gamma()} and defined in its help.)

The Feller-Pareto is the distribution of the random variable

\[
\mu + \theta \left( \frac{1 - X}{X} \right)^{1/\gamma},
\]

where \( X \) has a beta distribution with parameters \( \alpha \) and \( \tau \).

The Feller-Pareto defines a large family of distributions encompassing the transformed beta family and many variants of the Pareto distribution. Setting \( \mu = 0 \) yields the transformed beta distribution.

The Feller-Pareto distribution also has the following direct special cases:

- A Pareto IV distribution when \( \text{shape3} == 1 \);
- A Pareto III distribution when \( \text{shape1} \ \text{shape3} == 1 \);
- A Pareto II distribution when \( \text{shape1} \ \text{shape2} == 1 \);
- A Pareto I distribution when \( \text{shape1} \ \text{shape2} == 1 \) and \( \min = \text{scale} \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) for nonnegative integer values of \( k < \alpha \gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k] \) for nonnegative integer values of \( k \) and \( \alpha - j/\gamma, j = 1, \ldots, k \) not a negative integer.

Note that the range of admissible values for \( k \) in raw and limited moments is larger when \( \mu = 0 \).
Value

dfpareto gives the density, pfpareto gives the distribution function, qfpareto gives the quantile function, rfpareto generates random deviates, mfpareto gives the \( k \)th raw moment, and levfpareto gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levfpareto computes the limited expected value using betaint.

For the Feller-Pareto and other Pareto distributions, we use the classification of Arnold (2015) with the parametrization of Klugman et al. (2012).

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Nicholas Langevin

References


See Also

dtrbeta for the transformed beta distribution.

Examples

```r
exp(dfpareto(2, 1, 2, 3, 4, 5, log = TRUE))
p <- (1:10)/10
pfpareto(qfpareto(p, 1, 2, 3, 4, 5), 1, 2, 3, 4, 5)

## variance
mfpareto(2, 1, 2, 3, 4, 5) - mfpareto(1, 1, 2, 3, 4, 5)^2

## case with shape1 - order/shape2 > 0
levfpareto(10, 1, 2, 3, 4, scale = 1, order = 2)

## case with shape1 - order/shape2 < 0
levfpareto(20, 10, 0.1, 14, 2, scale = 1.5, order = 2)
```
Description

Raw moments, limited moments and moment generating function for the Gamma distribution with parameters shape and scale.

Usage

\[
mgamma(order, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate})
\]
\[
levgamma(\text{limit}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \text{order} = 1)
\]
\[
mgfgamma(t, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \text{log} = \text{FALSE})
\]

Arguments

- \textbf{order} order of the moment.
- \textbf{limit} limit of the loss variable.
- \textbf{rate} an alternative way to specify the scale.
- \textbf{shape, scale} shape and scale parameters. Must be strictly positive.
- \textbf{t} numeric vector.
- \textbf{log} logical; if TRUE, the cumulant generating function is returned.

Details

The \(k\)th raw moment of the random variable \(X\) is \(E[X^k]\), the \(k\)th limited moment at some limit \(d\) is \(E[\min(X, d)^k]\) and the moment generating function is \(E[e^{tX}], k > -\alpha\).

Value

- \text{mgamma} gives the \(k\)th raw moment,
- \text{levgamma} gives the \(k\)th moment of the limited loss variable,
- \text{mgfgamma} gives the moment generating function in \(t\).

Invalid arguments will result in return value NaN, with a warning.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Christophe Dutang and Mathieu Pigeon

References


See Also

GammaDist

Examples

mgamma(2, 3, 4) - mgamma(1, 3, 4)^2
levgamma(10, 3, 4, order = 2)
mgfgamma(1, 3, 2)

---

gdental  Grouped Dental Claims Data Set

Description

Grouped dental claims, that is presented in a number of claims per claim amount group form.

Usage

gdental

Format

An object of class "grouped.data" (inheriting from class "data.frame") consisting of 10 rows and 2 columns. The environment of the object contains the plain vector of cj of group boundaries

Source


See Also

grouped.data for a description of grouped data objects.

---

GeneralizedBeta  The Generalized Beta Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Generalized Beta distribution with parameters shape1, shape2, shape3 and scale.
Usage

dgenbeta(x, shape1, shape2, shape3, rate = 1, scale = 1/rate, log = FALSE)
pgenbeta(q, shape1, shape2, shape3, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qgenbeta(p, shape1, shape2, shape3, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rgenbeta(n, shape1, shape2, shape3, rate = 1, scale = 1/rate)
mgenbeta(order, shape1, shape2, shape3, rate = 1, scale = 1/rate)
levgenbeta(limit, shape1, shape2, shape3, rate = 1, scale = 1/rate, order = 1)

Arguments

x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
shape1, shape2, shape3, scale parameters. Must be strictly positive.
rate an alternative way to specify the scale.
log, log.p logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
order order of the moment.
limit limit of the loss variable.

Details

The generalized beta distribution with parameters \( \text{shape1} = \alpha, \text{shape2} = \beta, \text{shape3} = \tau \) and \( \text{scale} = \theta \), has density:

\[
f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} (x/\theta)^{\alpha\tau} (1 - (x/\theta)^\tau)^{\beta - 1} \frac{\tau}{x}
\]

for \( 0 < x < \theta, \alpha > 0, \beta > 0, \tau > 0 \) and \( \theta > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by R’s \texttt{gamma()} \) and defined in its help.)

The generalized beta is the distribution of the random variable

\[
\theta X^{1/\tau},
\]

where \( X \) has a beta distribution with parameters \( \alpha \) and \( \beta \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) and the \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)] \), \( k > -\alpha\tau \).
Value

dgenbeta gives the density, pgenbeta gives the distribution function, qgenbeta gives the quantile function, rgenbeta generates random deviates, mgenbeta gives the $k$th raw moment, and levgenbeta gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

This is not the generalized three-parameter beta distribution defined on page 251 of Johnson et al, 1995.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


Examples

```r
exp(dgenbeta(2, 2, 3, 4, 0.2, log = TRUE))
p <- (1:10)/10
genbeta(qgenbeta(p, 2, 3, 4, 0.2), 2, 3, 4, 0.2)
mgenbeta(2, 1, 2, 3, 0.25) - mgenbeta(1, 1, 2, 3, 0.25) + 2
levgenbeta(10, 1, 2, 3, 0.25, order = 2)
```

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Generalized Pareto distribution with parameters shape1, shape2 and scale.
Usage

dgenpareto(x, shape1, shape2, rate = 1, scale = 1/rate,
          log = FALSE)
pgenpareto(q, shape1, shape2, rate = 1, scale = 1/rate,
          lower.tail = TRUE, log.p = FALSE)
qgenpareto(p, shape1, shape2, rate = 1, scale = 1/rate,
          lower.tail = TRUE, log.p = FALSE)
rgenpareto(n, shape1, shape2, rate = 1, scale = 1/rate)
mgenpareto(order, shape1, shape2, rate = 1, scale = 1/rate)
levgenpareto(limit, shape1, shape2, rate = 1, scale = 1/rate,
             order = 1)

Arguments

x, q  vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
shape1, shape2, scale
parameters. Must be strictly positive.
rate an alternative way to specify the scale.
log, log.p logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].
order order of the moment.
limit limit of the loss variable.

Details

The Generalized Pareto distribution with parameters shape1 = α, shape2 = τ and scale = θ has density:

\[ f(x) = \frac{\Gamma(\alpha + \tau)}{\Gamma(\alpha)\Gamma(\tau)} \theta^\alpha x^{\tau - 1} \left( x + \theta \right)^{-\alpha - \tau} \]

for \( x > 0, \alpha > 0, \tau > 0 \) and \( \theta > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by R’s \texttt{gamma()} and defined in its help.)

The Generalized Pareto is the distribution of the random variable

\[ \theta \left( \frac{X}{1 - X} \right) \]

where \( X \) has a beta distribution with parameters \( \alpha \) and \( \tau \).

The Generalized Pareto distribution has the following special cases:

- A Pareto distribution when \( \text{shape2} == 1 \);
- An Inverse Pareto distribution when \( \text{shape1} == 1 \).

The \( k \)th raw moment of the random variable \( X \) is\( E[X^k], -\tau < k < \alpha \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k], k > -\tau \) and \( \alpha - k \) not a negative integer.
Value
dgenpareto gives the density, pgenpareto gives the distribution function, qgenpareto gives the quantile function, rgenpareto generates random deviates, mgenpareto gives the $k$th raw moment, and levgenpareto gives the $k$th moment of the limited loss variable.
Invalid arguments will result in return value NaN, with a warning.

Note
levgenpareto computes the limited expected value using betaint.
Distribution also known as the Beta of the Second Kind. See also Kleiber and Kotz (2003) for alternative names and parametrizations.
The Generalized Pareto distribution defined here is different from the one in Embrechts et al. (1997) and in Wikipedia; see also Kleiber and Kotz (2003, section 3.12). One may most likely compute quantities for the latter using functions for the Pareto distribution with the appropriate change of parametrization.
The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References

Examples
exp(dgenpareto(3, 3, 4, 4, log = TRUE))
p <- (1:10)/10
pgenpareto(qgenpareto(p, 3, 3, 1), 3, 3, 1)
qgenpareto(.3, 3, 4, 4, lower.tail = FALSE)
## variance
mgenpareto(2, 3, 2, 1) - mgenpareto(1, 3, 2, 1)^2
## case with shape1 - order > 0
levgenpareto(10, 3, 3, scale = 1, order = 2)
## case with shape1 - order < 0
levgenpareto(10, 1.5, 3, scale = 1, order = 2)
grouped.data

Grouped data

Description

Creation of grouped data objects, from either a provided set of group boundaries and group frequencies, or from individual data using automatic or specified breakpoints.

Usage

```r
grouped.data(..., breaks = "Sturges", include.lowest = TRUE,
     right = TRUE, nclass = NULL, group = FALSE,
     row.names = NULL, check.rows = FALSE,
     check.names = TRUE)
```

Arguments

- `...`: arguments of the form `value` or `tag = value`; see Details.
- `breaks`: same as for `hist`, namely one of:
  - a vector giving the breakpoints between groups;
  - a function to compute the vector of breakpoints;
  - a single number giving the number of groups;
  - a character string naming an algorithm to compute the number of groups (see `hist`);
  - a function to compute the number of groups.
  In the last three cases the number is a suggestion only; the breakpoints will be set to `pretty` values. If `breaks` is a function, the first element in `...` is supplied to it as the only argument.
- `include.lowest`: logical; if TRUE, a data point equal to the `breaks` value will be included in the first (or last, for `right = FALSE`) group. Used only for individual data; see Details.
- `right`: logical; indicating if the intervals should be closed on the right (and open on the left) or vice versa.
- `nclass`: numeric (integer); equivalent to `breaks` for a scalar or character argument.
- `group`: logical; an alternative way to force grouping of individual data.
- `row.names`, `check.rows`, `check.names`: arguments identical to those of `data.frame`.

Details

A grouped data object is a special form of data frame consisting of one column of contiguous group boundaries and one or more columns of frequencies within each group.

The function can create a grouped data object from two types of arguments.
1. Group boundaries and frequencies. This is the default mode of operation if the call has at least two elements in ....
   The first argument will then be taken as the vector of group boundaries. This vector must be exactly one element longer than the other arguments, which will be taken as vectors of group frequencies. All arguments are coerced to data frames.

2. Individual data. This mode of operation is active if there is a single argument in ...., or if either breaks or nclass is specified or group is TRUE.
   Arguments of ... are first grouped using hist. If needed, breakpoints are set using the first argument.

Missing (NA) frequencies are replaced by zeros, with a warning.
Extraction and replacement methods exist for grouped.data objects, but working on non adjacent groups will most likely yield useless results.

Value
An object of class c("grouped.data", "data.frame") with an environment containing the vector cj of group boundaries.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>, Mathieu Pigeon and Louis-Philippe Pouliot

References

See Also
[grouped.data for extraction and replacement methods.
 data.frame for usual data frame creation and manipulation.
 hist for details on the calculation of breakpoints.

Examples

```r
## Most common usage using a predetermined set of group boundaries and group frequencies.
cj <- c(0, 25, 50, 100, 250, 500, 1000)
nj <- c(30, 31, 57, 42, 45, 10)
(x <- grouped.data(Group = cj, Frequency = nj))
class(x)

x[, 1] # group boundaries
x[, 2] # group frequencies

## Multiple frequency columns are supported
x <- sample(1:100, 9)
y <- sample(1:100, 9)
```

grouped.data(cj = 1:10, nj.1 = x, nj.2 = y)

## Alternative usage with grouping of individual data.
grouped.data(x) # automatic breakpoints
grouped.data(x, breaks = 7) # forced number of groups
grouped.data(x, breaks = c(0,25,75,100)) # specified groups

## Not run: ## Providing two or more data sets and automatic breakpoints is
## very error-prone since the range of the first data set has to
## include the ranges of all the other data sets.
range(x)
range(y)
grouped.data(x, y, group = TRUE)
## End(Not run)

---

Gumbel

*The Gumbel Distribution*

**Description**

Density function, distribution function, quantile function, random generation and raw moments for the Gumbel extreme value distribution with parameters alpha and scale.

**Usage**

- `dgumbel(x, alpha, scale, log = FALSE)`
- `pgumbel(q, alpha, scale, lower.tail = TRUE, log.p = FALSE)`
- `qgumbel(p, alpha, scale, lower.tail = TRUE, log.p = FALSE)`
- `rgumbel(n, alpha, scale)`
- `mgumbel(order, alpha, scale)`
- `mgfgumbel(t, alpha, scale, log = FALSE)`

**Arguments**

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `alpha` location parameter.
- `scale` parameter. Must be strictly positive.
- `log, log.p` logical; if TRUE, probabilities/densities `p` are returned as `log(p)`.
- `lower.tail` logical; if TRUE (default), probabilities are `P[X ≤ x]`, otherwise, `P[X > x]`.
- `order` order of the moment. Only values 1 and 2 are supported.
- `t` numeric vector.
Details

The Gumbel distribution with parameters \( \alpha = \alpha \) and \( \theta = \theta \) has distribution function:

\[
F(x) = \exp[-\exp(-(x - \alpha)/\theta)]
\]

for \(-\infty < x < \infty\), \(-\infty < \alpha < \infty\) and \(\theta > 0\).

The mode of the distribution is in \(\alpha\), the mean is \(\alpha + \gamma \theta\), where \(\gamma = 0.57721566\) is the Euler-Mascheroni constant, and the variance is \(\pi^2 \theta^2 / 6\).

Value

dgumbel gives the density, pgumbel gives the distribution function, qgumbel gives the quantile function, rgumbel generates random deviates, mgumbel gives the \(k\)th raw moment, \(k = 1, 2\), and mgfgamma gives the moment generating function in \(t\).

Invalid arguments will result in return value NaN, with a warning.

Note

Distribution also known as the generalized extreme value distribution Type-I.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


Examples

dgumbel(c(-5, 0, 10, 20), 0.5, 2)

p <- (1:10)/10
pgumbel(qgumbel(p, 2, 3), 2, 3)

curve(pgumbel(x, 0.5, 2), from = -5, to = 20, col = "red")
curve(pgumbel(x, 1.0, 2), add = TRUE, col = "green")
curve(pgumbel(x, 1.5, 3), add = TRUE, col = "blue")
curve(pgumbel(x, 3.0, 4), add = TRUE, col = "cyan")

a <- 3; s <- 4
mgumbel(1, a, s) # mean
a - s * digamma(1) # same

mgumbel(2, a, s) - mgumbel(1, a, s)^2 # variance
(pi * s)^2/6 # same
hachemeister  

### Hachemeister Data Set

**Description**

Hachemeister (1975) data set giving average claim amounts in private passenger bodily injury insurance in five U.S. states over 12 quarters between July 1970 and June 1973 and the corresponding number of claims.

**Usage**

hachemeister

**Format**

A matrix with 5 rows and the following 25 columns:

- state the state number;
- ratio.1,...,ratio.12 the average claim amounts;
- weight.1,...,weight.12 the corresponding number of claims.

**Source**


---

hist.grouped.data  

### Histogram for Grouped Data

**Description**

This method for the generic function `hist` is mainly useful to plot the histogram of grouped data. If `plot = FALSE`, the resulting object of class "histogram" is returned for compatibility with `hist.default`, but does not contain much information not already in `x`.

**Usage**

```r
# S3 method for class 'grouped.data'
hist(x, freq = NULL, probability = !freq,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(x), ylim = NULL, xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE, ...)
```
Arguments

- **x**: an object of class "grouped.data"; only the first column of frequencies is used.
- **freq**: logical; if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE iff group boundaries are equidistant (and probability is not specified).
- **probability**: an alias for !freq, for S compatibility.
- **density**: the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.
- **angle**: the slope of shading lines, given as an angle in degrees (counter-clockwise).
- **col**: a colour to be used to fill the bars. The default of NULL yields unfilled bars.
- **border**: the color of the border around the bars. The default is to use the standard foreground color.
- **main, xlab, ylab**: these arguments to title have useful defaults here.
- **xlim, ylim**: the range of x and y values with sensible defaults. Note that xlim is not used to define the histogram (breaks), but only for plotting (when plot = TRUE).
- **axes**: logical. If TRUE (default), axes are draw if the plot is drawn.
- **plot**: logical. If TRUE (default), a histogram is plotted, otherwise a list of breaks and counts is returned.
- **labels**: logical or character. Additionally draw labels on top of bars, if not FALSE; see plot.histogram.
- **...**: further graphical parameters passed to plot.histogram and their to title and axis (if plot=TRUE).

Value

An object of class "histogram" which is a list with components:

- **breaks**: the r + 1 group boundaries.
- **counts**: r integers; the frequency within each group.
- **density**: the relative frequencies within each group \( n_j/n \), where \( n_j = \text{counts}[j] \).
- **intensities**: same as density. Deprecated, but retained for compatibility.
- **mids**: the r group midpoints.
- **xname**: a character string with the actual x argument name.
- **equidist**: logical, indicating if the distances between breaks are all the same.

Note

The resulting value does not depend on the values of the arguments freq (or probability) or plot. This is intentionally different from S.
References


See Also

hist and hist.default for histograms of individual data and fancy examples.

Examples

data(gdental)
hist(gdental)

InverseBurr

The Inverse Burr Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Inverse Burr distribution with parameters shape1, shape2 and scale.

Usage

dinvburr(x, shape1, shape2, rate = 1, scale = 1/rate, log = FALSE)

pinvburr(q, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)

qinvburr(p, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)

rinvburr(n, shape1, shape2, rate = 1, scale = 1/rate)

minvburr(order, shape1, shape2, rate = 1, scale = 1/rate)

levinvburr(limit, shape1, shape2, rate = 1, scale = 1/rate, order = 1)

Arguments

x, q
vector of quantiles.

p
vector of probabilities.

n
number of observations. If length(n) > 1, the length is taken to be the number required.

shape1, shape2, scale
parameters. Must be strictly positive.

rate
an alternative way to specify the scale.

log, log.p
logical; if TRUE, probabilities/densities p are returned as log(p).

lower.tail
logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].

order
order of the moment.

limit
limit of the loss variable.
Details

The inverse Burr distribution with parameters $\text{shape1} = \tau$, $\text{shape2} = \gamma$ and $\text{scale} = \theta$, has density:

$$f(x) = \frac{\tau \gamma (x/\theta)^\gamma}{x[1 + (x/\theta)^\gamma]^{\tau+1}}$$

for $x > 0$, $\tau > 0$, $\gamma > 0$ and $\theta > 0$.

The inverse Burr is the distribution of the random variable

$$\theta \left( \frac{X}{1 - X} \right)^{1/\gamma},$$

where $X$ has a beta distribution with parameters $\tau$ and 1.

The inverse Burr distribution has the following special cases:

- A Loglogistic distribution when $\text{shape1} == 1$;
- An Inverse Pareto distribution when $\text{shape2} == 1$;
- An Inverse Paralogistic distribution when $\text{shape1} == \text{shape2}$.

The $k$th raw moment of the random variable $X$ is $E[X^k]$, $-\tau \gamma < k < \gamma$.

The $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$, $k > -\tau \gamma$ and $1 - k/\gamma$ not a negative integer.

Value

dinvburr gives the density, invburr gives the distribution function, qinvburr gives the quantile function, rinvburr generates random deviates, minvburr gives the $k$th raw moment, and levinvburr gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levinvburr computes the limited expected value using betaint.

Also known as the Dagum distribution. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


### Examples

```r
dinvburr(2, 2, 3, 1, log = TRUE)
p <- (1:10)/10
pinvburr(qinvburr(p, 2, 3, 1), 2, 3, 1)
## variance
minvburr(2, 2, 3, 1) - minvburr(1, 2, 3, 1) ^ 2
## case with 1 - order/shape2 > 0
levinvburr(10, 2, 3, 1, order = 2)
## case with 1 - order/shape2 < 0
levinvburr(10, 2, 1.5, 1, order = 2)
```

---

### InverseExponential

**The Inverse Exponential Distribution**

**Description**

Density function, distribution function, quantile function, random generation raw moments and limited moments for the Inverse Exponential distribution with parameter `scale`.

**Usage**

```r
dinvexp(x, rate = 1, scale = 1/rate, log = FALSE)
pinvexp(q, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qinvexp(p, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rinvexp(n, rate = 1, scale = 1/rate)
minvexp(order, rate = 1, scale = 1/rate)
levinvexp(limit, rate = 1, scale = 1/rate, order)
```

**Arguments**

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `scale` parameter. Must be strictly positive.
- `rate` an alternative way to specify the scale.
- `log, log.p` logical; if TRUE, probabilities/densities `p` are returned as `log(p)`.
- `lower.tail` logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
- `order` order of the moment.
- `limit` limit of the loss variable.
Details

The inverse exponential distribution with parameter $\text{scale} = \theta$ has density:

$$f(x) = \frac{\theta e^{-\theta/x}}{x^2}$$

for $x > 0$ and $\theta > 0$.

The $k$th raw moment of the random variable $X$ is $E[X^k]$, $k < 1$, and the $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$, all $k$.

Value

dinvexp gives the density, pinvexp gives the distribution function, qinvexp gives the quantile function, rinvexp generates random deviates, minvexp gives the $k$th raw moment, and leinvexp calculates the $k$th limited moment.

Invalid arguments will result in return value NaN, with a warning.

Note

leinvexp computes the limited expected value using gammainc from package expint.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


Examples

```r
exp(dinvexp(2, 2, log = TRUE))
p <- (1:10)/10
pinvexp(qinvexp(p, 2), 2)
minvexp(0.5, 2)
```
Usage

\begin{verbatim}
dinvgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pinvgamma(q, shape, rate = 1, scale = 1/rate,
  lower.tail = TRUE, log.p = FALSE)
qinvgamma(p, shape, rate = 1, scale = 1/rate,
  lower.tail = TRUE, log.p = FALSE)
rinvgamma(n, shape, rate = 1, scale = 1/rate)
minvgamma(order, shape, rate = 1, scale = 1/rate)
levinvgamma(limit, shape, rate = 1, scale = 1/rate,
  order = 1)
mgfinvgamma(t, shape, rate =1, scale = 1/rate, log =FALSE)
\end{verbatim}

Arguments

- \(x, q\) vector of quantiles.
- \(p\) vector of probabilities.
- \(n\) number of observations. If \(\text{length}(n) > 1\), the length is taken to be the number required.
- \(\text{shape, scale}\) parameters. Must be strictly positive.
- \(\text{rate}\) an alternative way to specify the scale.
- \(\text{log, log.p}\) logical; if \(\text{TRUE}\), probabilities/densities \(p\) are returned as \(\log(p)\).
- \(\text{lower.tail}\) logical; if \(\text{TRUE}\) (default), probabilities are \(P[X \leq x]\), otherwise, \(P[X > x]\).
- \(\text{order}\) order of the moment.
- \(\text{limit}\) limit of the loss variable.
- \(t\) numeric vector.

Details

The inverse gamma distribution with parameters \(\text{shape} = \alpha\) and \(\text{scale} = \theta\) has density:

\[
f(x) = \frac{u^{\alpha}e^{-u}}{x\Gamma(\alpha)}, \quad u = \theta/x
\]

for \(x > 0, \alpha > 0\) and \(\theta > 0\). (Here \(\Gamma(\alpha)\) is the function implemented by R’s \texttt{gamma()} and defined in its help.)

The special case \(\text{shape} = 1\) is an Inverse Exponential distribution.

The \(k\)th raw moment of the random variable \(X\) is 
\(E[X^k], k < \alpha,\) and the \(k\)th limited moment at some limit \(d\) is 
\(E[\min(X, d)^k],\) all \(k\).

The moment generating function is given by \(E[e^{tX}].\)

Value

dinvgamma gives the density, pinvgamma gives the distribution function, qinvgamma gives the quantile function, rinvgamma generates random deviates, minvgamma gives the \(k\)th raw moment, levinvgamma gives the \(k\)th moment of the limited loss variable, and mgfinvgamma gives the moment generating function in \(t\).

Invalid arguments will result in return value NaN, with a warning.
Note
levinvgamma computes the limited expected value using gammainc from package expint.
Also known as the Vinci distribution. See also Kleiber and Kotz (2003) for alternative names and
parametrizations.
The "distributions" package vignette provides the interrelations between the continuous size
distributions in actuar and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References
Kleiber, C. and Kotz, S. (2003), Statistical Size Distributions in Economics and Actuarial Sciences,
Wiley.
Klugman, S. A., Panjer, H. H. and Willmot, G. E. (2012), Loss Models, From Data to Decisions,

Examples
exp(dinvgamma(2, 3, 4, log = TRUE))
p <- (1:10)/10
pinvgamma(qinvgamma(p, 2, 3), 2, 3)
minvgamma(-1, 2, 2) ^ 2
levinvgamma(10, 2, 2, order = 1)
mgfinvgamma(-1, 3, 2)

Usage
dinvgauss(x, mean, shape = 1, dispersion = 1/shape,
    log = FALSE)
pinvgauss(q, mean, shape = 1, dispersion = 1/shape,
    lower.tail = TRUE, log.p = FALSE)
qinvgauss(p, mean, shape = 1, dispersion = 1/shape,
    lower.tail = TRUE, log.p = FALSE,
    tol = 1e-14, maxit = 100, echo = FALSE, trace = echo)
rinvgauss(n, mean, shape = 1, dispersion = 1/shape)
minvgauss(order, mean, shape = 1, dispersion = 1/shape)
levinvgauss(limit, mean, shape = 1, dispersion = 1/shape, order = 1)
mgfinvgauss(t, mean, shape = 1, dispersion = 1/shape, log = FALSE)

Arguments

x, q scalar or vector of quantiles.
p scalar or vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
mean, shape scalar or vector of parameters. Must be strictly positive. Infinite values are supported.
dispersion alternative way to specify the shape.
log, log.p logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].
order order of the moment. Only order = 1 is supported by levinvgauss.
limit limit of the loss variable.
tol small positive value. Tolerance to assess convergence in the Newton computation of quantiles.
maxit positive integer; maximum number of recursions in the Newton computation of quantiles.
echo, trace logical; echo the recursions to screen in the Newton computation of quantiles.
t numeric vector.

details

The inverse Gaussian distribution with parameters mean = \mu and dispersion = \phi has density:

\[ f(x) = \left( \frac{1}{2\pi \phi x^3} \right)^{1/2} \exp \left( -\frac{(x - \mu)^2}{2\mu^2 \phi x} \right), \]

for \( x \geq 0, \mu > 0 \) and \( \phi > 0 \).

The limiting case \( \mu = \infty \) is an inverse chi-squared distribution (or inverse gamma with shape = 1/2 and rate = 2\phi). This distribution has no finite strictly positive, integer moments.

The limiting case \( \phi = 0 \) is an infinite spike in \( x = 0 \).

If the random variable \( X \) is IG(\( \mu, \phi \)), then \( X/\mu \) is IG(1, \( \phi \mu \)).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], k = 1, 2, \ldots, \) the limited expected value at some limit \( d \) is \( E[\min(X, d)] \) and the moment generating function is \( E[e^{tX}] \).

The moment generating function of the inverse Gaussian is defined for \( t \leq 1/(2 * mean^2 * phi) \).

value

dinvgauss gives the density, pinvgauss gives the distribution function, qinvgauss gives the quantile function, rinvgauss generates random deviates, minvgauss gives the \( k \)th raw moment, levinvgauss gives the limited expected value, and mgfinvgauss gives the moment generating function in \( t \).

Invalid arguments will result in return value NaN, with a warning.
Note

Functions `dinvgauss`, `pinvgauss` and `qinvgauss` are C implementations of functions of the same name in package `statmod`; see Giner and Smyth (2016).

Devroye (1986, chapter 4) provides a nice presentation of the algorithm to generate random variates from an inverse Gaussian distribution.

The "distributions" package vignette provides the interrelations between the continuous size distributions in `actuar` and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

dinvgamma for the inverse gamma distribution.

Examples

dinvgauss(c(-1, 0, 1, 2, Inf), mean = 1.5, dis = 0.7)
dinvgauss(c(-1, 0, 1, 2, Inf), mean = Inf, dis = 0.7)
dinvgauss(c(-1, 0, 1, 2, Inf), mean = 1.5, dis = Inf) # spike at zero

  ## Typical graphical representations of the inverse Gaussian
  ## distribution. First fixed mean and varying shape; second
  ## varying mean and fixed shape.
  col = c("red", "blue", "green", "cyan", "yellow", "black")
  par = c(0.125, 0.5, 1, 2, 8, 32)
  curve(dinvgauss(x, 1, par[1]), from = 0, to = 2, col = col[1])
  for (i in 2:6)
    curve(dinvgauss(x, 1, par[i]), from = 0, to = 2, col = col[i], add = TRUE)
  curve(dinvgauss(x, par[1], 1), from = 0, to = 2, col = col[1])
  for (i in 2:6)
    curve(dinvgauss(x, par[i], 1), from = 0, to = 2, col = col[i], add = TRUE)
  pinnvgauss(qinvgauss((1:10)/10, 1.5, shape = 2), 1.5, 2)
  minvgauss(1:4, 1.5, 2)
  levinvgauss(c(0, 0.5, 1, 1.2, 10, Inf), 1.5, 2)
## The Inverse Paralogistic Distribution

### Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Inverse Paralogistic distribution with parameters shape and scale.

### Usage


dinvparalogis(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pinvparalogis(q, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qinvparalogis(p, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rinvparalogis(n, shape, rate = 1, scale = 1/rate)
minvparalogis(order, shape, rate = 1, scale = 1/rate)
levinvparalogis(limit, shape, rate = 1, scale = 1/rate, order = 1)

### Arguments

- x, q: vector of quantiles.
- p: vector of probabilities.
- n: number of observations. If length(n) > 1, the length is taken to be the number required.
- shape, scale: parameters. Must be strictly positive.
- rate: an alternative way to specify the scale.
- log, log.p: logical; if TRUE, probabilities/densities p are returned as log(p).
- lower.tail: logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- order: order of the moment.
- limit: limit of the loss variable.

### Details

The inverse paralogistic distribution with parameters shape = \( \tau \) and scale = \( \theta \) has density:

\[
f(x) = \frac{\tau^2(x/\theta)^\tau^2}{x[1 + (x/\theta)^\tau]^{\tau+1}}
\]

for \( x > 0, \tau > 0 \) and \( \theta > 0 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \), \(-\tau^2 < k < \tau\).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k] \), \( k > -\tau^2 \) and \( 1 - k/\tau \) not a negative integer.
InversePareto

Value

dinvparalogis gives the density, pinvparalogis gives the distribution function, qinvparalogis gives the quantile function, rinvparalogis generates random deviates, minvparalogis gives the \(k\)th raw moment, and leinvparalogis gives the \(k\)th moment of the limited loss variable.

Invalid arguments will result in return value \(\text{NaN}\), with a warning.

Note

leinvparalogis computes the limited expected value using \(\text{betaint}\).

See Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in \texttt{actuar} and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


Examples

\begin{verbatim}
exp(dinvparalogis(2, 3, 4, log = TRUE))
p <- (1:10)/10
pinvparalogis(qinvparalogis(p, 2, 3), 2, 3)

## first negative moment
minvparalogis(-1, 2, 2)

## case with 1 - order/shape > 0
leinvparalogis(10, 2, 2, order = 1)

## case with 1 - order/shape < 0
leinvparalogis(10, 2/3, 2, order = 1)
\end{verbatim}

---

\textbf{Description}

Density function, distribution function, quantile function, random generation raw moments and limited moments for the Inverse Pareto distribution with parameters shape and scale.
InversePareto

Usage

dinvpareto(x, shape, scale, log = FALSE)
pinvpareto(q, shape, scale, lower.tail = TRUE, log.p = FALSE)
qinvpareto(p, shape, scale, lower.tail = TRUE, log.p = FALSE)
rinvpareto(n, shape, scale)
minvpareto(order, shape, scale)
levinvpareto(limit, shape, scale, order = 1)

Arguments

x, q  vector of quantiles.
p  vector of probabilities.
n  number of observations. If length(n) > 1, the length is taken to be the number required.
shape, scale  parameters. Must be strictly positive.
log, log.p  logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail  logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].
order  order of the moment.
limit  limit of the loss variable.

Details

The inverse Pareto distribution with parameters shape = τ and scale = θ has density:

\[ f(x) = \frac{\tau \theta x^{\tau-1}}{(x + \theta)^{\tau+1}} \]

for \( x > 0, \tau > 0 \) and \( \theta > 0 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], -\tau < k < 1 \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k], k > -\tau \).

Value

dinvpareto gives the density, pinvpareto gives the distribution function, qinvpareto gives the quantile function, rinvpareto generates random deviates, minvpareto gives the \( k \)th raw moment, and levinvpareto calculates the \( k \)th limited moment.

Invalid arguments will result in return value NaN, with a warning.

Note

Evaluation of levinvpareto is done using numerical integration.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon
References


Examples

```r
exp(dinvpareto(2, 3, 4, log = TRUE))
p <- (1:10)/10
pinvpareto(qinvpareto(p, 2, 3), 2, 3)
minvpareto(0.5, 1, 2)
```

---

The Inverse Transformed Gamma Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments, and limited moments for the Inverse Transformed Gamma distribution with parameters shape1, shape2 and scale.

Usage

```r
dinvtrgamma(x, shape1, shape2, rate = 1, scale = 1/rate, log = FALSE)
pinvtrgamma(q, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qinvtrgamma(p, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rinvtrgamma(n, shape1, shape2, rate = 1, scale = 1/rate)
minvtrgamma(order, shape1, shape2, rate = 1, scale = 1/rate)
leinvtrgamma(limit, shape1, shape2, rate = 1, scale = 1/rate, order = 1)
```

Arguments

- `x, q`: vector of quantiles.
- `p`: vector of probabilities.
- `n`: number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `shape1, shape2, scale`: parameters. Must be strictly positive.
- `rate`: an alternative way to specify the scale.
- `log, log.p`: logical; if TRUE, probabilities/densities `p` are returned as `log(p)`.
- `lower.tail`: logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- `order`: order of the moment.
- `limit`: limit of the loss variable.
Details

The inverse transformed gamma distribution with parameters \( \text{shape1} = \alpha \), \( \text{shape2} = \tau \) and \( \text{scale} = \theta \), has density:

\[
f(x) = \frac{\tau u^\alpha e^{-u}}{x \Gamma(\alpha)}, \quad u = \left(\frac{\theta}{x}\right)^\tau
\]

for \( x > 0, \alpha > 0, \tau > 0 \) and \( \theta > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by \( R \)'s \text{gamma()} \) and defined in its help.)

The inverse transformed gamma is the distribution of the random variable \( \theta X^{-1/\tau} \), where \( X \) has a gamma distribution with shape parameter \( \alpha \) and scale parameter 1 or, equivalently, of the random variable \( Y^{-1/\tau} \) with \( Y \) a gamma distribution with shape parameter \( \alpha \) and scale parameter \( \theta^{-\tau} \).

The inverse transformed gamma distribution defines a family of distributions with the following special cases:

- An Inverse Gamma distribution when \( \text{shape2} == 1 \);
- An Inverse Weibull distribution when \( \text{shape1} == 1 \);
- An Inverse Exponential distribution when \( \text{shape1} == \text{shape2} == 1 \);

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \), \( k < \alpha \tau \), and the \( k \)th limited moment at some limit \( d \) is \( E[\min(X,d)^k] \) for all \( k \).

Value

dinvtrgamma gives the density, pinvtrgamma gives the distribution function, qinvtrgamma gives the quantile function, rinvtrgamma generates random deviates, minvtrgamma gives the \( k \)th raw moment, and levinvtrgamma gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levinvtrgamma computes the limited expected value using gammainc from package \text{expint}.

Distribution also known as the Inverse Generalized Gamma. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in \text{actuar} and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


InverseWeibull

Examples

\[
\begin{align*}
\exp(\text{dinvtrgamma}(2, 3, 4, 5, \text{log} = \text{TRUE})) \\
p &<-(1:10)/10 \\
\text{pinvtrgamma}(\text{qinvtrgamma}(p, 2, 3, 4), 2, 3, 4) \\
\text{minvtrgamma}(2, 3, 4, 5) \\
\text{levinvtrgamma}(200, 3, 4, 5, \text{order} = 2)
\end{align*}
\]

The Inverse Weibull Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Inverse Weibull distribution with parameters shape and scale.

Usage

\[
\begin{align*}
\text{dinvweibull}(x, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate, log} = \text{FALSE}) \\
\text{pinvweibull}(q, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \\
\quad \text{lower.tail} = \text{TRUE, log.p} = \text{FALSE}) \\
\text{qinvweibull}(p, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \\
\quad \text{lower.tail} = \text{TRUE, log.p} = \text{FALSE}) \\
\text{rinvweibull}(n, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}) \\
\text{minvweibull}(\text{order}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}) \\
\text{levinvweibull}(\text{limit}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \\
\quad \text{order} = 1)
\end{align*}
\]

Arguments

\[
\begin{align*}
x, q & \quad \text{vector of quantiles.} \\
p & \quad \text{vector of probabilities.} \\
n & \quad \text{number of observations. If length}(n) > 1, \text{the length is taken to be the number required.} \\
\text{shape, scale} & \quad \text{parameters. Must be strictly positive.} \\
\text{rate} & \quad \text{an alternative way to specify the scale.} \\
\text{log, log.p} & \quad \text{logical; if TRUE, probabilities/densities } p \text{ are returned as log}(p). \\
\text{lower.tail} & \quad \text{logical; if TRUE (default), probabilities are } P[X \leq x], \text{otherwise, } P[X > x]. \\
\text{order} & \quad \text{order of the moment.} \\
\text{limit} & \quad \text{limit of the loss variable.}
\end{align*}
\]
Details

The inverse Weibull distribution with parameters shape = τ and scale = θ has density:

\[ f(x) = \frac{\tau (\theta/x)^\tau e^{-(\theta/x)^\tau}}{x} \]

for \( x > 0, \tau > 0 \) and \( \theta > 0 \).

The special case \( \tau = 1 \) is an Inverse Exponential distribution.

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], k < \tau \), and the \( k \)th limited moment at some limit \( d \) is \( E[\min(X,d)^k] \), all \( k \).

Value

dinvweibull gives the density, pinvweibull gives the distribution function, qinvweibull gives the quantile function, rinvweibull generates random deviates, minvweibull gives the \( k \)th raw moment, and levinvweibull gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levinvweibull computes the limited expected value using gammainc from package expint.

Distribution also know as the log-Gompertz. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


Examples

```r
exp(dinvweibull(2, 3, 4, log = TRUE))
p <- (1:10)/10
pinvweibull(qinvweibull(p, 2, 3), 2, 3)
mlgompertz(-1, 3, 3)
levinvweibull(10, 2, 3, order = 1)
```
The Logarithmic Distribution

Description

Density function, distribution function, quantile function and random generation for the Logarithmic (or log-series) distribution with parameter prob.

Usage

dlogarithmic(x, prob, log = FALSE)
plogarithmic(q, prob, lower.tail = TRUE, log.p = FALSE)
qlogarithmic(p, prob, lower.tail = TRUE, log.p = FALSE)
rlogarithmic(n, prob)

Arguments

x vector of (strictly positive integer) quantiles.
q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
prob parameter. 0 <= prob < 1.
log, log.p logical; if TRUE, probabilities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].

Details

The logarithmic (or log-series) distribution with parameter prob = θ has probability mass function

\[ p(x) = \frac{a\theta^x}{x} \]

with \( a = -1 / \log(1 - \theta) \) and for \( x = 1, 2, \ldots, 0 < \theta < 1 \).

The logarithmic distribution is the limiting case of the zero-truncated negative binomial distribution with size parameter equal to 0. Note that in this context, parameter prob generally corresponds to the probability of failure of the zero-truncated negative binomial.

If an element of x is not integer, the result of dlogarithmic is zero, with a warning.

The quantile is defined as the smallest value x such that \( F(x) \geq p \), where F is the distribution function.
Value

dlogarithmic gives the probability mass function, plogarithmic gives the distribution function, qlogarithmic gives the quantile function, and rlogarithmic generates random deviates.

Invalid prob will result in return value NaN, with a warning.

The length of the result is determined by n for rlogarithmic, and is the maximum of the lengths of the numerical arguments for the other functions.

Note

qlogarithmic is based on qbinom et al.; it uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rlogarithmic is an implementation of the LS and LK algorithms of Kemp (1981) with automatic selection. As suggested by Devroye (1986), the LS algorithm is used when prob < 0.95, and the LK algorithm otherwise.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

dztnbinom for the zero-truncated negative binomial distribution.

Examples

```r
## Table 1 of Kemp (1981) [also found in Johnson et al. (2005), chapter 7]
p <- c(0.1, 0.3, 0.5, 0.7, 0.8, 0.85, 0.9, 0.95, 0.99, 0.995, 0.999, 0.9999)
round(rbind(dlogarithmic(1, p),
            dlogarithmic(2, p),
            plogarithmic(9, p, lower.tail = FALSE),
            -p/((1 - p) * log(1 - p))), 2)
qlogarithmic(plogarithmic(1:10, 0.9), 0.9)
x <- rlogarithmic(1000, 0.8)
y <- sort(unique(x))
plot(y, table(x)/length(x), type = "h", lwd = 2,
```


The Loggamma Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Loggamma distribution with parameters shapelog and ratelog.

Usage

dlgamma(x, shapelog, ratelog, log = FALSE)
plgamma(q, shapelog, ratelog, lower.tail = TRUE, log.p = FALSE)
qlgamma(p, shapelog, ratelog, lower.tail = TRUE, log.p = FALSE)
rlgamma(n, shapelog, ratelog)
mlgamma(order, shapelog, ratelog)
levlgamma(limit, shapelog, ratelog, order = 1)

Arguments

x, q
vector of quantiles.
p
vector of probabilities.
n
number of observations. If length(n) > 1, the length is taken to be the number required.
shapelog, ratelog
parameters. Must be strictly positive.
log, log.p
logical; if TRUE, probabilities/densities p are returned as \log(p).
lower.tail
logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
order
order of the moment.
limit
limit of the loss variable.

Details

The loggamma distribution with parameters \( \text{shapelog} = \alpha \) and \( \text{ratelog} = \lambda \) has density:

\[
f(x) = \frac{\lambda^\alpha (\log x)^{\alpha-1}}{\Gamma(\alpha) x^{\lambda+1}}
\]

for \( x > 1, \alpha > 0 \) and \( \lambda > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by R’s \texttt{gamma()} and defined in its help.)
The loggamma is the distribution of the random variable $e^X$, where $X$ has a gamma distribution with shape parameter $\alpha$ and scale parameter $1/\lambda$.

The $k$th raw moment of the random variable $X$ is $E[X^k]$ and the $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$, $k < \lambda$.

**Value**

dlgamma gives the density, plgamma gives the distribution function, qlgamma gives the quantile function, rlgamma generates random deviates, mlgamma gives the $k$th raw moment, and levlgamma gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

**Note**

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

**References**


**Examples**

```r
exp(dlgamma(2, 3, 4, log = TRUE))
p <- (1:10)/10
plgamma(qlgamma(p, 2, 3), 2, 3)
mlgamma(2, 3, 4) - mlgamma(1, 3, 4)^2
levlgamma(10, 3, 4, order = 2)
```

---

**Loglogistic**

The Loglogistic Distribution

**Description**

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Loglogistic distribution with parameters shape and scale.
Usage

dllogis(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pllogis(q, shape, rate = 1, scale = 1/rate,
    lower.tail = TRUE, log.p = FALSE)
qlllogis(p, shape, rate = 1, scale = 1/rate,
    lower.tail = TRUE, log.p = FALSE)
rllogis(n, shape, rate = 1, scale = 1/rate)
mllogis(order, shape, rate = 1, scale = 1/rate)
levllogis(limit, shape, rate = 1, scale = 1/rate,
    order = 1)

Arguments

x, q  vector of quantiles.
p  vector of probabilities.
n  number of observations. If length(n) > 1, the length is taken to be the number required.
shape, scale  parameters. Must be strictly positive.
rate  an alternative way to specify the scale.
log, log.p  logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail  logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
order  order of the moment.
limit  limit of the loss variable.

Details

The loglogistic distribution with parameters shape = \( \gamma \) and scale = \( \theta \) has density:

\[
f(x) = \frac{\gamma(x/\theta)^\gamma}{x[1 + (x/\theta)\gamma]^2}
\]

for \( x > 0, \gamma > 0 \) and \( \theta > 0 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], -\gamma < k < \gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k], k > -\gamma \) and \( 1 - k/\gamma \) not a negative integer.

Value

dllogis gives the density, pllogis gives the distribution function, qlllogis gives the quantile function, rllogis generates random deviates, mllogis gives the \( k \)th raw moment, and levllogis gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.
LognormalMoments

Note
levllogis computes the limited expected value using betaint.
Also known as the Fisk distribution. See also Kleiber and Kotz (2003) for alternative names and parametrizations.
The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References

See Also
dpareto3 for an equivalent distribution with a location parameter.

Examples
exp(dllogis(2, 3, 4, log = TRUE))
p <- (1:10)/10
pllogis qllogis(p, 2, 3), 2, 3)
## mean
mllogis(1, 2, 3)
## case with 1 - order/shape > 0
levllogis(10, 2, 3, order = 1)
## case with 1 - order/shape < 0
levllogis(10, 2/3, 3, order = 1)

LognormalMoments Raw and Limited Moments of the Lognormal Distribution

Description
Raw moments and limited moments for the Lognormal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.
mde

Usage

mlnorm(order, meanlog = 0, sdlog = 1)
levlnorm(limit, meanlog = 0, sdlog = 1, order = 1)

Arguments

order order of the moment.
limit limit of the loss variable.
meanlog, sdlog mean and standard deviation of the distribution on the log scale with default values of 0 and 1 respectively.

Value

mlnorm gives the $k$th raw moment and levlnorm gives the $k$th moment of the limited loss variable. Invalid arguments will result in return value NaN, with a warning.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

Lognormal for details on the lognormal distribution and functions [dpqr]lnorm.

Examples

mlnorm(2, 3, 4) - mlnorm(1, 3, 4)^2
levlnorm(10, 3, 4, order = 2)

mde

Minimum Distance Estimation

Description

Minimum distance fitting of univariate distributions, allowing parameters to be held fixed if desired.

Usage

mde(x, fun, start, measure = c("CvM", "chi-square", "LAS"),
    weights = NULL, ...)

Arguments

- **x**: a vector or an object of class "grouped data" (in which case only the first column of frequencies is used).
- **fun**: function returning a cumulative distribution (for measure = "CvM" and measure = "chi-square") or a limited expected value (for measure = "LAS") evaluated at its first argument.
- **start**: a named list giving the parameters to be optimized with initial values
- **measure**: either "CvM" for the Cramer-von Mises method, "chi-square" for the modified chi-square method, or "LAS" for the layer average severity method.
- **weights**: weights; see Details.
- **...**: Additional parameters, either for fun or for optim. In particular, it can be used to specify bounds via lower or upper or both. If arguments of fun are included they will be held fixed.

Details

The Cramer-von Mises method ("CvM") minimizes the squared difference between the theoretical cdf and the empirical cdf at the data points (for individual data) or the ogive at the knots (for grouped data).

The modified chi-square method ("chi-square") minimizes the modified chi-square statistic for grouped data, that is the squared difference between the expected and observed frequency within each group.

The layer average severity method ("LAS") minimizes the squared difference between the theoretical and empirical limited expected value within each group for grouped data.

All sum of squares can be weighted. If arguments weights is missing, weights default to 1 for measure = "CvM" and measure = "LAS"; for measure = "chi-square", weights default to \(1/n_j\), where \(n_j\) is the frequency in group \(j = 1, \ldots, r\).

Optimization is performed using optim. For one-dimensional problems the Nelder-Mead method is used and for multi-dimensional problems the BFGS method, unless arguments named lower or upper are supplied when L-BFGS-B is used or method is supplied explicitly.

Value

An object of class "mde", a list with two components:

- **estimate**: the parameter estimates, and
- **distance**: the distance.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References

Examples

## Individual data example
```
data(dental)
mde(dental, pexp, start = list(rate = 1/200), measure = "CvM")
```

## Example 2.21 of Klugman et al. (1998)
```
data(gdental)
mde(gdental, pexp, start = list(rate = 1/200), measure = "CvM")
mde(gdental, pexp, start = list(rate = 1/200), measure = "chi-square")
mde(gdental, levexp, start = list(rate = 1/200), measure = "LAS")
```

## Two-parameter distribution example
```
try(mde(gdental, ppareto, start = list(shape = 3, scale = 600),
measurement = "CvM")) # no convergence
```

## Working in log scale often solves the problem
```
pparetolog <- function(x, shape, scale)
  ppareto(x, exp(shape), exp(scale))
```
```
(p <- mde(gdental, pparetolog, start = list(shape = log(3),
  scale = log(600)), measure = "CvM")
exp(p$estimate)
```

mean.grouped.data

**Arithmetic Mean**

Description

Mean of grouped data objects.

Usage

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

Arguments

- `x`: an object of class "grouped.data".
- `...`: further arguments passed to or from other methods.

Details

The mean of grouped data with group boundaries \( c_0, c_1, \ldots, c_r \) and group frequencies \( n_1, \ldots, n_r \) is
\[
\frac{1}{n} \sum_{j=1}^{r} a_j n_j,
\]
where \( a_j = (c_{j-1} + c_j)/2 \) is the midpoint of the \( j \)th interval, and \( n = \sum_{j=1}^{r} n_j \).
Value
A named vector of means.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>

References

See Also
grouped.data to create grouped data objects; emm to compute higher moments.

Examples
```r
data(gdental)
mean(gdental)
```

<table>
<thead>
<tr>
<th>NormalSupp</th>
<th>Moments and Moment generating function of the Normal Distribution</th>
</tr>
</thead>
</table>

Description
Raw moments and moment generating function for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage
```r
mnorm(order, mean = 0, sd = 1)
mgfnorm(t, mean = 0, sd = 1, log = FALSE)
```

Arguments
- `order`: vector of integers; order of the moment.
- `mean`: vector of means.
- `sd`: vector of standard deviations.
- `t`: numeric vector.
- `log`: logical; if TRUE, the cumulant generating function is returned.

Details
The $k$th raw moment of the random variable $X$ is $E[X^k]$ and the moment generating function is $E[e^{tX}]$.

Only integer moments are supported.
ogive

Value

mnorm gives the $k$th raw moment and mgfnorm gives the moment generating function in $t$.
Invalid arguments will result in return value NaN, with a warning.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Christophe Dutang

References


See Also

Normal

Examples

mgfnorm(0:4,1,2)
mnorm(3)

ogive

*Ogive for Grouped Data*

Description

Compute a smoothed empirical distribution function for grouped data.

Usage

ogive(x, ...)

## Default S3 method:
ogive(x, y = NULL, breaks = "Sturges", nclass = NULL, ...)

## S3 method for class 'grouped.data'
ogive(x, ...)

## S3 method for class 'ogive'
print(x, digits = getOption("digits") - 2, ...)

## S3 method for class 'ogive'
summary(object, ...)

## S3 method for class 'ogive'
knots(Fn, ...)

## S3 method for class 'ogive'
plot(x, main = NULL, xlab = "x", ylab = "F(x)", ...)
Arguments

- **x**: for the generic and all but the default method, an object of class "grouped.data"; for the default method, a vector of individual data if \( y \) is NULL, a vector of group boundaries otherwise.
- **y**: a vector of group frequencies.
- **breaks, nclass**: arguments passed to `grouped.data`; used only for individual data (when \( y \) is NULL).
- **digits**: number of significant digits to use, see `print`.
- **Fn, object**: an R object inheriting from "ogive".
- **main**: main title.
- **xlab, ylab**: labels of x and y axis.
- **...**: arguments to be passed to subsequent methods.

Details

The ogive is a linear interpolation of the empirical cumulative distribution function. The equation of the ogive is

\[
G_n(x) = \frac{(c_j - x)F_n(c_{j-1}) + (x - c_{j-1})F_n(c_j)}{c_j - c_{j-1}}
\]

for \( c_{j-1} < x \leq c_j \) and where \( c_0, \ldots, c_r \) are the \( r + 1 \) group boundaries and \( F_n \) is the empirical distribution function of the sample.

Value

For `ogive`, a function of class "ogive", inheriting from the "function" class.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

`grouped.data` to create grouped data objects; `quantile.grouped.data` for the inverse function; `approxfun`, which is used to compute the ogive; `stepfun` for related documentation (even though the ogive is not a step function).
Examples

```r
## Most common usage: create ogive from grouped data object.
Fn <- ogive(gdental)
Fn
summary(Fn)  # the group boundaries
knots(Fn)  # the group boundaries
Fn(knots(Fn))  # true values of the empirical cdf
Fn(c(80, 200, 2000))  # linear interpolations
plot(Fn)  # graphical representation

## Alternative 1: create ogive directly from individual data
## without first creating a grouped data object.
give(dental)  # automatic class boundaries
give(dental, breaks = c(0, 50, 200, 500, 1500, 2000))

## Alternative 2: create ogive from set of group boundaries and
## group frequencies.
cj <- c(0, 25, 50, 100, 250, 500, 1000)
nj <- c(30, 31, 57, 42, 45, 10)
give(cj, nj)
```

---

**Paralogistic**

*The Paralogistic Distribution*

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Paralogistic distribution with parameters `shape` and `scale`.

Usage

```r
dparalogis(x, shape, rate = 1, scale = 1/rate, log = FALSE)
pparalogis(q, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qparalogis(p, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rparalogis(n, shape, rate = 1, scale = 1/rate)
mparalogis(order, shape, rate = 1, scale = 1/rate)
levparalogis(limit, shape, rate = 1, scale = 1/rate, order = 1)
```

Arguments

- `x, q` vector of quantiles.
- `p` vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
shape, scale parameters. Must be strictly positive.
rate an alternative way to specify the scale.
log, log.p logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
order order of the moment.
limit limit of the loss variable.

Details
The paralogistic distribution with parameters shape = \( \alpha \) and scale = \( \theta \) has density:

\[
f(x) = \frac{\alpha^2(x/\theta)^\alpha}{x[1 + (x/\theta)^\alpha]^{\alpha+1}}
\]

for \( x > 0, \alpha > 0 \) and \( \theta > 0 \).
The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], -\alpha < k < \alpha^2 \).
The \( k \)th limited moment at some limit \( d \) is \( E[\min(X,d)^k], k > -\alpha \) and \( \alpha - k/\alpha \) not a negative integer.

Value
dparalogis gives the density, pparalogis gives the distribution function, qparalogis gives the quantile function, rparalogis generates random deviates, mparalogis gives the \( k \)th raw moment, and levparalogis gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note
levparalogis computes the limited expected value using betaint.
See Kleiber and Kotz (2003) for alternative names and parametrizations.
The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References
Pareto

Examples

\[ \exp(dparalogis(2, 3, 4, \text{log} = \text{TRUE})) \]
\[ p \leftarrow (1:10)/10 \]
\[ pparalogis(qparalogis(p, 2, 3), 2, 3) \]

## variance
\[ mparalogis(2, 2, 3) - mparalogis(1, 2, 3)^2 \]

## case with shape - order/shape \textgreater{} 0
\[ levparalogis(10, 2, 3, \text{order} = 2) \]

## case with shape - order/shape < 0
\[ levparalogis(10, 1.25, 3, \text{order} = 2) \]

---

The Pareto Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Pareto distribution with parameters shape and scale.

Usage

\[
\text{dpareto}(x, \text{shape}, \text{scale}, \text{log} = \text{FALSE}) \\
\text{ppareto}(q, \text{shape}, \text{scale}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{qpareto}(p, \text{shape}, \text{scale}, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
\text{rpareto}(n, \text{shape}, \text{scale}) \\
\text{mpareto}(\text{order}, \text{shape}, \text{scale}) \\
\text{levpareto}(\text{limit}, \text{shape}, \text{scale}, \text{order} = 1)
\]

Arguments

- \( x, q \) vector of quantiles.
- \( p \) vector of probabilities.
- \( n \) number of observations. If \( \text{length}(n) > 1 \), the length is taken to be the number required.
- \( \text{shape}, \text{scale} \) parameters. Must be strictly positive.
- \( \text{log}, \text{log.p} \) logical; if \text{TRUE}, probabilities/densities \( p \) are returned as \( \log(p) \).
- \( \text{lower.tail} \) logical; if \text{TRUE} (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- \( \text{order} \) order of the moment.
- \( \text{limit} \) limit of the loss variable.
The Pareto distribution with parameters $\alpha$ and $\theta$ has density:

$$f(x) = \frac{\alpha \theta^\alpha}{(x + \theta)^{\alpha+1}}$$

for $x > 0$, $\alpha > 0$ and $\theta$.

There are many different definitions of the Pareto distribution in the literature; see Arnold (2015) or Kleiber and Kotz (2003). In the nomenclature of actuar, the “Pareto distribution” does not have a location parameter. The version with a location parameter is the Pareto II.

The $k$th raw moment of the random variable $X$ is $E[X^k]$, $-1 < k < \alpha$.

The $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$, $k > -1$ and $\alpha - k$ not a negative integer.

### Value

dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, rpareto generates random deviates, mpareto gives the $k$th raw moment, and levpareto gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

### Note

levpareto computes the limited expected value using $\text{betaint}$.

The version of the Pareto defined for $x > \theta$ is named Single Parameter Pareto, or Pareto I, in actuar.

### Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

### References


### See Also

dpareto2 for an equivalent distribution with location parameter.

dpareto1 for the Single Parameter Pareto distribution.

"distributions" package vignette for details on the interrelations between the continuous size distributions in actuar and complete formulas underlying the above functions.
Examples

\[
\text{exp(dpareto(2, 3, 4, log = TRUE))}
\]
\[
p <- (1:10)/10
\]
\[
\text{ppareto(qpareto(p, 2, 3), 2, 3)}
\]

## variance
\[
\text{mpareto(2, 4, 1) - mpareto(1, 4, 1)^2}
\]

## case with shape - order > 0
\[
\text{levpareto(10, 3, scale = 1, order = 2)}
\]

## case with shape - order < 0
\[
\text{levpareto(10, 1.5, scale = 1, order = 2)}
\]

---

Pareto2  The Pareto II Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Pareto II distribution with parameters \(\text{min}, \text{shape}\) and \(\text{scale}\).

Usage

\[
dpareto2(x, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \log = \text{FALSE})
\]
\[
ppareto2(q, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE})
\]
\[
qpareto2(p, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE})
\]
\[
rpareto2(n, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate})
\]
\[
mpareto2(order, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate})
\]
\[
levpareto2(limit, \text{min}, \text{shape}, \text{rate} = 1, \text{scale} = 1/\text{rate}, \text{order} = 1)
\]

Arguments

- **x, q** vector of quantiles.
- **p** vector of probabilities.
- **n** number of observations. If \(\text{length}(n) > 1\), the length is taken to be the number required.
- **min** lower bound of the support of the distribution.
- **shape, scale** parameters. Must be strictly positive.
- **rate** an alternative way to specify the scale.
- **log, log.p** logical; if \(\text{TRUE}\), probabilities/densities \(p\) are returned as \(\log(p)\).
- **lower.tail** logical; if \(\text{TRUE}\) (default), probabilities are \(P[X \leq x]\), otherwise, \(P[X > x]\).
- **order** order of the moment.
- **limit** limit of the loss variable.
The Pareto II (or “type II”) distribution with parameters $\min = \mu$, shape $= \alpha$ and scale $= \theta$ has density:

$$f(x) = \frac{\alpha}{\theta} \left[1 + \frac{x - \mu}{\theta}\right]^{\alpha + 1}$$

for $x > \mu$, $-\infty < \mu < \infty$, $\alpha > 0$ and $\theta > 0$.

The Pareto II is the distribution of the random variable

$$\mu + \theta \left(\frac{X}{1-X}\right),$$

where $X$ has a beta distribution with parameters $1$ and $\alpha$. It derives from the Feller-Pareto distribution with $\tau = \gamma = 1$. Setting $\mu = 0$ yields the familiar Pareto distribution.

The Pareto I (or Single parameter Pareto) distribution is a special case of the Pareto II with $\min = \text{scale}$.

The $k$th raw moment of the random variable $X$ is $E[X^k]$ for nonnegative integer values of $k < \alpha$.

The $k$th limited moment at some limit $d$ is $E[\min(X,d)^k]$ for nonnegative integer values of $k$ and $\alpha - j$, $j = 1, \ldots, k$ not a negative integer.

Invalid arguments will result in return value NaN, with a warning.

Note

levpareto2 computes the limited expected value using beta.int.

For Pareto distributions, we use the classification of Arnold (2015) with the parametrization of Klugman et al. (2012).

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


The Pareto III Distribution

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Pareto III distribution with parameters min, shape and scale.

Usage

dpareto3(x, min, shape, rate = 1, scale = 1/rate, log = FALSE)
ppareto3(q, min, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qpareto3(p, min, shape, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rpareto3(n, min, shape, rate = 1, scale = 1/rate)
mpareto3(order, min, shape, rate = 1, scale = 1/rate)
levpareto3(limit, min, shape, rate = 1, scale = 1/rate, order = 1)

Arguments

- x, q: vector of quantiles.
- p: vector of probabilities.
- n: number of observations. If length(n) > 1, the length is taken to be the number required.
- min: lower bound of the support of the distribution.
- shape, scale: parameters. Must be strictly positive.
rate

an alternative way to specify the scale.

log, log.p

logical; if TRUE, probabilities/densities \( p \) are returned as \( \log(p) \).

lower.tail

logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).

order

order of the moment.

limit

limit of the loss variable.

Details

The Pareto III (or “type III”) distribution with parameters \( \text{min} = \mu \), \text{shape} = \gamma \) and \text{scale} = \theta \) has density:

\[
f(x) = \frac{\gamma((x - \mu)/\theta)^{\gamma-1}}{\theta[1 + ((x - \mu)/\theta)^{\gamma}]^2}
\]

for \( x > \mu \), \(-\infty < \mu < \infty\), \( \gamma > 0 \) and \( \theta > 0 \).

The Pareto III is the distribution of the random variable

\[
\mu + \theta \left( \frac{X}{1 - X} \right)^{1/\gamma},
\]

where \( X \) has a uniform distribution on \((0, 1)\). It derives from the Feller-Pareto distribution with \( \alpha = \tau = 1 \). Setting \( \mu = 0 \) yields the loglogistic distribution.

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) for nonnegative integer values of \( k < \gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X^k, d)] \) for nonnegative integer values of \( k \) and \( 1 - j/\gamma \), \( j = 1, \ldots, k \) not a negative integer.

Value

dpareto3 gives the density, ppareto3 gives the distribution function, qpareto3 gives the quantile function, rpareto3 generates random deviates, mpareto3 gives the \( k \)th raw moment, and levpareto3 gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

levpareto3 computes the limited expected value using betaint.

For Pareto distributions, we use the classification of Arnold (2015) with the parametrization of Klugman et al. (2012).

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>
References


See Also

dllogis for the loglogistic distribution.

Examples

```r
dpareto4(1, min = 10, 3, 4, log = TRUE))
p <- (1:10)/10
ppareto4(qpareto4(p, min = 10, 2, 3), min = 10, 2, 3)

## mean
mpareto4(1, min = 10, 2, 3)

## case with 1 - order/shape > 0
levpareto4(20, min = 10, 2, 3, order = 1)

## case with 1 - order/shape < 0
levpareto4(20, min = 10, 2/3, 3, order = 1)
```

---

The Pareto IV Distribution

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Pareto IV distribution with parameters min, shape1, shape2 and scale.

Usage

```r
dpareto4(x, min, shape1, shape2, rate = 1, scale = 1/rate, log = FALSE)
ppareto4(q, min, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qpareto4(p, min, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rpareto4(n, min, shape1, shape2, rate = 1, scale = 1/rate)
mpareto4(order, min, shape1, shape2, rate = 1, scale = 1/rate)
levpareto4(limit, min, shape1, shape2, rate = 1, scale = 1/rate, order = 1)
```
Arguments

- **x, q**: vector of quantiles.
- **p**: vector of probabilities.
- **n**: number of observations. If \( \text{length}(n) > 1 \), the length is taken to be the number required.
- **min**: lower bound of the support of the distribution.
- **shape1, shape2, scale**: parameters. Must be strictly positive.
- **rate**: an alternative way to specify the scale.
- **log, log.p**: logical; if TRUE, probabilities/densities \( p \) are returned as \( \log(p) \).
- **lower.tail**: logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
- **order**: order of the moment.
- **limit**: limit of the loss variable.

Details

The Pareto IV (or “type IV”) distribution with parameters \( \text{min} = \mu \), \( \text{shape1} = \alpha \), \( \text{shape2} = \gamma \) and \( \text{scale} = \theta \) has density:

\[
f(x) = \frac{\alpha \gamma ((x - \mu)/\theta) ^ {\gamma - 1}}{\theta [1 + ((x - \mu)/\theta) ^ {\gamma}] ^ {\alpha + 1}}
\]

for \( x > \mu \), \( -\infty < \mu < \infty \), \( \alpha > 0 \), \( \gamma > 0 \) and \( \theta > 0 \).

The Pareto IV is the distribution of the random variable

\[
\mu + \theta \left( \frac{X}{1 - X} \right)^{1/\gamma}
\]

where \( X \) has a beta distribution with parameters 1 and \( \alpha \). It derives from the Feller-Pareto distribution with \( \tau = 1 \). Setting \( \mu = 0 \) yields the Burr distribution.

The Pareto IV distribution also has the following direct special cases:

- A Pareto III distribution when \( \text{shape1} = 1 \);
- A Pareto II distribution when \( \text{shape1} = 1 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) for nonnegative integer values of \( k < \alpha \gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k] \) for nonnegative integer values of \( k \) and \( \alpha - j/\gamma, j = 1, \ldots, k \) not a negative integer.

Value

dpareto4 gives the density, ppareto4 gives the distribution function, qpareto4 gives the quantile function, rpareto4 generates random deviates, mpareto4 gives the \( k \)th raw moment, and levpareto4 gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.
Note
levpareto4 computes the limited expected value using betaint.

For Pareto distributions, we use the classification of Arnold (2015) with the parametrization of Klugman et al. (2012).
The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>

References

See Also
dburr for the Burr distribution.

Examples
exp(dpareto4(1, min = 10, 2, 3, log = TRUE))
p <- (1:10)/10
ppareto4(qpareto4(p, min = 10, 2, 3, 2), min = 10, 2, 3, 2)

## variance
mpareto4(2, min = 10, 2, 3, 1) - mpareto4(1, min = 10, 2, 3, 1) ^ 2

## case with shape1 - order/shape2 > 0
levpareto4(10, min = 10, 2, 3, 1, order = 2)

## case with shape1 - order/shape2 < 0
levpareto4(10, min = 10, 1.5, 0.5, 1, order = 2)

PhaseType
The Phase-type Distribution

Description
Density, distribution function, random generation, raw moments and moment generating function for the (continuous) Phase-type distribution with parameters prob and rates.
Usage

dphtype(x, prob, rates, log = FALSE)
pphtype(q, prob, rates, lower.tail = TRUE, log.p = FALSE)
rphype(n, prob, rates)
mhype(order, prob, rates)
mgfphpype(t, prob, rates, log = FALSE)

Arguments

x, q  vector of quantiles.
n  number of observations. If length(n) > 1, the length is taken to be the number required.
prob  vector of initial probabilities for each of the transient states of the underlying Markov chain. The initial probability of the absorbing state is 1 - sum(prob).
rates  square matrix of the rates of transition among the states of the underlying Markov chain.
log, log.p  logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail  logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).
order  order of the moment.
t  numeric vector.

Details

The phase-type distribution with parameters \( \pi \) and \( T \) has density:

\[
f(x) = \pi e^{Tx} t
\]

for \( x \geq 0 \) and \( f(0) = 1 - \pi e \), where \( e \) is a column vector with all components equal to one, \( t = -Te \) is the exit rates vector and \( e^{Tx} \) denotes the matrix exponential of \( Tx \). The matrix exponential of a matrix \( M \) is defined as the Taylor series

\[
e^M = \sum_{n=0}^{\infty} \frac{M^n}{n!}.
\]

The parameters of the distribution must satisfy \( \pi e \leq 1 \), \( T_{ii} < 0 \), \( T_{ij} \geq 0 \) and \( Te \leq 0 \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) and the moment generating function is \( E[e^{tX}] \).

Value

dphase gives the density, pphase gives the distribution function, rphase gives random deviates, mphase gives the \( k \)th raw moment, and mgfphase gives the moment generating function in \( x \).

Invalid arguments will result in return value NaN, with a warning.
Note
The "distributions" package vignette provides the interrelations between the continuous size distributions in *actuar* and the complete formulas underlying the above functions.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Christophe Dutang

References

https://en.wikipedia.org/wiki/Phase-type_distribution


Examples

```r
## Erlang(3, 2) distribution
T <- cbind(c(-2, 0, 0), c(2, -2, 0), c(0, 2, -2))
pi <- c(1,0,0)
x <- 0:10
dphptype(x, pi, T) # density
dgamma(x, 3, 2) # same
pphptype(x, pi, T) # cdf
pgamma(x, 3, 2) # same
rphtype(10, pi, T) # random values
mptype(1, pi, T) # expected value
curve(mgfphtype(x, pi, T), from = -10, to = 1)
```

---

**PoissonInverseGaussian**

*The Poisson-Inverse Gaussian Distribution*

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Density function, distribution function, quantile function and random generation for the Poisson-inverse Gaussian discrete distribution with parameters mean and shape.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dpoisinvgauss(x, mean, shape = 1, dispersion = 1/shape, log = FALSE)</code></td>
</tr>
<tr>
<td><code>ppoisinvgauss(q, mean, shape = 1, dispersion = 1/shape, lower.tail = TRUE, log.p = FALSE)</code></td>
</tr>
<tr>
<td><code>qpoisinvgauss(p, mean, shape = 1, dispersion = 1/shape, lower.tail = TRUE, log.p = FALSE)</code></td>
</tr>
<tr>
<td><code>rpoisinvgauss(n, mean, shape = 1, dispersion = 1/shape)</code></td>
</tr>
</tbody>
</table>
Arguments

- `x` vector of (positive integer) quantiles.
- `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `mean`, `shape` parameters. Must be strictly positive. Infinite values are supported.
- `dispersion` an alternative way to specify the shape.
- `log`, `log.p` logical; if `TRUE`, probabilities `p` are returned as `log(p)`.
- `lower.tail` logical; if `TRUE` (default), probabilities are `P[X ≤ x]`, otherwise, `P[X > x]`.

Details

The Poisson-inverse Gaussian distribution is the result of the continuous mixture between a Poisson distribution and an inverse Gaussian, that is, the distribution with probability mass function

\[ p(x) = \int_0^\infty \frac{\lambda^x e^{-\lambda}}{x!} g(\lambda; \mu, \phi) \, d\lambda, \]

where \( g(\lambda; \mu, \phi) \) is the density function of the inverse Gaussian distribution with parameters `mean = \mu` and `dispersion = \phi` (see `dinvgauss`).

The resulting probability mass function is

\[ p(x) = \sqrt{\frac{2}{\pi \phi}} \frac{e^{(\phi \mu)^{-1}}}{x!} \left( \sqrt{2\phi \left( 1 + \frac{1}{2\phi \mu^2} \right)} \right)^{(x-\frac{1}{2})} K_{x-\frac{1}{2}} \left( \sqrt{2 \phi \left( 1 + \frac{1}{2\phi \mu^2} \right)} \right), \]

for \( x = 0, 1, \ldots, \mu > 0, \phi > 0 \) and where \( K_{\nu}(x) \) is the modified Bessel function of the third kind implemented by R’s `besselK()` and defined in its help.

The limiting case \( \mu = \infty \) has well defined probability mass and distribution functions, but has no finite strictly positive, integer moments. The pmf in this case reduces to

\[ p(x) = \sqrt{\frac{2}{\pi \phi}} \frac{1}{x!} \left( \sqrt{2\phi} \right)^{(x-\frac{1}{2})} K_{x-\frac{1}{2}} \left( \sqrt{2\phi} \right). \]

The limiting case \( \phi = 0 \) is a degenerate distribution in \( x = 0 \).

If an element of `x` is not integer, the result of `dpoisinvgauss` is zero, with a warning.

The quantile is defined as the smallest value `x` such that \( F(x) \geq p \), where `F` is the distribution function.

Value

dpoisinvgauss gives the probability mass function, ppoisinvgauss gives the distribution function, qpoisinvgauss gives the quantile function, and rpoisinvgauss generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the result is determined by `n` for `rpoisinvgauss`, and is the maximum of the lengths of the numerical arguments for the other functions.
Note

[dpqr]pig are aliases for [dpqr]poisinvgauss.

qpoisinvgauss is based on qbinom et al.; it uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

dpois for the Poisson distribution, dinvgauss for the inverse Gaussian distribution.

Examples

```r
## Tables I and II of Shaban (1981)
x <- 0:2
sapply(c(0.4, 0.8, 1), dpoisinvgauss, x = x, mean = 0.1)
sapply(c(40, 80, 100, 130), dpoisinvgauss, x = x, mean = 1)
qpoisinvgauss(ppoisinvgauss(0:10, 1, dis = 2.5), 1, dis = 2.5)
x <- rpoisinvgauss(1000, 1, dis = 2.5)
y <- sort(unique(x))
plot(y, table(x)/length(x), type = "h", lwd = 2,
pch = 19, col = "black", xlab = "x", ylab = "p(x)",
main = "Empirical vs theoretical probabilities")
points(y, dpoisinvgauss(y, 1, dis = 2.5),
pch = 19, col = "red")
legend("topright", c("empirical", "theoretical"),
1ty = c(1, NA), pch = c(NA, 19), col = c("black", "red"))
```
quantile.aggregateDist

Quantiles of Aggregate Claim Amount Distribution

Description

Quantile and Value-at-Risk methods for objects of class "aggregateDist".

Usage

```r
## S3 method for class 'aggregateDist'
quantile(x,
  probs = c(0.25, 0.5, 0.75, 0.9, 0.95, 0.975, 0.99, 0.995),
  smooth = FALSE, names = TRUE, ...)

## S3 method for class 'aggregateDist'
VaR(x, conf.level = c(0.9, 0.95, 0.99),
  smooth = FALSE, names = TRUE, ...)
```

Arguments

- `x`: an object of class "aggregateDist".
- `probs, conf.level`: numeric vector of probabilities with values in \([0, 1)\).
- `smooth`: logical; when TRUE and x is a step function, quantiles are linearly interpolated between knots.
- `names`: logical; if true, the result has a names attribute. Set to FALSE for speedup with many probs.
- `...`: further arguments passed to or from other methods.

Details

The quantiles are taken directly from the cumulative distribution function defined in x. Linear interpolation is available for step functions.

Value

A numeric vector, named if names is TRUE.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Louis-Philippe Pouliot

See Also

`aggregateDist`
quantile.grouped.data

Examples

```r
model.freq <- expression(data = rpois(3))
model.sev <- expression(data = rlnorm(10, 1.5))
Fs <- aggregateDist("simulation", model.freq, model.sev, nb.simul = 1000)
quantile(Fs, probs = c(0.25, 0.5, 0.75))
VaR(Fs)
```

quantile.grouped.data  Quantiles of Grouped Data

Description

Sample quantiles corresponding to the given probabilities for objects of class "grouped.data".

Usage

```r
## S3 method for class 'grouped.data'
quantile(x, probs = seq(0, 1, 0.25),
         names = TRUE, ...)

## S3 method for class 'grouped.data'
summary(object, ...)
```

Arguments

- `x, object`: an object of class "grouped.data".
- `probs`: numeric vector of probabilities with values in [0, 1].
- `names`: logical; if true, the result has a names attribute. Set to FALSE for speedup with many probs.
- `...`: further arguments passed to or from other methods.

Details

The quantile function is the inverse of the ogive, that is a linear interpolation of the empirical quantile function.

The equation of the quantile function is

\[
x = \frac{c_j(F_n(c_{j-1}) - q) + c_{j-1}(q - F_n(c_j))}{F_n(c_j) - F_n(c_{j-1})}
\]

for \(0 \leq q \leq c_j\) and where \(c_0, \ldots, c_r\) are the \(r + 1\) group boundaries and \(F_n\) is the empirical distribution function of the sample.

Value

For quantile, a numeric vector, named if names is TRUE.

For the summary method, an object of class c("summaryDefault", "table") which has specialized format and print methods.
Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>

See Also
ogive for the smoothed empirical distribution of which quantile.grouped.data is an inverse; mean.grouped.data and var.grouped.data to compute the mean and variance of grouped data. grouped.data to create grouped data objects.

Examples
data(gdental)
quantile(gdental)
summary(gdental)
Fn <- ogive(gdental)
Fn(quantile(gdental)) # inverse function

rcomphierarc Simulation from Compound Hierarchical Models

Description
Simulate data for insurance applications allowing hierarchical structures and separate models for the frequency and severity of claims distributions.

Usage
rcomphierarc(nodes, model.freq = NULL, model.sev = NULL, weights = NULL)

## S3 method for class 'portfolio'
print(x, ...)

Arguments
nodes a vector or a named list giving the number of "nodes" at each level in the hierarchy of the portfolio. The nodes are listed from top (portfolio) to bottom (usually the years of experience).
model.freq a named vector of expressions specifying the frequency of claims model (see Details); if NULL, only claim amounts are simulated.
model.sev a named vector of expressions specifying the severity of claims model (see Details); if NULL, only claim numbers are simulated.
weights a vector of weights.
x a portfolio object.
... potential further arguments required by generic.
Details

The order and the names of the elements in `nodes`, `model.freq` and `model.sev` must match. At least one of `model.freq` and `model.sev` must be non-NULL.

`nodes` may be a basic vector, named or not, for non hierarchical models. The rule above still applies, so `model.freq` and `model.sev` should not be named if `nodes` is not. However, for non hierarchical models, `rcompound` is faster and has a simpler interface.

`nodes` specifies the hierarchical layout of the portfolio. Each element of the list is a vector of the number of nodes at a given level. Vectors are recycled as necessary.

`model.freq` and `model.sev` specify the simulation models for claim numbers and claim amounts, respectively. A model is expressed in a semi-symbolic fashion using an object of mode `expression`. Each element of the object must be named and should be a complete call to a random number generation function, with the number of variates omitted. Hierarchical (or mixtures of) models are achieved by replacing one or more parameters of a distribution at a given level by any combination of the names of the levels above. If no mixing is to take place at a level, the model for this level can be NULL.

The argument of the random number generation functions for the number of variates to simulate must be named `n`.

Weights will be used wherever the name "weights" appears in a model. It is the user's responsibility to ensure that the length of `weights` will match the number of nodes when weights are to be used. Normally, there should be one weight per node at the lowest level of the model.

Data is generated in lexicographic order, that is by row in the output matrix.

Value

An object of class "portfolio". A print method for this class displays the models used in the simulation as well as the frequency of claims for each year and entity in the portfolio.

An object of class "portfolio" is a list containing the following components:

- `data` a two dimension list where each element is a vector of claim amounts;
- `weights` the vector of weights given in argument reshaped as a matrix matching element `data`, or NULL;
- `classification` a matrix of integers where each row is a unique set of subscripts identifying an entity in the portfolio (e.g. integers `i`, `j` and `k` for data `X_{ijkt}`);
- `nodes` the nodes argument, appropriately recycled;
- `model.freq` the frequency model as given in argument;
- `model.sev` the severity model as given in argument.

It is recommended to manipulate objects of class "portfolio" by means of the corresponding methods of functions `aggregate`, `frequency` and `severity`.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Sébastien Auclair and Louis-Philippe Pouliot
References


See Also

*rcomphierarc.summaries* for the functions to create the matrices of aggregate claim amounts, frequencies and individual claim amounts.

*rcompound* for a simpler and much faster way to generate variates from standard, non hierarchical, compound models.

Examples

```r
## Two level (contracts and years) portfolio with frequency model
## Nit|Theta_i ~ Poisson(Theta_i), Theta_i ~ Gamma(2, 3) and severity
## model X ~ Lognormal(5, 1)
rcomphierarc(nodes = list(contract = 10, year = 5),
             model.freq = expression(contract = rgamma(2, 3),
                                      year = rpois(contract)),
             model.sev = expression(contract = NULL,
                                     year = rlnorm(5, 1)))

## Model with weights and mixtures for both frequency and severity
## models
nodes <- list(entity = 8, year = c(5, 4, 4, 5, 3, 4, 5, 3))
mf <- expression(entity = rgamma(2, 3),
                 year = rpois(weights * entity))
ms <- expression(entity = rnorm(5, 1),
                 year = rlnorm(entity, 1))
wit <- sample(2:10, 35, replace = TRUE)
pf <- rcomphierarc(nodes, mf, ms, wit)
pf # print method
weights(pf) # extraction of weights
aggregate(pf)[, -1]/weights(pf)[, -1] # ratios

## Four level hierarchical model for frequency only
nodes <- list(sector = 3, unit = c(3, 4),
             employer = c(3, 4, 3, 4, 2, 3, 4), year = 5)
mf <- expression(sector = rexp(1),
                 unit = rexp(sector),
                 employer = rgamma(unit, 1),
                 year = rpois(employer))
pf <- rcomphierarc(nodes, mf, NULL)
pf # print method
aggregate(pf) # aggregate claim amounts
frequency(pf) # frequencies
severity(pf) # individual claim amounts

## Standard, non hierarchical, compound model with simplified
## syntax (function rcompound() is much faster for such cases)
rcomphierarc(10,
```

Summary Statistics of a Portfolio

Description

Methods for class "portfolio" objects.
aggregate splits portfolio data into subsets and computes summary statistics for each.
frequency computes the frequency of claims for subsets of portfolio data.
severity extracts the individual claim amounts.
weights extracts the matrix of weights.

Usage

## S3 method for class 'portfolio'
aggregate(x, by = names(x$nodes), FUN = sum,
classification = TRUE, prefix = NULL, ...)

## S3 method for class 'portfolio'
frequency(x, by = names(x$nodes),
classification = TRUE, prefix = NULL, ...)

## S3 method for class 'portfolio'
severity(x, by = head(names(x$node), -1), splitcol = NULL,
classification = TRUE, prefix = NULL, ...)

## S3 method for class 'portfolio'
weights(object, classification = TRUE, prefix = NULL, ...)

Arguments

x, object an object of class "portfolio", typically created with simul.
by character vector of grouping elements using the level names of the portfolio in x. The names can be abbreviated.
FUN the function to be applied to data subsets.
classification boolean; if TRUE, the node identifier columns are included in the output.
prefix characters to prefix column names with; if NULL, sensible defaults are used when appropriate.
splitcol columns of the data matrix to extract separately; usual matrix indexing methods are supported.
... optional arguments to FUN, or passed to or from other methods.
Details

By default, aggregate.portfolio computes the aggregate claim amounts for the grouping specified in by. Any other statistic based on the individual claim amounts can be used through argument FUN.

frequency.portfolio is equivalent to using aggregate.portfolio with argument FUN equal to if (identical(x, NA)) NA else length(x).

severity.portfolio extracts individual claim amounts of a portfolio by groupings using the default method of severity. Argument splitcol allows to get the individual claim amounts of specific columns separately.

weights.portfolio extracts the weight matrix of a portfolio.

Value

A matrix or vector depending on the groupings specified in by.

For the aggregate and frequency methods: if at least one level other than the last one is used for grouping, the result is a matrix obtained by binding the appropriate node identifiers extracted from x$classification if classification = TRUE, and the summaries per grouping. If the last level is used for grouping, the column names of x$data are retained; if the last level is not used for grouping, the column name is replaced by the deparsed name of FUN. If only the last level is used (column summaries), a named vector is returned.

For the severity method: a list of two elements:

main NULL or a matrix of claim amounts for the columns not specified in splitcol, with the appropriate node identifiers extracted from x$classification if classification = TRUE;

split same as above, but for the columns specified in splitcol.

For the weights method: the weight matrix of the portfolio with node identifiers if classification = TRUE.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>, Louis-Philippe Pouliot.

See Also

rcomphierarc

Examples

nodes <- list(sector = 3, unit = c(3, 4),
 employer = c(3, 4, 3, 4, 2, 3, 4), year = 5)
model.freq <- expression(sector = rexp(1),
 unit = rexp(sector),
 employer = rgamma(unit, 1),
 year = rpois(employer))
model.sev <- expression(sector = rnorm(6, 0.1),
 unit = rnorm(sector, 1),

rcompound

Simulation from Compound Models

Description

rcompound generates random variates from a compound model.
 rcomppois is a simplified version for a common case.

Usage

rcompound(n, model.freq, model.sev, SIMPLIFY = TRUE)

rcomppois(n, lambda, model.sev, SIMPLIFY = TRUE)

Arguments

n
number of observations. If length(n) > 1, the length is taken to be the number required.

model.freq, model.sev
expressions specifying the frequency and severity simulation models with the number of variates omitted; see Details.

lambda
Poisson parameter.

SIMPLIFY
boolean; if FALSE the frequency and severity variates are returned along with the aggregate variates.
Details

`rcompound` generates variates from a random variable of the form

\[ S = X_1 + \ldots X_N, \]

where \( N \) is the frequency random variable and \( X_1, X_2, \ldots \) are the severity random variables. The latter are mutually independent, identically distributed and independent from \( N \).

`model.freq` and `model.sev` specify the simulation models for the frequency and the severity random variables, respectively. A model is a complete call to a random number generation function, with the number of variates omitted. This is similar to `rcomphierarc`, but the calls need not be wrapped into `expression`. Either argument may also be the name of an object containing an expression, in which case the object will be evaluated in the parent frame to retrieve the expression.

The argument of the random number generation functions for the number of variates to simulate must be named `n`.

`rcomppois` generates variates from the common Compound Poisson model, that is when random variable \( N \) is Poisson distributed with mean `lambda`.

Value

When `SIMPLIFY = TRUE`, a vector of aggregate amounts \( S_1, \ldots, S_n \).

When `SIMPLIFY = FALSE`, a list of three elements:

- `aggregate`: vector of aggregate amounts \( S_1, \ldots, S_n \);
- `frequency`: vector of frequencies \( N_1, \ldots, N_n \);
- `severity`: vector of severities \( X_1, X_2, \ldots \).

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

See Also

`rcomphierarc` to simulate from compound hierarchical models.

Examples

```r
## Compound Poisson model with gamma severity.
rcompound(10, rpois(2), rgamma(2, 3))
rcomppois(10, 2, rgamma(2, 3)) # same

## Frequencies and individual claim amounts along with aggregate
## values.
rcomppois(10, 2, rgamma(2, 3), SIMPLIFY = FALSE)

## Wrapping the simulation models into expression() is allowed, but
## not needed.
rcompound(10, expression(rpois(2)), expression(rgamma(2, 3)))

## Not run: ## Speed comparison between rcompound() and rcomphierarc().
```
# Simulation from Discrete Mixtures

**Description**

Generate random variates from a discrete mixture of distributions.

**Usage**

```r
rmixture(n, probs, models, shuffle = TRUE)
```

**Arguments**

- `n`: number of random variates to generate. If `length(n) > 1`, the length is taken to be the number required.
- `probs`: numeric non-negative vector specifying the probability for each model; is internally normalized to sum 1. Infinite and missing values are not allowed. Values are recycled as necessary to match the length of `models`.
- `models`: vector of expressions specifying the simulation models with the number of variates omitted; see Details. Models are recycled as necessary to match the length of `probs`.
- `shuffle`: logical; should the random variates from the distributions be shuffled?

**Details**

`rmixture` generates variates from a discrete mixture, that is the random variable with a probability density function of the form

\[ f(x) = p_1 f_1(x) + \ldots + p_n f_n(x), \]

where \( f_1, \ldots, f_n \) are densities and \( \sum_{i=1}^n p_i = 1 \).

The values in `probs` will be internally normalized to be used as probabilities \( p_1 + \cdots + p_n \).

The specification of simulation models uses the syntax of `rcomphierarc`. Models \( f_1, \ldots, f_n \) are expressed in a semi-symbolic fashion using an object of mode `expression` where each element is a complete call to a random number generation function, with the number of variates omitted.

The argument of the random number generation functions for the number of variates to simulate must be named `n`. 

```r
## Also note the simpler syntax for rcompound().
system.time(rcompound(1e6, rpois(2), rgamma(2, 3)))
system.time(rcomphierarc(1e6, expression(rpois(2)), expression(rgamma(2, 3))))
## End(Not run)
## The severity can itself be a compound model. It makes sense
## in such a case to use a zero-truncated frequency distribution
## for the second level model.
rcomppois(10, 2,
        rcompound(rztnbinom(1.5, 0.7), rlnorm(1.2, 1)))

```

---

---
If `shuffle` is `FALSE`, the output vector contains all the random variates from the first model, then all the random variates from the second model, and so on. If the order of the variates is irrelevant, this cuts the time to generate the variates roughly in half.

**Value**

A vector of random variates from the mixture with density $f(x)$.

**Note**

Building the expressions in `models` from the arguments of another function is delicate. The expressions must be such that evaluation is possible in the frame of `rmixture` or its parent. See the examples.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**See Also**

- `rcompound` to simulate from compound models.
- `rcomphierarc` to simulate from compound hierarchical models.

**Examples**

```r
## Mixture of two exponentials (with means 1/3 and 1/7) with equal
## probabilities.
rmixture(10, 0.5, expression(rexp(3), rexp(7)))
rmixture(10, 42, expression(rexp(3), rexp(7))) # same

## Mixture of two lognormals with different probabilities.
rmixture(10, probs = c(0.55, 0.45),
        models = expression(rlnorm(3.6, 0.6),
                            rlnorm(4.6, 0.3)))

## Building the model expressions in the following example
## works as 'rate' is defined in the parent frame of
## 'rmixture'.
probs <- c(2, 5)
g <- function(n, p, rate)
  rmixture(n, p, expression(rexp(rate[1]), rexp(rate[2])))
g(10, probs, c(3, 7))

## The following example does not work: 'rate' does not exist
## in the evaluation frame of 'rmixture'.
f <- function(n, p, model) rmixture(n, p, model)
h <- function(n, p, rate)
  f(n, p, expression(rexp(rate[1]), rexp(rate[2])))
## Not run: h(10, probs, c(3, 7))

## Fix: substitute the values in the model expressions.
```
ruin

Description
Calulation of infinite time probability of ruin in the models of Cramér-Lundberg and Sparre Andersen, that is with exponential or phase-type (including mixtures of exponentials, Erlang and mixture of Erlang) claims interarrival time.

Usage
ruin(claims = c("exponential", "Erlang", "phase-type"), par.claims, wait = c("exponential", "Erlang", "phase-type"), par.wait, premium.rate = 1, tol = sqrt(.Machine$double.eps), maxit = 200L, echo = FALSE)

## S3 method for class 'ruin'
plot(x, from = NULL, to = NULL, add = FALSE, xlab = "u", ylab = expression(psi(u)), main = "Probability of Ruin", xlim = NULL, ...)

Arguments
claims character; the type of claim severity distribution.
wait character; the type of claim interarrival (wait) time distribution.
par.claims, par.wait named list containing the parameters of the distribution; see Details.
premium.rate numeric list of length 1; the premium rate.
tol, maxit, echo respectively the tolerance level of the stopping criteria, the maximum number of iterations and whether or not to echo the procedure when the transition rates matrix is determined iteratively. Ignored if wait = "exponential".
x an object of class "ruin".
from, to the range over which the function will be plotted.
add logical; if TRUE add to already existing plot.
xlim numeric of length 2; if specified, it serves as default for c(from, to).
xlab, ylab label of the x and y axes, respectively.
main main title.
... further graphical parameters accepted by curve.
Details
The names of the parameters in `par.claims` and `par.wait` must the same as in `dexp`, `dgamma` or `dphtype`, as appropriate. A model will be a mixture of exponential or Erlang distributions (but not phase-type) when the parameters are vectors of length > 1 and the parameter list contains a vector weights of the coefficients of the mixture.

Parameters are recycled when needed. Their names can be abbreviated.

Combinations of exponentials as defined in Dufresne and Gerber (1988) are not supported.

Ruin probabilities are evaluated using `pphtype` except when both distributions are exponential, in which case an explicit formula is used.

When `wait != "exponential"` (Sparre Andersen model), the transition rate matrix $Q$ of the distribution of the probability of ruin is determined iteratively using a fixed point-like algorithm. The stopping criteria used is

$$\max \left\{ \sum_{j=1}^{n} |Q_{ij} - Q'_{ij}| \right\} < tol,$$

where $Q$ and $Q'$ are two successive values of the matrix.

Value
A function of class "ruin" inheriting from the "function" class to compute the probability of ruin given initial surplus levels. The function has arguments:

- `u` numeric vector of initial surplus levels;
- `survival` logical; if FALSE (default), probabilities are $\psi(u)$, otherwise, $\phi(u) = 1 - \psi(u)$;
- `lower.tail` an alias for `!survival`.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>, and Christophe Dutang

References


Examples
```r
## Case with an explicit formula: exponential claims and exponential interarrival times.
psi <- ruin(claims = "e", par.claims = list(rate = 5),
           wait = "e", par.wait = list(rate = 3))
spi
psi(0:10)
plot(psi, from = 0, to = 10)
```
## Mixture of two exponentials for claims, exponential interarrival times (Gerber 1979)

```r
psi <- ruin(claims = "e", par.claims = list(rate = c(3, 7), w = 0.5),
            wait = "e", par.wait = list(rate = 3), pre = 1)

u <- 0:10
psi(u)
(24 * exp(-u) + exp(-6 * u))/35 # same
```

## Phase-type claims, exponential interarrival times (Asmussen and Rolski 1991)

```r
p <- c(0.5614, 0.4386)
r <- matrix(c(-8.64, 0.101, 1.997, -1.095), 2, 2)
lambda <- 1/(1.1 * mphtype(1, p, r))
psi <- ruin(claims = "p", par.claims = list(prob = p, rates = r),
            wait = "e", par.wait = list(rate = lambda))

plot(psi, xlim = c(0, 50))
```

## Phase-type claims, mixture of two exponentials for interarrival times (Asmussen and Rolski 1991)

```r
a <- (0.4/5 + 0.6) * lambda
ruin(claims = "p", par.claims = list(prob = p, rates = r),
     wait = "e", par.wait = list(rate = c(5 * a, a), weights =
                                 c(0.4, 0.6)),
     maxit = 225L)
```

---

### severity

**Manipulation of Individual Claim Amounts**

#### Description

severity is a generic function created to manipulate individual claim amounts. The function invokes particular methods which depend on the class of the first argument.

#### Usage

```r
severity(x, ...)
```

#### Arguments

- `x` an R object.
- `bycol` logical; whether to “unroll” horizontally (FALSE) or vertically (TRUE)
- `...` further arguments to be passed to or from other methods.
- `drop` logical; if TRUE, the result is coerced to the lowest possible dimension.
**Details**

Currently, the default method is equivalent to `unroll`. This is liable to change since the link between the name and the use of the function is rather weak.

**Value**

A vector or matrix.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Louis-Philippe Pouliot

**See Also**

`severity.portfolio` for the original motivation of these functions.

**Examples**

```r
x <- list(c(1:3), c(1:8), c(1:4), c(1:3))
(mat <- matrix(x, 2, 2))
severity(mat)
severity(mat, bycol = TRUE)
```

---

**Description**

Density function, distribution function, quantile function, random generation, raw moments, and limited moments for the Single-parameter Pareto distribution with parameter `shape`.

**Usage**

```r
dpareto1(x, shape, min, log = FALSE)
ppareto1(q, shape, min, lower.tail = TRUE, log.p = FALSE)
qpareto1(p, shape, min, lower.tail = TRUE, log.p = FALSE)
rpareto1(n, shape, min)
mpareto1(order, shape, min)
levpareto1(limit, shape, min, order = 1)
```

**Arguments**

- `x, q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `shape` parameter. Must be strictly positive.
The single-parameter Pareto, or Pareto I, distribution with parameter shape $\alpha$ has density:

$$f(x) = \frac{\alpha \theta^\alpha}{x^{\alpha+1}}$$

for $x > \theta$, $\alpha > 0$ and $\theta > 0$.

Although there appears to be two parameters, only shape is a true parameter. The value of $\min = \theta$ must be set in advance.

The $k$th raw moment of the random variable $X$ is $E[X^k]$, $k < \alpha$ and the $k$th limited moment at some limit $d$ is $E[\min(X,d)^k]$, $x \geq \theta$.

Value

dpareto1 gives the density, ppareto1 gives the distribution function, qpareto1 gives the quantile function, rpareto1 generates random deviates, mpareto1 gives the $k$th raw moment, and levpareto1 gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

For Pareto distributions, we use the classification of Arnold (2015) with the parametrization of Klugman et al. (2012).

The "distributions" package vignette provides the interrelations between the continuous size distributions in actuar and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

dpareto for the two-parameter Pareto distribution.
TransformedBeta

The Transformed Beta Distribution

Description
Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Transformed Beta distribution with parameters shape1, shape2, shape3 and scale.

Usage

dtrbeta(x, shape1, shape2, shape3, rate = 1, scale = 1/rate, log = FALSE)
ptrbeta(q, shape1, shape2, shape3, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qtrbeta(p, shape1, shape2, shape3, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rtrbeta(n, shape1, shape2, shape3, rate = 1, scale = 1/rate)
mtrbeta(order, shape1, shape2, shape3, rate = 1, scale = 1/rate)
levtrbeta(limit, shape1, shape2, shape3, rate = 1, scale = 1/rate, order = 1)

Arguments
x, q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
shape1, shape2, shape3, scale parameters. Must be strictly positive.
rate an alternative way to specify the scale.
log, log.p logical; if TRUE, probabilities/densities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X ≤ x], otherwise, P[X > x].
order order of the moment.
limit limit of the loss variable.
TransformedBeta

Details

The transformed beta distribution with parameters \( \text{shape1} = \alpha \), \( \text{shape2} = \gamma \), \( \text{shape3} = \tau \) and \( \text{scale} = \theta \), has density:

\[
f(x) = \frac{\Gamma(\alpha + \tau)}{\Gamma(\alpha)\Gamma(\tau)} \frac{\gamma^{\tau}}{\theta^{\alpha + \tau}} x^{\alpha - 1} \left( 1 + \frac{x}{\theta} \right)^{-\alpha - \gamma} \]

for \( x > 0 \), \( \alpha > 0 \), \( \gamma > 0 \), \( \tau > 0 \) and \( \theta > 0 \). (Here \( \Gamma(\alpha) \) is the function implemented by \texttt{R}'s \texttt{gamma()} and defined in its help.)

The transformed beta is the distribution of the random variable

\[
\theta \left( \frac{X}{1 - X} \right)^{1/\gamma},
\]

where \( X \) has a beta distribution with parameters \( \tau \) and \( \alpha \).

The transformed beta distribution defines a family of distributions with the following special cases:

- A **Burr** distribution when \( \text{shape3} == 1 \);
- A **loglogistic** distribution when \( \text{shape1} == \text{shape3} == 1 \);
- A **paralogistic** distribution when \( \text{shape3} == 1 \) and \( \text{shape2} == \text{shape1} \);
- A **generalized Pareto** distribution when \( \text{shape2} == 1 \);
- A **Pareto** distribution when \( \text{shape2} == \text{shape3} == 1 \);
- An **inverse Burr** distribution when \( \text{shape1} == 1 \);
- An **inverse Pareto** distribution when \( \text{shape2} == \text{shape1} == 1 \);
- An **inverse paralogistic** distribution when \( \text{shape1} == 1 \) and \( \text{shape3} == \text{shape2} \).

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k], -\tau\gamma < k < \alpha\gamma \).

The \( k \)th limited moment at some limit \( d \) is \( E[\min(X,d)^k], k > -\tau\gamma \) and \( \alpha - k/\gamma \) not a negative integer.

Value

\dtrbeta \ gives the density, \ptrbeta \ gives the distribution function, \qtrbeta \ gives the quantile function, \rtrbeta \ generates random deviates, \mtrbeta \ gives the \( k \)th raw moment, and \levtrbeta \ gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value \texttt{NaN}, with a warning.

Note

\levtrbeta \ computes the limited expected value using \texttt{betaint}.

Distribution also known as the Generalized Beta of the Second Kind and Pearson Type VI. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in \texttt{actuar} and the complete formulas underlying the above functions.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon
TransformedGamma

References


See Also
dfpareto for an equivalent distribution with a location parameter.

Examples

exp(dtrbeta(2, 3, 4, 5, log = TRUE))

p <- (1:10)/10
ptrbeta(qtrbeta(p, 2, 3, 4, 5), 2, 3, 4, 5)
qpearson6(0.3, 2, 3, 4, 5, lower.tail = FALSE)

## variance
mtrbeta(2, 2, 3, 4, 5) - mtrbeta(1, 2, 3, 4, 5)^2

## case with shape1 - order/shape2 > 0
levtrbeta(10, 2, 3, 4, scale = 1, order = 2)

## case with shape1 - order/shape2 < 0
levtrbeta(10, 1/3, 0.75, 4, scale = 0.5, order = 2)

---

TransformedGamma

*The Transformed Gamma Distribution*

Description

Density function, distribution function, quantile function, random generation, raw moments and limited moments for the Transformed Gamma distribution with parameters shape1, shape2 and scale.

Usage

dtrgamma(x, shape1, shape2, rate = 1, scale = 1/rate, log = FALSE)
ptrgamma(q, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
qtrgamma(p, shape1, shape2, rate = 1, scale = 1/rate, lower.tail = TRUE, log.p = FALSE)
rtrgamma(n, shape1, shape2, rate = 1, scale = 1/rate)
mtrgamma(order, shape1, shape2, rate = 1, scale = 1/rate)
levtrgamma(limit, shape1, shape2, rate = 1, scale = 1/rate, order = 1)
TransformedGamma

Arguments

- `x`, `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `shape1`, `shape2`, `scale` parameters. Must be strictly positive.
- `rate` an alternative way to specify the scale.
- `log`, `log.p` logical; if `TRUE`, probabilities/densities `p` are returned as `log(p)`.
- `lower.tail` logical; if `TRUE` (default), probabilities are `P[X ≤ x]`, otherwise, `P[X > x]`.
- `order` order of the moment.
- `limit` limit of the loss variable.

Details

The transformed gamma distribution with parameters `shape1 = α`, `shape2 = τ` and `scale = θ` has density:

\[ f(x) = \frac{τ^{u^α}e^{-u}}{xΓ(α)}, \quad u = (x/θ)^τ \]

for \( x > 0, \alpha > 0, \tau > 0 \) and \( θ > 0 \). (Here \( Γ(α) \) is the function implemented by R’s `gamma()` and defined in its help.)

The transformed gamma is the distribution of the random variable \( θX^{1/τ} \), where \( X \) has a gamma distribution with shape parameter \( α \) and scale parameter \( 1 \) or, equivalently, of the random variable \( Y^{1/τ} \) with \( Y \) a gamma distribution with shape parameter \( α \) and scale parameter \( θ^{τ} \).

The transformed gamma probability distribution defines a family of distributions with the following special cases:

- A Gamma distribution when `shape2 == 1`;
- A Weibull distribution when `shape1 == 1`;
- An Exponential distribution when `shape2 == shape1 == 1`.

The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \) and the \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k], k > -ατ \).

Value

- `dtrgamma` gives the density,
- `ptrgamma` gives the distribution function,
- `qtrgamma` gives the quantile function,
- `rtrgamma` generates random deviates,
- `mtrgamma` gives the \( k \)th raw moment,
- `levtrgamma` gives the \( k \)th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Note

Distribution also known as the Generalized Gamma. See also Kleiber and Kotz (2003) for alternative names and parametrizations.

The "distributions" package vignette provides the interrelations between the continuous size distributions in `actuar` and the complete formulas underlying the above functions.
UniformSupp

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References

Examples
exp(dtrgamma(2, 3, 4, 5, log = TRUE))
p <- (1:10)/10
ptrgamma(qtrgamma(p, 2, 3, 4), 2, 3, 4)
mtrgamma(2, 3, 4, 5) - mtrgamma(1, 3, 4, 5) ^ 2
levtrgamma(10, 3, 4, 5, order = 2)

---

UniformSupp  Moments and Moment Generating Function of the Uniform Distribution

Description
Raw moments, limited moments and moment generating function for the Uniform distribution from min to max.

Usage
munif(order, min = 0, max = 1)
levunif(limit, min = 0, max = 1, order = 1)
mgunif(t, min = 0, max = 1, log = FALSE)

Arguments
order  order of the moment.
min, max  lower and upper limits of the distribution. Must be finite.
limit  limit of the random variable.
t  numeric vector.
log  logical; if TRUE, the cumulant generating function is returned.

Details
The \( k \)th raw moment of the random variable \( X \) is \( E[X^k] \), the \( k \)th limited moment at some limit \( d \) is \( E[\min(X, d)^k] \) and the moment generating function is \( E[e^{tX}] \).
Value
munif gives the $k$th raw moment, levunif gives the $k$th moment of the limited random variable, and mgfunif gives the moment generating function in $t$.
Invalid arguments will result in return value NaN, with a warning.

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>, Christophe Dutang

References
https://en.wikipedia.org/wiki/Uniform_distribution_%28continuous%29

See Also
Uniform.

Examples
munif(-1)
munif(1:5)
levunif(3, order=1:5)
levunif(3, 2, 4)
mgfunif(1, 1, 2)

unroll
Display a Two-Dimension Version of a Matrix of Vectors

Description
Displays all values of a matrix of vectors by “unrolling” the object vertically or horizontally.

Usage
unroll(x, bycol = FALSE, drop = TRUE)

Arguments
x a list of vectors with a dim attribute of length 0, 1 or 2.
bycol logical; whether to unroll horizontally (FALSE) or vertically (TRUE).
drop logical; if TRUE, the result is coerced to the lowest possible dimension.

Details
unroll returns a matrix where elements of x are concatenated (“unrolled”) by row (bycol = FALSE) or by column (bycol = TRUE). NA is used to make rows/columns of equal length. Vectors and one dimensional arrays are coerced to row matrices.
**Value**

A vector or matrix.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Louis-Philippe Pouliot

**See Also**

This function was originally written for use in `severity.portfolio`.

**Examples**

```r
x <- list(c(1:3), c(1:8), c(1:4), c(1:3))
(mat <- matrix(x, 2, 2))
unroll(mat)
unroll(mat, bycol = TRUE)
unroll(mat[1, ])
unroll(mat[1, ], drop = FALSE)
```

---

<table>
<thead>
<tr>
<th>VaR</th>
<th>Value at Risk</th>
</tr>
</thead>
</table>

**Description**

Value at Risk.

**Usage**

`VaR(x, ...)`

**Arguments**

- `x` an R object.
- `...` further arguments passed to or from other methods.

**Details**

This is a generic function with, currently, only a method for objects of class "aggregateDist".

**Value**

An object of class numeric.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Tommy Ouellet
See Also

VaR.aggregateDist.aggregateDist

---

**var**  

**Variance and Standard Deviation**

**Description**

Generic functions for the variance and standard deviation, and methods for individual and grouped data.

The default methods for individual data are the functions from the stats package.

**Usage**

```r
var(x, ...)  
## Default S3 method:  
var(x, y = NULL, na.rm = FALSE, use, ...)  
## S3 method for class 'grouped.data'  
var(x, ...)  
sd(x, ...)  
## Default S3 method:  
sd(x, na.rm = FALSE, ...)  
## S3 method for class 'grouped.data'  
sd(x, ...)
```

**Arguments**

- `x` a vector or matrix of individual data, or an object of class "grouped data".
- `y` see stats::var.
- `na.rm` see stats::var.
- `use` see stats::var.
- `...` further arguments passed to or from other methods.

**Details**

This page documents variance and standard deviation computations for grouped data. For individual data, see var and sd from the stats package.

For grouped data with group boundaries $c_0, c_1, \ldots, c_r$ and group frequencies $n_1, \ldots, n_r$, var computes the sample variance

$$\frac{1}{n-1} \sum_{j=1}^{r} n_j(a_j - m_1)^2,$$
where \( a_j = (c_{j-1} + c_j)/2 \) is the midpoint of the \( j \)th interval, \( m_1 \) is the sample mean (or sample first moment) of the data, and \( n = \sum_{j=1}^{r} n_j \). The sample sample standard deviation is the square root of the sample variance.

The sample variance for grouped data differs from the variance computed from the empirical raw moments with \texttt{emm} in two aspects. First, it takes into account the degrees of freedom. Second, it applies Sheppard’s correction factor to compensate for the overestimation of the true variation in the data. For groups of equal width \( k \), Sheppard’s correction factor is equal to \(-k^2/12\).

**Value**

A named vector of variances or standard deviations.

**Author(s)**

Vincent Goulet \(<\text{vincent.goulet@act.ulaval.ca}>\). Variance and standard deviation methods for grouped data contributed by Walter Garcia-Fontes \(<\text{walter.garcia@upf.edu}>\).

**References**


Heumann, C., Schomaker, M., Shalabh (2016), \textit{Introduction to Statistics and Data Analysis}, Springer.

**See Also**

grouped.data to create grouped data objects; mean.grouped.data for the mean and \texttt{emm} for higher moments.

**Examples**

data(gdental)
var(gdental)
sd(gdental)

## Illustration of Sheppard’s correction factor
cj <- c(0, 2, 4, 6, 8)
nj <- c(1, 5, 3, 2)
gd <- grouped.data(Group = cj, Frequency = nj)
(sum(nj) - 1)/sum(nj) * var(gd)
(emm(gd, 2) - emm(gd)^2) - 4/12

---

**WeibullMoments**

Raw and Limited Moments of the Weibull Distribution

**Description**

Raw moments and limited moments for the Weibull distribution with parameters shape and scale.
Usage

\texttt{mweibull(order, shape, scale = 1)}
\texttt{levweibull(limit, shape, scale = 1, order = 1)}

Arguments

- \texttt{order} order of the moment.
- \texttt{limit} limit of the loss variable.
- \texttt{shape, scale} shape and scale parameters, the latter defaulting to 1.

Details

The $k$th raw moment of the random variable $X$ is $E[X^k]$ and the $k$th limited moment at some limit $d$ is $E[\min(X, d)^k]$, $k > -\tau$.

Value

\texttt{mweibull} gives the $k$th raw moment and \texttt{levweibull} gives the $k$th moment of the limited loss variable.

Invalid arguments will result in return value NaN, with a warning.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca> and Mathieu Pigeon

References


See Also

\texttt{Weibull} for details on the Weibull distribution and functions \texttt{[dpqr]weibull}.

Examples

\texttt{mweibull(2, 3, 4) - mweibull(1, 3, 4)^2}
\texttt{levweibull(10, 3, 4, order = 2)}
The Zero-Modified Binomial Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Modified Binomial distribution with parameters size and prob, and probability at zero $p_0$.

Usage

dzmbinom(x, size, prob, p0, log = FALSE)
pzmbinom(q, size, prob, p0, lower.tail = TRUE, log.p = FALSE)
qzmbinom(p, size, prob, p0, lower.tail = TRUE, log.p = FALSE)
rzmbinom(n, size, prob, p0)

Arguments

- x: vector of (strictly positive integer) quantiles.
- q: vector of quantiles.
- p: vector of probabilities.
- n: number of observations. If length(n) > 1, the length is taken to be the number required.
- size: number of trials (strictly positive integer).
- prob: probability of success on each trial. $0 \leq \text{prob} \leq 1$.
- p0: probability mass at zero. $0 \leq p_0 \leq 1$.
- log, log.p: logical; if TRUE, probabilities p are returned as log(p).
- lower.tail: logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

Details

The zero-modified binomial distribution with size $= n$, prob $= p$ and p0 $= p_0$ is a discrete mixture between a degenerate distribution at zero and a (standard) binomial. The probability mass function is $p(0) = p_0$ and

$$p(x) = \frac{(1 - p_0)}{1 - (1 - p)^n} f(x)$$

for $x = 1, \ldots, n$, $0 < p \leq 1$ and $0 \leq p_0 \leq 1$, where $f(x)$ is the probability mass function of the binomial. The cumulative distribution function is

$$P(x) = p_0 + (1 - p_0) \left( \frac{F(x) - F(0)}{1 - F(0)} \right)$$

The mean is $(1 - p_0)\mu$ and the variance is $(1 - p_0)\sigma^2 + p_0(1 - p_0)\mu^2$, where $\mu$ and $\sigma^2$ are the mean and variance of the zero-truncated binomial.

In the terminology of Klugman et al. (2012), the zero-modified binomial is a member of the $(a, b, 1)$ class of distributions with $a = -p/(1 - p)$ and $b = (n + 1)p/(1 - p)$. 
The special case $p_0 = 0$ is the zero-truncated binomial.

If an element of $x$ is not integer, the result of $dzmbinom$ is zero, with a warning.

The quantile is defined as the smallest value $x$ such that $P(x) \geq p$, where $P$ is the distribution function.

**Value**

$dzmbinom$ gives the probability mass function, $pzmbinom$ gives the distribution function, $qzmbinom$ gives the quantile function, and $rzmbinom$ generates random deviates.

Invalid size, prob or $p_0$ will result in return value NaN, with a warning.

The length of the result is determined by $n$ for $rzmbinom$, and is the maximum of the lengths of the numerical arguments for the other functions.

**Note**

Functions $\{d,p,q\}zmbinom$ use $\{d,p,q\}binom$ for all but the trivial input values and $p(0)$.

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**References**


**See Also**

`dbinom` for the binomial distribution.

`dztbinom` for the zero-truncated binomial distribution.

**Examples**

```r
dzmbinom(1:5, size = 5, prob = 0.4, p0 = 0.2)
(1-0.2) * dbinom(1:5, 5, 0.4)/pbinom(0, 5, 0.4, lower = FALSE) # same

## simple relation between survival functions
pzmbinom(0:5, 5, 0.4, p0 = 0.2, lower = FALSE)
(1-0.2) * pbinom(0:5, 5, 0.4, lower = FALSE) / pbinom(0, 5, 0.4, lower = FALSE) # same

qzmbinom(pzmbinom(1:10, 10, 0.6, p0 = 0.1), 10, 0.6, p0 = 0.1)
n <- 8; p <- 0.3; p0 <- 0.025
x <- 0:n
title <- paste("ZM Binomial(", n, ", ", p, ", ", p0, ",") and Binomial("", n, ", ", p, ") PDF",
sep = ")
plot(x, dzbinom(x, n, p, p0), type = "h", lwd = 2, ylab = "p(x)",
main = title)
```
The Zero-Modified Geometric Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Modified Geometric distribution with parameter prob and arbitrary probability at zero \( p_0 \).

Usage

- `dzmgeom(x, prob, p0, log = FALSE)`
- `pzmgeom(q, prob, p0, lower.tail = TRUE, log.p = FALSE)`
- `qzmgeom(p, prob, p0, lower.tail = TRUE, log.p = FALSE)`
- `rzmgeom(n, prob, p0)`

Arguments

- `x` vector of (strictly positive integer) quantiles.
- `q` vector of quantiles.
- `p` vector of probabilities.
- `n` number of observations. If `length(n) > 1`, the length is taken to be the number required.
- `prob` parameter. \( 0 < prob \leq 1 \).
- `p0` probability mass at zero. \( 0 \leq p0 \leq 1 \).
- `log, log.p` logical; if TRUE, probabilities \( p \) are returned as \( \log(p) \).
- `lower.tail` logical; if TRUE (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).

Details

The zero-modified geometric distribution with \( prob = p \) and \( p0 = p0 \) is a discrete mixture between a degenerate distribution at zero and a (standard) geometric. The probability mass function is \( p(0) = p0 \) and

\[
p(x) = \frac{(1 - p0)}{(1 - p)} \cdot f(x)
\]

for \( x = 1, 2, \ldots, 0 < p < 1 \) and \( 0 \leq p0 \leq 1 \), where \( f(x) \) is the probability mass function of the geometric. The cumulative distribution function is

\[
P(x) = p0 + (1 - p0) \left( \frac{F(x) - F(0)}{1 - F(0)} \right)
\]

The mean is \( (1 - p0)\mu \) and the variance is \( (1 - p0)\sigma^2 + p0(1 - p0)\mu^2 \), where \( \mu \) and \( \sigma^2 \) are the mean and variance of the zero-truncated geometric.
In the terminology of Klugman et al. (2012), the zero-modified geometric is a member of the $(a, b, 1)$ class of distributions with $a = 1 - p$ and $b = 0$.

The special case $p0 == 0$ is the zero-truncated geometric.

If an element of $x$ is not integer, the result of dzmgeom is zero, with a warning.

The quantile is defined as the smallest value $x$ such that $P(x) \geq p$, where $P$ is the distribution function.

Value
dzmgeom gives the (log) probability mass function, pzmgeom gives the (log) distribution function, qzmgeom gives the quantile function, and rzmgeom generates random deviates.

Invalid prob or p0 will result in return value NaN, with a warning.

The length of the result is determined by n for rzmgeom, and is the maximum of the lengths of the numerical arguments for the other functions.

Note

Functions {d,p,q}zmgeom use {d,p,q}geom for all but the trivial input values and $p(0)$.

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also
dgeom for the geometric distribution.
dztgeom for the zero-truncated geometric distribution.
dzmnb for the zero-modified negative binomial, of which the zero-modified geometric is a special case.

Examples

```r
p <- 1/(1 + 0.5)
dzmgeom(1:5, prob = p, p0 = 0.6)
(1-0.6) * dgeom(1:5, p)/pgeom(0, p, lower = FALSE) # same

## simple relation between survival functions
pzmgeom(0:5, p, p0 = 0.2, lower = FALSE)
(1-0.2) * pgeom(0:5, p, lower = FALSE)/pgeom(0, p, lower = FALSE) # same

qzmgeom(pzmgeom(0:10, 0.3, p0 = 0.6), 0.3, p0 = 0.6)
```
The Zero-Modified Logarithmic Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Modified Logarithmic (or log-series) distribution with parameter prob and arbitrary probability at zero p0.

Usage

dzmlogarithmic(x, prob, p0, log = FALSE)
pzmlogarithmic(q, prob, p0, lower.tail = TRUE, log.p = FALSE)
qzmlogarithmic(p, prob, p0, lower.tail = TRUE, log.p = FALSE)
rzmlogarithmic(n, prob, p0)

Arguments

x vector of (strictly positive integer) quantiles.
q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
prob parameter. 0 <= prob < 1.
p0 probability mass at zero. 0 <= p0 <= 1.
log, log.p logical; if TRUE, probabilities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].

Details

The zero-modified logarithmic distribution with prob = p and p0 = p0 is a discrete mixture between a degenerate distribution at zero and a (standard) logarithmic. The probability mass function is

\[ p(0) = p_0 \quad \text{and} \quad p(x) = (1 - p_0) f(x) \]

for \( x = 1, 2, \ldots, 0 < p < 1 \) and \( 0 \leq p_0 \leq 1 \), where \( f(x) \) is the probability mass function of the logarithmic. The cumulative distribution function is

\[ P(x) = p_0 + (1 - p_0) F(x) \]

The special case \( p_0 = 0 \) is the standard logarithmic.

The zero-modified logarithmic distribution is the limiting case of the zero-modified negative binomial distribution with size parameter equal to 0. Note that in this context, parameter prob generally corresponds to the probability of failure of the zero-truncated negative binomial.
If an element of \( x \) is not integer, the result of \( \text{dzmlogarithmic} \) is zero, with a warning.

The quantile is defined as the smallest value \( x \) such that \( F(x) \geq p \), where \( F \) is the distribution function.

**Value**

\( \text{dzmlogarithmic} \) gives the probability mass function, \( \text{pzmlogarithmic} \) gives the distribution function, \( \text{qzmlogarithmic} \) gives the quantile function, and \( \text{rzmlogarithmic} \) generates random deviates.

Invalid prob or \( p0 \) will result in return value NaN, with a warning.

The length of the result is determined by \( n \) for \( \text{rzmlogarithmic} \), and is the maximum of the lengths of the numerical arguments for the other functions.

**Note**

Functions \( \{d,p,q\}zmlogarithmic \) use \( \{d,p,q\}logarithmic \) for all but the trivial input values and \( p(0) \).

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**References**


**See Also**

\( \text{dlogarithmic} \) for the logarithmic distribution.

\( \text{dztnbinom} \) for the zero modified negative binomial distribution.

**Examples**

\[
p \leftarrow \frac{1}{1 + 0.5} \\
\text{dzmlogarithmic}(1:5, \text{prob} = p, p0 = 0.6) \\
(1-0.6) \times \text{dlogarithmic}(1:5, p) / \text{plogarithmic}(0, p, \text{lower} = \text{FALSE}) \text{ # same}
\]

## simple relation between survival functions

\[
\text{pzmlogarithmic}(0:5, p, p0 = 0.2, \text{lower} = \text{FALSE}) \\
(1-0.2) \times \text{plogarithmic}(0:5, p, \text{lower} = \text{FALSE}) / \text{plogarithmic}(0, p, \text{lower} = \text{FALSE}) \text{ # same}
\]

\[
\text{qzmlogarithmic}(\text{pzmlogarithmic}(0:10, 0.3, p0 = 0.6), 0.3, p0 = 0.6)
\]
The Zero-Modified Negative Binomial Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Modified Negative Binomial distribution with parameters size and prob, and arbitrary probability at zero p0.

Usage

\[
\begin{align*}
dzmnb(x, \text{size}, \text{prob}, p0, \text{log} = \text{FALSE}) \\
pzmnb(q, \text{size}, \text{prob}, p0, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
qzmnb(p, \text{size}, \text{prob}, p0, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
rzmnb(n, \text{size}, \text{prob}, p0)
\end{align*}
\]

Arguments

- \(x\): vector of (strictly positive integer) quantiles.
- \(q\): vector of quantiles.
- \(p\): vector of probabilities.
- \(n\): number of observations. If \(\text{length}(n) > 1\), the length is taken to be the number required.
- \(\text{size}\): target for number of successful trials, or dispersion parameter. Must be positive, need not be integer.
- \(\text{prob}\): parameter. \(0 < \text{prob} \leq 1\).
- \(p0\): probability mass at zero. \(0 \leq p0 \leq 1\).
- \(\text{log, log.p}\): logical; if \(\text{TRUE}\), probabilities \(p\) are returned as \(\log(p)\).
- \(\text{lower.tail}\): logical; if \(\text{TRUE}\) (default), probabilities are \(P[X \leq x]\), otherwise, \(P[X > x]\).

Details

The zero-modified negative binomial distribution with \(\text{size} = r\), \(\text{prob} = p\) and \(p0 = p0\) is a discrete mixture between a degenerate distribution at zero and a (standard) negative binomial. The probability mass function is \(p(0) = p0\) and

\[
p(x) = \frac{(1 - p0)}{(1 - p^r)} f(x)
\]

for \(x = 1, 2, \ldots, r \geq 0, 0 < p < 1 \) and \(0 \leq p0 \leq 1\), where \(f(x)\) is the probability mass function of the negative binomial. The cumulative distribution function is

\[
P(x) = p0 + (1 - p0) \left( \frac{F(x) - F(0)}{1 - F(0)} \right)
\]
The mean is \((1 - p_0) \mu\) and the variance is \((1 - p_0) \sigma^2 + p_0(1 - p_0) \mu^2\), where \(\mu\) and \(\sigma^2\) are the mean and variance of the zero-truncated negative binomial.

In the terminology of Klugman et al. (2012), the zero-modified negative binomial is a member of the \((a, b, 1)\) class of distributions with \(a = 1 - p\) and \(b = (r - 1)(1 - p)\).

The special case \(p_0 = 0\) is the zero-truncated negative binomial.

The limiting case \(\text{size} = 0\) is the zero-modified logarithmic distribution with parameters \(1 - \text{prob}\) and \(p_0\).

Unlike the standard negative binomial functions, parametrization through the mean \(\mu\) is not supported to avoid ambiguity as to whether \(\mu\) is the mean of the underlying negative binomial or the mean of the zero-modified distribution.

If an element of \(x\) is not integer, the result of \(\text{dzmnb}\) is zero, with a warning.

The quantile is defined as the smallest value \(x\) such that \(P(x) \geq p\), where \(P\) is the distribution function.

Value

\(\text{dzmnb}\) gives the (log) probability mass function, \(\text{pzmnb}\) gives the (log) distribution function, \(\text{qzmnb}\) gives the quantile function, and \(\text{rzmnb}\) generates random deviates.

Invalid \text{size}, \text{prob} or \(p_0\) will result in return value NaN, with a warning.

The length of the result is determined by \(n\) for \(\text{rzmnb}\), and is the maximum of the lengths of the numerical arguments for the other functions.

Note

Functions \{d,p,q\}zmnb use \{d,p,q\}nbinom for all but the trivial input values and \(p(0)\).

Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

\text{dnbinom} for the negative binomial distribution.
\text{dztbnbinom} for the zero-truncated negative binomial distribution.
\text{dzmgeom} for the zero-modified geometric and \text{dzmlogarithmic} for the zero-modified logarithmic, which are special cases of the zero-modified negative binomial.
## Example 6.3 of Klugman et al. (2012)

\[
p \left( 1 \left( 1 + 0.5 \right) \right) \\
\text{dzmbinom}(1:5, \text{size} = 2.5, \text{prob} = p, p0 = 0.6) \\
(1-0.6) * \text{dnbinom}(1:5, 2.5, p)/\text{pnbinom}(0, 2.5, p, \text{lower} = \text{FALSE}) \ # \text{same} \\
\]

## simple relation between survival functions

\[
p\text{zmbinom}(0:5, 2.5, p, p0 = 0.2, \text{lower} = \text{FALSE}) \\
(1-0.2) * \text{pnbinom}(0:5, 2.5, p, \text{lower} = \text{FALSE}) / \\
\text{pnbinom}(0, 2.5, p, \text{lower} = \text{FALSE}) \ # \text{same} \\
\text{qzmbinom(pzmbinom(0:10, 2.5, 0.3, p0 = 0.1), 2.5, 0.3, p0 = 0.1)} \\
\]

---

### ZeroModifiedPoisson

**The Zero-Modified Poisson Distribution**

**Description**

Density function, distribution function, quantile function, random generation for the Zero-Modified Poisson distribution with parameter \(\lambda\) and arbitrary probability at zero \(p0\).

**Usage**

\[
dzm\text{pois}(x, \lambda, p0, \text{log} = \text{FALSE}) \\
pzm\text{pois}(q, \lambda, p0, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
qzm\text{pois}(p, \lambda, p0, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}) \\
rzm\text{pois}(n, \lambda, p0) \\
\]

**Arguments**

- \(x\) vector of (strictly positive integer) quantiles.
- \(q\) vector of quantiles.
- \(p\) vector of probabilities.
- \(n\) number of values to return.
- \(\lambda\) vector of (non negative) means.
- \(p0\) probability mass at zero. \(0 <= p0 <= 1\).
- \(\text{log, log.p}\) logical; if \(\text{TRUE}\), probabilities \(p\) are returned as \(\log(p)\).
- \(\text{lower.tail}\) logical; if \(\text{TRUE}\) (default), probabilities are \(P[X \leq x]\), otherwise, \(P[X > x]\).

**Details**

The zero-modified Poisson distribution is a discrete mixture between a degenerate distribution at zero and a (standard) Poisson. The probability mass function is \(p(0) = p0\) and

\[
p(x) = \frac{(1 - p0)}{(1 - e^{-\lambda} f(x)}
\]
for \( x = 1, 2, \ldots, \lambda > 0 \) and \( 0 \leq p_0 \leq 1 \), where \( f(x) \) is the probability mass function of the Poisson. The cumulative distribution function is

\[
P(x) = p_0 + (1 - p_0) \left( \frac{F(x) - F(0)}{1 - F(0)} \right).
\]

The mean is \((1 - p_0) \mu\) and the variance is \((1 - p_0) \sigma^2 + p_0 (1 - p_0) \mu^2\), where \(\mu\) and \(\sigma^2\) are the mean and variance of the zero-truncated Poisson.

In the terminology of Klugman et al. (2012), the zero-modified Poisson is a member of the \((a, b, 1)\) class of distributions with \(a = 0\) and \(b = \lambda\).

The special case \(p_0 = 0\) is the zero-truncated Poisson.

If an element of \(x\) is not integer, the result of \texttt{dzmpois} is zero, with a warning.

The quantile is defined as the smallest value \(x\) such that \(P(x) \geq p\), where \(P\) is the distribution function.

**Value**

\texttt{dzmpois} gives the (log) probability mass function, \texttt{pzmpois} gives the (log) distribution function, \texttt{qzmpois} gives the quantile function, and \texttt{rzmpois} generates random deviates.

Invalid lambda or \(p\) will result in return value NaN, with a warning.

The length of the result is determined by \(n\) for \texttt{rzmpois}, and is the maximum of the lengths of the numerical arguments for the other functions.

**Note**

Functions \{d,p,q\}zmpois use \{d,p,q\}pois for all but the trivial input values and \(p(0)\).

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**References**


**See Also**

\texttt{dpois} for the standard Poisson distribution.
\texttt{dztpois} for the zero-truncated Poisson distribution.

**Examples**

\[
\texttt{dzmpois(0:5, lambda = 1, p0 = 0.2)}
\]

\[
(1-0.2) \times \texttt{dpois(0:5, lambda = 1)/ppois(0, 1, lower = FALSE)} \ # \text{same}
\]

\[
\# \text{simple relation between survival functions}
\texttt{pzmpois(0:5, 1, p0 = 0.2, lower = FALSE)}
\]
ZeroTruncatedBinomial

The Zero-Truncated Binomial Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Truncated Binomial distribution with parameters size and prob.

Usage

dztbinom(x, size, prob, log = FALSE)
pztbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qztbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rztbinom(n, size, prob)

Arguments

x vector of (strictly positive integer) quantiles.
q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
size number of trials (strictly positive integer).
prob probability of success on each trial. 0 <= prob <= 1.
log, log.p logical; if TRUE, probabilities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].

Details

The zero-truncated binomial distribution with size = n and prob = p has probability mass function

\[ p(x) = \binom{n}{x} p^x (1 - p)^{n-x} / 1 - (1 - p)^n \]

for \( x = 1, \ldots, n \) and \( 0 < p \leq 1 \), and \( p(1) = 1 \) when \( p = 0 \). The cumulative distribution function is

\[ P(x) = \frac{F(x) - F(0)}{1 - F(0)}, \]

where \( F(x) \) is the distribution function of the standard binomial.

The mean is \( np / (1 - (1 - p)^n) \) and the variance is \( np[(1 - p) - (1 - p + np)(1 - p)^n] / [1 - (1 - p)^n]^2 \).
In the terminology of Klugman et al. (2012), the zero-truncated binomial is a member of the \((a, b, 1)\) class of distributions with \(a = -p/(1 - p)\) and \(b = (n + 1)p/(1 - p)\).

If an element of \(x\) is not integer, the result of \(\text{dztbinom}\) is zero, with a warning.

The quantile is defined as the smallest value \(x\) such that \(P(x) \geq p\), where \(P\) is the distribution function.

**Value**

\(\text{dztbinom}\) gives the probability mass function, \(\text{pztbinom}\) gives the distribution function, \(\text{qztbinom}\) gives the quantile function, and \(\text{rztbinom}\) generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by \(n\) for \(\text{rztbinom}\), and is the maximum of the lengths of the numerical arguments for the other functions.

**Note**

Functions \(\{d, p, q\}\text{ztbinom}\) use \(\{d, p, q\}\text{binom}\) for all but the trivial input values and \(p(0)\).

\(\text{rztbinom}\) uses the simple inversion algorithm suggested by Peter Dalgaard on the r-help mailing list on 1 May 2005 (https://stat.ethz.ch/pipermail/r-help/2005-May/070680.html).

**Author(s)**

Vincent Goulet <vincent.goulet@act.ulaval.ca>

**References**


**See Also**

dbinom for the binomial distribution.

**Examples**

dztbinom(1:5, size = 5, prob = 0.4)
dbinom(1:5, 5, 0.4)/pbion(0, 5, 0.4, lower = FALSE) # same

pztbinom(1, 2, prob = 0) # point mass at 1

qztbinom(pztbinom(1:10, 10, 0.6), 10, 0.6)

n <- 8; p <- 0.3
x <- 0:n
title <- paste("ZT Binomial(", n, ", ", p,
") and Binomial("n", ", ", p," PDF",
sep = "")
plot(x, dztbinom(x, n, p), type = "h", lwd = 2, ylab = "p(x)",
main = title)
The Zero-Truncated Geometric Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Truncated Geometric distribution with parameter prob.

Usage

dztgeom(x, prob, log = FALSE)
pztgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
qztgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
rztgeom(n, prob)

Arguments

x vector of (strictly positive integer) quantiles.
q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
prob parameter. 0 < prob <= 1.
log, log.p logical; if TRUE, probabilities p are returned as \log(p).
lower.tail logical; if TRUE (default), probabilities are \Pr[X \leq x], otherwise, \Pr[X > x].

Details

The zero-truncated geometric distribution with prob = p has probability mass function

\[ p(x) = p(1 - p)^{x-1} \]

for \( x = 1, 2, \ldots \) and 0 < p < 1, and \( p(1) = 1 \) when \( p = 1 \). The cumulative distribution function is

\[ P(x) = \frac{F(x) - F(0)}{1 - F(0)}, \]

where \( F(x) \) is the distribution function of the standard geometric.

The mean is 1/p and the variance is \((1 - p)/p^2\).

In the terminology of Klugman et al. (2012), the zero-truncated geometric is a member of the \((a, b, 1)\) class of distributions with \( a = 1 - p \) and \( b = 0 \).

If an element of x is not integer, the result of dztgeom is zero, with a warning.

The quantile is defined as the smallest value \( x \) such that \( P(x) \geq p \), where \( P \) is the distribution function.
ZeroTruncatedGeometric

Value

dztgeom gives the (log) probability mass function, pztgeom gives the (log) distribution function, qztgeom gives the quantile function, and rztgeom generates random deviates.

Invalid prob will result in return value NaN, with a warning.

The length of the result is determined by n for rztgeom, and is the maximum of the lengths of the numerical arguments for the other functions.

Note

Functions \{d,p,q\}ztgeom use \{d,p,q\}geom for all but the trivial input values and \( p(0) \).


Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

dgeom for the geometric distribution.

dzttnbinom for the zero-truncated negative binomial, of which the zero-truncated geometric is a special case.

Examples

```r
p <- 1/(1 + 0.5)
dztgeom(c(1, 2, 3), prob = p)
dgeom(c(1, 2, 3), p)/pgeom(0, p, lower = FALSE) # same
dgeom(c(1, 2, 3) - 1, p) # same

pztgeom(1, prob = 1) # point mass at 1

qztgeom(pztgeom(1:10, 0.3), 0.3)
```
ZeroTruncatedNegativeBinomial

The Zero-Truncated Negative Binomial Distribution

Description

Density function, distribution function, quantile function and random generation for the Zero-Truncated Negative Binomial distribution with parameters size and prob.

Usage

dztnbinom(x, size, prob, log = FALSE)
pztnbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qztnbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rztnbinom(n, size, prob)

Arguments

x vector of (strictly positive integer) quantiles.
q vector of quantiles.
p vector of probabilities.
n number of observations. If length(n) > 1, the length is taken to be the number required.
size target for number of successful trials, or dispersion parameter. Must be positive, need not be integer.
prob parameter. 0 < prob <= 1.
log, log.p logical; if TRUE, probabilities p are returned as log(p).
lower.tail logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x].

Details

The zero-truncated negative binomial distribution with size = r and prob = p has probability mass function

\[ p(x) = \frac{\Gamma(x + r) p^r (1-p)^x}{\Gamma(r)x! (1-p^r)} \]

for x = 1, 2, ..., r ≥ 0 and 0 < p < 1, and p(1) = 1 when p = 1. The cumulative distribution function is

\[ P(x) = \frac{F(x) - F(0)}{1 - F(0)}, \]

where F(x) is the distribution function of the standard negative binomial.

The mean is r(1 - p)/(p(1 − p^r)) and the variance is [r(1 − p)(1 + r(1 − p)p^r)]/[p(1 − p^r)]^2. 

In the terminology of Klugman et al. (2012), the zero-truncated negative binomial is a member of the (a, b, 1) class of distributions with a = 1 − p and b = (r − 1)(1 − p).
The limiting case size == 0 is the logarithmic distribution with parameter 1 - prob.

Unlike the standard negative binomial functions, parametrization through the mean mu is not supported to avoid ambiguity as to whether mu is the mean of the underlying negative binomial or the mean of the zero-truncated distribution.

If an element of x is not integer, the result of dztnbinom is zero, with a warning.

The quantile is defined as the smallest value x such that P(x) ≥ p, where P is the distribution function.

Value

dztnbinom gives the (log) probability mass function, pztnbinom gives the (log) distribution function, qztnbinom gives the quantile function, and rztnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rztnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

Note

Functions {d,p,q}ztnbinom use {d,p,q}nbinom for all but the trivial input values and p(0).


Author(s)

Vincent Goulet <vincent.goulet@act.ulaval.ca>

References


See Also

dnbinom for the negative binomial distribution.
dztgeom for the zero-truncated geometric and dlogarithmic for the logarithmic, which are special cases of the zero-truncated negative binomial.

Examples

```r
# Example 6.3 of Klugman et al. (2012)
p <- 1/(1 + 0.5)
dztnbinom(c(1, 2, 3), size = 2.5, prob = p)
dnbinom(c(1, 2, 3), 2.5, p)/pnbinom(0, 2.5, p, lower = FALSE) # same

pztnbinom(1, 2, prob = 1)  # point mass at 1
dztnbinom(2, size = 1, 0.25)  # == dztgeom(2, 0.25)
dztnbinom(2, size = 0, 0.25)  # == dlogarithmic(2, 0.75)
```
```
zqtnbinom(pztnbinom(1:10, 2.5, 0.3), 2.5, 0.3)
x <- rztnbinom(1000, size = 2.5, prob = 0.4)
y <- sort(unique(x))
plot(y, table(x)/length(x), type = "h", lwd = 2,
    pch = 19, col = "black", xlab = "x", ylab = "p(x)",
    main = "Empirical vs theoretical probabilities")
points(y, dztnbinom(y, size = 2.5, prob = 0.4),
    pch = 19, col = "red")
legend("topright", c("empirical", "theoretical"),
    lty = c(1, NA), lwd = 2, pch = c(NA, 19), col = c("black", "red"))
```

---

### ZeroTruncatedPoisson

**The Zero-Truncated Poisson Distribution**

**Description**

Density function, distribution function, quantile function, random generation for the Zero-Truncated Poisson distribution with parameter \( \lambda \).

**Usage**

- `dztpois(x, lambda, log = FALSE)`
- `pztpois(q, lambda, lower.tail = TRUE, log.p = FALSE)`
- `qztpois(p, lambda, lower.tail = TRUE, log.p = FALSE)`
- `rztpois(n, lambda)`

**Arguments**

- `x`: vector of (strictly positive integer) quantiles.
- `q`: vector of quantiles.
- `p`: vector of probabilities.
- `n`: number of values to return.
- `lambda`: vector of (non-negative) means.
- `log`, `log.p`: logical; if `TRUE`, probabilities \( p \) are returned as `log(p)`.  
- `lower.tail`: logical; if `TRUE` (default), probabilities are \( P[X \leq x] \), otherwise, \( P[X > x] \).

**Details**

The zero-truncated Poisson distribution has probability mass function

\[
p(x) = \frac{e^{-/lambda} \lambda^x}{x!(1 - e^{-/lambda})} = \frac{\lambda^x}{x!(e^\lambda - 1)}
\]

for \( x = 1, 2, \ldots \), and \( p(1) = 1 \) when \( \lambda = 0 \). The cumulative distribution function is

\[
P(x) = \frac{F(x) - F(0)}{1 - F(0)}
\]
where $F(x)$ is the distribution function of the standard Poisson.
The mean is $\lambda/(1 - e^{-\lambda})^2$ and the variance is $\lambda [1 - (\lambda + 1)e^{-\lambda}] / (1 - e^{-\lambda})^2$.
In the terminology of Klugman et al. (2012), the zero-truncated Poisson is a member of the $(a, b, 1)$
class of distributions with $a = 0$ and $b = \lambda$.
If an element of $x$ is not integer, the result of dztpois is zero, with a warning.
The quantile is defined as the smallest value $x$ such that $P(x) \geq p$, where $P$ is the distribution
function.

Value
dztpois gives the (log) probability mass function, pztpois gives the (log) distribution function,
qztpois gives the quantile function, and rztpois generates random deviates.
Invalid lambda will result in return value NaN, with a warning.
The length of the result is determined by n for rztpois, and is the maximum of the lengths of the
numerical arguments for the other functions.

Note
Functions {d,p,q}ztpois use {d,p,q}pois for all but the trivial input values and $p(0)$.
rztpois uses the simple inversion algorithm suggested by Peter Dalgaard on the r-help mailing list

Author(s)
Vincent Goulet <vincent.goulet@act.ulaval.ca>

References
Klugman, S. A., Panjer, H. H. and Willmot, G. E. (2012), Loss Models, From Data to Decisions,

See Also
dpois for the standard Poisson distribution.

Examples
dztpois(1:5, lambda = 1)
dpois(1:5, lambda = 1)/ppois(0, 1, lower = FALSE) # same
pztpois(1, lambda = 0) # point mass at 1
qztpois(pztpois(1:10, 1), 1)
x <- seq(0, 8)
plot(x, dztpois(x, 2), type = "h", lwd = 2, ylab = "p(x)",
     main = "Zero-Truncated Poisson(2) and Poisson(2) PDF")
points(x, dpois(x, 2), pch = 19, col = "red")
legend("topright", c("ZT Poisson probabilities", "Poisson probabilities"),
     col = "red")
col = c("black", "red"), lty = c(1, 0), lwd = 2, pch = c(NA, 19))
Index

* array
  Extract.grouped.data, 36
* classes
  grouped.data, 46
* datagen
  rcomphierarc, 96
  rcompound, 101
  rmixture, 103
  severity, 107
* datasets
  dental, 29
  gdental, 41
  hachemeister, 50
* distribution
  actuar-package, 4
  aggregateDist, 8
  betaint, 13
  BetaMoments, 15
  Burr, 16
  ChisqSupp, 18
  discretize, 29
  ExponentialSupp, 35
  FellerPareto, 37
  GammaSupp, 40
  GeneralizedBeta, 41
  GeneralizedPareto, 43
  Gumbel, 48
  hist.grouped.data, 50
  InverseBurr, 52
  InverseExponential, 54
  InverseGamma, 55
  InverseGaussian, 57
  InverseParalogistic, 60
  InversePareto, 61
  InverseTransformedGamma, 63
  InverseWeibull, 65
  Logarithmic, 67
  Loggamma, 69
  Loglogistic, 70
  LognormalMoments, 72
  mde, 73
  NormalSupp, 76
  Paralogistic, 79
  Pareto, 81
  Pareto2, 83
  Pareto3, 85
  Pareto4, 87
  PhaseType, 89
  PoissonInverseGaussian, 91
  SingleParameterPareto, 108
  TransformedBeta, 110
  TransformedGamma, 112
  UniformSupp, 114
  WeibullMoments, 118
  ZeroModifiedBinomial, 120
  ZeroModifiedGeometric, 122
  ZeroModifiedLogarithmic, 124
  ZeroModifiedNegativeBinomial, 126
  ZeroModifiedPoisson, 128
  ZeroTruncatedBinomial, 130
  ZeroTruncatedGeometric, 132
  ZeroTruncatedNegativeBinomial, 134
  ZeroTruncatedPoisson, 136
* dplot
  elev, 32
  hist.grouped.data, 50
  ogive, 77
* hplot
  elev, 32
  hist.grouped.data, 50
  ogive, 77
* htest
  mde, 73
* manip
  Extract.grouped.data, 36
  severity, 107
  unroll, 115
* math
betaint, 13

* methods
  grouped.data, 46
  rcomphierarc.summaries, 99

* models
  actuar-package, 4
  aggregateDist, 8
  cm, 19
  coverage, 25
  discretize, 29
  rcomphierarc.summaries, 99
  ruin, 105

* optimize
  adjCoef, 5

* package
  actuar-package, 4

* univar
  actuar-package, 4
  adjCoef, 5
  CTE, 27
  emm, 33
  mean.grouped.data, 75
  quantile.aggregateDist, 94
  quantile.grouped.data, 95
  VaR, 116
  var, 117

[.data.frame, 37
[.grouped.data, 47
[.grouped.data (Extract.grouped.data), 36
[<-.grouped.data
  (Extract.grouped.data), 36

actuar (actuar-package), 4
actuar-package, 4
adjCoef, 5, 5
aggregate.portfolio
  (rcomphierarc.summaries), 99
aggregateDist, 5, 8, 28, 29, 31, 94, 117
approxfun, 78
as.data.frame, 20
as.integer, 36
axis, 51
besselK, 92
Beta, 16
betaint, 13, 17, 39, 45, 53, 61, 72, 80, 82, 84, 86, 89, 111
BetaMoments, 15

Burr, 16, 88, 111
ChisqSupp, 18
Chisquare, 19
class, 97, 99, 107
cm, 5, 19
colMeans, 34
Coverage (coverage), 25
coverage, 5, 25
CTE, 27
CTE.aggregateDist, 11
curve, 30, 105
data.frame, 46, 47
dbeta, 24
dbinom, 10, 24, 121, 131
dburr, 89
dburr (Burr), 16
dental, 29
dexp, 106
dfpareto, 112
dfpareto (FellerPareto), 37
dgamma, 24, 106
dgenbeta (GeneralizedBeta), 41
dgenpareto (GeneralizedPareto), 43
dgeom, 10, 123, 133
dgumbel (Gumbel), 48
diff.aggregateDist (aggregateDist), 8
dim, 115
dinvburr (InverseBurr), 52
dinvexp (InverseExponential), 54
dinvgamma, 59
dinvgamma (InverseGamma), 55
dinvgauss, 92, 93
dinvgauss (InverseGaussian), 57
dinvparalogis (InverseParalogistic), 60
dinvpareto (InversePareto), 61
dinvtrgamma (InverseTransformedGamma), 63
dinvweibull (InverseWeibull), 65
discretise (discretize), 29
discretize, 5, 11, 29
dlgamma (Loggamma), 69
dlgompertz (InverseWeibull), 65
dllogis, 87
dllogis (Loglogistic), 70
dllogarithmic, 10, 125, 135
dllogarithmic (Loglogistic), 67
dnbinom, 10, 24, 127, 135
INDEX

Index entries:

dnorm, 24
dparalogis (Paralogistic), 79
dpareto, 85, 109
dpareto (Pareto), 81
dpareto1, 82
dpareto1 (SingleParameterPareto), 108
dpareto2, 82
dpareto2 (Pareto2), 83
dpareto3, 72
dpareto3 (Pareto3), 85
dpareto4, 18
dpareto4 (Pareto4), 87
dpearson6 (TransformedBeta), 110
dptype, 106
dptype (PhaseType), 89
dpig (PoissonInverseGaussian), 91
dpois, 10, 93, 129, 137
dpoisinvgauss (PoissonInverseGaussian), 91
dtrbeta, 39
dtrbeta (TransformedBeta), 110
dtrgamma (TransformedGamma), 112
dzmbinom (ZeroModifiedBinomial), 120
dzmgeomo (ZeroModifiedGeometric), 122
dzmlogarithmic, 127
dzmlogarithmic
   (ZeroModifiedLogarithmic), 124
dzmnbinn (123
dzmnbinn
   (ZeroModifiedNegativeBinomial), 126
dzmpois (ZeroModifiedPoisson), 128
dztinbinom, 121
dztbinom (ZeroTruncatedBinomial), 130
dztgeom, 123, 135
dztgeom (ZeroTruncatedGeometric), 132
dztnbinom, 68, 125, 127, 133
dztnbinom
   (ZeroTruncatedNegativeBinomial), 134
dztpois, 129
dztpois (ZeroTruncatedPoisson), 136
elev, 5, 32
eomm, 5, 33, 76, 118
Exponential, 35, 113
ExponentialSupp, 35
expression, 97, 102, 103
Extract.grouped.data, 36
Feller-Pareto, 84, 86, 88
FellerPareto, 5, 37
formals, 26
format, 20, 95
formula, 20, 24
classify.portfolio
   (rcomphierarc.summaries), 99
function, 33, 78
Gamma, 113
gamma, 13, 38, 42, 44, 56, 64, 69, 111, 113
GammaDist, 41
GammaSupp, 40
gdental, 41
generalized Pareto, 111
GeneralizedBeta, 41
GeneralizedPareto, 43
grouped.data, 5, 33, 37, 41, 46, 76, 78, 96, 118
Gumbel, 5, 48
hachemeister, 5, 50
hist, 46, 47, 50, 52
hist.default, 50, 52
hist.grouped.data, 50
inverse Burr, 111
Inverse Exponential, 56, 64, 66
Inverse Gamma, 64
Inverse Paralogistic, 53
inverse paralogistic, 111
Inverse Pareto, 44, 53
inverse Pareto, 111
Inverse Weibull, 64
InverseBurr, 52
InverseExponential, 54
InverseGamma, 55
InverseGaussian, 5, 57
InverseParalogistic, 60
InversePareto, 61
InverseTransformedGamma, 63
InverseWeibull, 65
knots.elev (elev), 32
knots.ogive (ogive), 77
levbeta (BetaMoments), 15
levburr (Burr), 16
levchisq (ChisqSupp), 18
levexp (ExponentialSupp), 35
levfpareto (FellerPareto), 37
levgamma (GammaSupp), 40
levgenbeta (GeneralizedBeta), 41
levgenpareto (GeneralizedPareto), 43
levinvburr (InverseBurr), 52
levinvexp (InverseExponential), 54
levinvgamma (InverseGamma), 55
levinvgauss (InverseGaussian), 57
levinvparalogis (InverseParalogistic), 60
levinvpareto (InversePareto), 61
levinvtgamma (InverseTransformedGamma), 63
levinweibull (InverseWeibull), 65
levlgamma (LogGamma), 69
levlgompertz (InverseWeibull), 65
levllogis (Loglogistic), 70
levlnorm (LogNormalMoments), 72
levparalogis (Paralogistic), 79
levpareto (Pareto), 81
levpareto1 (SingleParameterPareto), 108
levpareto2 (Pareto2), 83
levpareto3 (Pareto3), 85
levpareto4 (Pareto4), 87
levpearson6 (TransformedBeta), 110
levtrbeta (TransformedBeta), 110
levtrgamma (TransformedGamma), 112
levunif (UniformSupp), 114
levweibull (WeibullMoments), 118
lines, 6
lm, 20, 24
lm.fit, 21
lm.wfit, 21
log-series (Logarithmic), 67
Logarithmic, 67
logarithmic, 135
Loggamma, 5, 69
Loglogistic, 17, 53, 70
Loglogistic, 86, 111
Lognormal, 73
LognormalMoments, 72
mbeta (BetaMoments), 15
mburr (Burr), 16
mchisq (ChisqSupp), 18
Mde (mde), 73
mde, 5, 73
mean, 34
mean.aggregateDist, 11
mean.aggregateDist (aggregateDist), 8
mean.grouped.data, 34, 75, 96, 118
mexp (ExponentialSupp), 35
mfpbeta (FellerPareto), 37
mgamma (GammaSupp), 40
mgenbeta (GeneralizedBeta), 41
mgenpareto (GeneralizedPareto), 43
mgfchisq (ChisqSupp), 18
mgfexp (ExponentialSupp), 35
mgffgamma (GammaSupp), 40
mgfgumbel (Gumbel), 48
mgfngamma (InverseGamma), 55
mgfngauss (InverseGaussian), 57
mgfnorm (NormalSupp), 76
mgfphype (PhaseType), 89
mgfunif (UniformSupp), 114
mgfweibull (InverseBurr), 52
mivexp (InverseExponential), 54
mivgamma (InverseGamma), 55
mivgauss (InverseGaussian), 57
mivparalogis (InverseParalogistic), 60
mivpareto (InversePareto), 61
mivtrgamma (InverseTransformedGamma), 63
mivweibull (InverseWeibull), 65
mlgamma (LogGamma), 69
mlgompertz (InverseWeibull), 65
mllogis (Loglogistic), 70
mlnorm (LogNormalMoments), 72
mnorm (NormalSupp), 76
mparalogis (Paralogistic), 79
mpbeta (Pareto), 81
mpbeta1 (SingleParameterPareto), 108
mpbeta2 (Pareto2), 83
mpbeta3 (Pareto3), 85
mpbeta4 (Pareto4), 87
mpearson6 (TransformedBeta), 110
mphtype (PhaseType), 89
mtrbeta (TransformedBeta), 110
mtrgamma (TransformedGamma), 112
munif (UniformSupp), 114
mweibull (WeibullMoments), 118
Normal, 77
NormalSupp, 76
INDEX

ogive, 5, 77, 96
optim, 74

Paralogistic, 17, 79
paralogistic, 111
Pareto, 17, 44, 45, 81, 84, 111
Pareto I, 38, 84
Pareto II, 38, 82, 88
Pareto III, 38, 88
Pareto IV, 38
Pareto2, 83
Pareto3, 85
Pareto4, 87
pburr (Burr), 16
Pearson6 (TransformedBeta), 110
pfpareto (FellerPareto), 37
pgenbeta (GeneralizedBeta), 41
pgenpareto (GeneralizedPareto), 43
pgumbel (Gumbel), 48
PhaseType, 5, 89
PIG (PoissonInverseGaussian), 91
pinvburr (InverseBurr), 52
pinvexp (InverseExponential), 54
pinvgamma (InverseGamma), 55
pinvgauss (InverseGaussian), 57
pinvparalogis (InverseParalogistic), 60
pinvpareto (InversePareto), 61
pinvtgamma (InverseTransformedGamma), 63
pinvweibull (InverseWeibull), 65
plgamma (Loggamma), 69
plgompertz (InverseWeibull), 65
pllogis (Loglogistic), 70
plot, 6
plot.adjCoef (adjCoef), 5
plot.aggregateDist (aggregateDist), 8
plot.elev (elev), 32
plot.histogram, 51
plot.obl (ogive), 77
plot.ruin (ruin), 105
PoissonInverseGaussian, 5, 91
pparalogis (Paralogistic), 79
ppareto (Pareto), 81
ppareto1 (SingleParameterPareto), 108
ppareto2 (Pareto2), 83
ppareto3 (Pareto3), 85
ppareto4 (Pareto4), 87
pparson6 (TransformedBeta), 110

pphype, 106
pphype (PhaseType), 89
ppig (PoissonInverseGaussian), 91
ppoisinvgauss (PoissonInverseGaussian), 91
predict.cm (cm), 19
predict.lm, 22, 24
pretty, 46
print, 32, 78, 95
print.aggregateDist (aggregateDist), 8
print.cm (cm), 19
print.elev (elev), 32
print.obl (ogive), 77
print.portfolio (rcompheierarc), 96
print.summary.cm (cm), 19
ptrbeta (TransformedBeta), 110
ptrgamma (TransformedGamma), 112
pzbinom (ZeroModifiedBinomial), 120
pzlogormal (ZeroModifiedLogarithmic), 124
pzmbinom (ZeroModifiedNegativeBinomial), 126
pzmpois (ZeroModifiedPoisson), 128
pztnbinom (ZeroTruncatedBinomial), 130
pztgeom (ZeroTruncatedGeometric), 132
pztbinom (ZeroTruncatedNegativeBinomial), 134
pztipois (ZeroTruncatedPoisson), 136
qburr (Burr), 16
qfpareto (FellerPareto), 37
qggenbeta (GeneralizedBeta), 41
qggenpareto (GeneralizedPareto), 43
qgumbel (Gumbel), 48
qinvburr (InverseBurr), 52
qinvexp (InverseExponential), 54
qinvgamma (InverseGamma), 55
qinvgauss (InverseGaussian), 57
qinvparalogis (InverseParalogistic), 60
qinvpareto (InversePareto), 61
qinvtrgamma (InverseTransformedGamma), 63
qinvweibull (InverseWeibull), 65
qlgamma (Loggamma), 69
qlgompertz (InverseWeibull), 65
qllogis (Loglogistic), 70
qlogarithmic (Logarithmic), 67
qparalogis (Paralogistic), 79
qpareto (Pareto), 81
qpareto1 (SingleParameterPareto), 108
qpareto2 (Pareto2), 83
qpareto3 (Pareto3), 85
qpareto4 (Pareto4), 87
qpearson6 (TransformedBeta), 110
qpig (PoissonInverseGaussian), 91
qpoisinvgauss (PoissonInverseGaussian), 91
qtrbeta (TransformedBeta), 110
qtrgamma (TransformedGamma), 112
quantile.aggregateDist, 11, 94
quantile.grouped.data, 78, 95
qzmbinom (ZeroModifiedBinomial), 120
qzmgeom (ZeroModifiedGeometric), 122
qzmlogarithmic
  (ZeroModifiedLogarithmic), 124
qzmnbmnom
  (ZeroModifiedNegativeBinomial), 126
qzmpois (ZeroModifiedPoisson), 128
qztbinom (ZeroTruncatedBinomial), 130
qztdgeom (ZeroTruncatedGeometric), 132
qztnbinom
  (ZeroTruncatedNegativeBinomial), 134
qztpps (ZeroTruncatedPoisson), 136
rburr (Burr), 16
rcomphierarc, 9, 11, 96, 100, 102–104
rcomphierarc.summaries, 98, 99
rcompound, 97, 98, 101, 104
rcomppois (rcompound), 101
rfpareto (FellerPareto), 37
rgenbeta (GeneralizedBeta), 41
rgenpareto (GeneralizedPareto), 43
rgumbel (Gumbel), 48
rinburr (InverseBurr), 52
rinexp (InverseExponential), 54
rinvgamma (InverseGamma), 55
rinvgauss (InverseGaussian), 57
rinvparalogis (InverseParalogistic), 60
rinvpareto (InversePareto), 61
rintrgamma (InverseTransformedGamma), 63
rinweibull (InverseWeibull), 65
rlgamma (Loggamma), 69
rlgompertz (InverseWeibull), 65
rllogis (Loglogistic), 70
rlogarithmic (Logarithmic), 67
rmixture, 103
rparalogis (Paralogistic), 79
rpareto (Pareto), 81
rpareto1 (SingleParameterPareto), 108
rpareto2 (Pareto2), 83
rpareto3 (Pareto3), 85
rpareto4 (Pareto4), 87
rpearson6 (TransformedBeta), 110
rphype (PhaseType), 89
rpoig (PoissonInverseGaussian), 91
rpoisinvgauss (PoissonInverseGaussian), 91
rrbeta (TransformedBeta), 110
rrgamma (TransformedGamma), 112
ruin, 5, 105
rzbinom (ZeroModifiedBinomial), 120
rzmgeom (ZeroModifiedGeometric), 122
rzmlogarithmic
  (ZeroModifiedLogarithmic), 124
rzmbinom
  (ZeroModifiedNegativeBinomial), 126
rzmpois (ZeroModifiedPoisson), 128
rztnbinom (ZeroTruncatedBinomial), 130
rztgeom (ZeroTruncatedGeometric), 132
rztnbinom
  (ZeroTruncatedNegativeBinomial), 134
rztpois (ZeroTruncatedPoisson), 136
sd, 117
sd(var), 117
severity, 100, 107
severity.portfolio, 108, 116
severity.portfolio
  (rcomphierarc.summaries), 99
simul, 99
simul (rcomphierarc), 96
SingleParameterPareto, 108
stats::var, 117
stepfun, 33, 78
subset, 21, 24
summary.aggregateDist (aggregateDist), 8
summary.cm (cm), 19
summary.elev (elev), 32
summary.grouped.data (quantile.grouped.data), 95
summary.ogive (ogive), 77
table, 95
terms, 21
title, 51
transformed beta, 38
TransformedBeta, 110
TransformedGamma, 5, 112
TVaR (CTE), 27

Uniform, 115
UniformSupp, 114
unroll, 108, 115

VaR, 29, 116
var, 117, 117
VaR.aggregateDist, 117
VaR.aggregateDist (quantile.aggregateDist), 94
var.grouped.data, 96

Weibull, 113, 119
WeibullMoments, 118
weights.portfolio (rcomphierarc.summaries), 99

ZeroModifiedBinomial, 120
ZeroModifiedGeometric, 122
ZeroModifiedLogarithmic, 124
ZeroModifiedNegativeBinomial, 126
ZeroModifiedPoisson, 5, 128
ZeroTruncatedBinomial, 130
ZeroTruncatedGeometric, 132
ZeroTruncatedNegativeBinomial, 134
ZeroTruncatedPoisson, 5, 136
ZMBinomial (ZeroModifiedBinomial), 120
Zmgeometric (ZeroModifiedGeometric), 122
ZMLogarithmic (ZeroModifiedLogarithmic), 124
ZMNegativeBinomial (ZeroModifiedNegativeBinomial), 126
ZMNegBinomial (ZeroModifiedNegativeBinomial), 126
ZMPoisson (ZeroModifiedPoisson), 128
ZTBinomial (ZeroTruncatedBinomial), 130
ZTGeometric (ZeroTruncatedGeometric), 132
ZTNegativeBinomial (ZeroTruncatedNegativeBinomial), 134
ZTNegBinomial (ZeroTruncatedNegativeBinomial), 134
ZTPoisson (ZeroTruncatedPoisson), 136