Package ‘distributionsrd’

October 13, 2022

Title Distribution Fitting and Evaluation

Version 0.0.6

Description A library of density, distribution function, quantile function, (bounded) raw moments and random generation for a collection of distributions relevant for the firm size literature. Additionally, the package contains tools to fit these distributions using maximum likelihood and evaluate these distributions based on (i) log-likelihood ratio and (ii) deviations between the empirical and parametrically implied moments of the distributions. We add flexibility by allowing the considered distributions to be combined into piecewise composite or finite mixture distributions, as well as to be used when truncated. See Dewitte (2020) <https://hdl.handle.net/1854/LU-8644700> for a description and application of methods available in this package.

Depends R (>= 3.6.0)

Imports Rdpack, stats, flexmix, modeltools, methods

RdMacros Rdpack

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Suggests knitr, rmarkdown, tidyverse, testthat

NeedsCompilation no

Author Ruben Dewitte [aut, cre]

Maintainer Ruben Dewitte <ruben0dewitte@gmail.com>

Repository CRAN

Date/Publication 2020-05-25 18:50:03 UTC

R topics documented:

`burr` *The Burr distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for the Burr distribution, also known as the Burr Type XII distribution or the Singh-Maddala distribution.

Usage

```
dburr(x, shape1 = 2, shape2 = 1, scale = 0.5, log = FALSE)

pburr(q, shape1 = 2, shape2 = 1, scale = 0.5, log.p = FALSE, lower.tail = TRUE)

qburr(p, shape1 = 2, shape2 = 1, scale = 0.5, log.p = FALSE, lower.tail = TRUE)

mburr(
  r = 0,
  truncation = 0,
  shape1 = 2,
  shape2 = 1,
  scale = 0.5,
  lower.tail = TRUE
)

rburr(n, shape1 = 2, shape2 = 1, scale = 0.5)
```

Arguments

x, q	vector of quantiles
shape1, shape2, scale	Shape1, shape2 and scale of the Burr distribution, defaults to 2, 1 and 0.5.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r \mid X \leq y])$, otherwise, $P[X > x]$ $(E[x^r \mid X > y])$
p	vector of probabilities
r	rth raw moment of the distribution
truncation	lower truncation parameter
n	number of observations

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{\frac{kc}{scale}(\frac{\omega}{scale})^{shape2-1}}{(1+(\frac{\omega}{scale})^shape2)^{shape1+1}}, \qquad F_X(x) = 1 - \frac{1}{(1+(\frac{\omega}{scale})^shape2)^shape1}$$

The y-bounded r-th raw moment of the Fréchet distribution equals:

$$scale^r shape1[\boldsymbol{B}(\frac{r}{shape2}+1, shape1-\frac{r}{shape2})-\boldsymbol{B}(\frac{(\frac{y}{scale})^{shape2}}{1+(\frac{y}{scale})^{shape2}}; \frac{r}{shape2}+1, shape1-\frac{r}{shape2})], \qquad shape2 > r$$

Value

dburr returns the density, pburr the distribution function, qburr the quantile function, mburr the rth moment of the distribution and rburr generates random deviates.

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

Examples

```
## Burr density
plot(x = seq(0, 5, length.out = 100), y = dburr(x = seq(0, 5, length.out = 100)))
plot(x = seq(0, 5, length.out = 100), y = dburr(x = seq(0, 5, length.out = 100), shape2 = 3))

## Demonstration of log functionality for probability and quantile function
qburr(pburr(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
pburr(2)
mburr(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a
#(truncated) random sample, for large enough samples.
x <- rburr(1e5, shape2 = 3)

mean(x)
mburr(r = 1, shape2 = 3, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mburr(r = 1, shape2 = 3, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

---

burr_plt *Burr coefficients after power-law transformation*

Description

Coefficients of a power-law transformed Burr distribution

Usage

```
burr_plt(shape1 = 2, shape2 = 1, scale = 0.5, a = 1, b = 1, inv = FALSE)
```

Arguments

shape1, shape2, scale
: Shape1, shape2 and scale of the Burr distribution, defaults to 2, 1 and 1 respectively.

a, b
: constant and power of power-law transformation, defaults to 1 and 1 respectively.

`inv`	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable x is Burr distributed with scale shape and shape scale, then the power-law transformed variable

$$y = ax^b$$

is Burr distributed with shape1 $shape1$, shape2 $b * shape2$ and scale $(\frac{scale}{a})^{\frac{1}{b}}$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables pburr(3,shape1=2,shape2=3,scale=1) coeff = burr_plt(shape1=2,shape2=3,scale=1,a=5,b=7)$coefficients pburr(5*3^7,shape1=coeff[["shape1"]],shape2=coeff[["s

pburr(5*0.9^7,shape1=2,shape2=3,scale=1) coeff = burr_plt(shape1=2,shape2=3,scale=1,a=5,b=7, inv=TRUE)$coefficients pburr(0.9,shape1=coeff[["shape1"]],shape2=coeff[["shape2"]],scale=coeff[["scale"]])

Comparing the first moments and sample means of power-law transformed variables for large enough samples x = rburr(1e5,shape1=2,shape2=3,scale=1) coeff = burr_plt(shape1=2,shape2=3,scale=1,a=2,b=0.5)$coeffici y = rburr(1e5,shape1=coeff[["shape1"]],shape2=coeff[["shape2"]],scale=coeff[["scale"]]) mean(2*x^0.5) mean(y) mburr(r=1,shape1=coeff[["shape1"]],shape2=coeff[["shape2"]],scale=coeff[["scale"]],lower.tail=FALSE)

`clauset.xmax` *Pareto scale determination à la Clauset*

Description

This method determines the optimal scale parameter of the Inverse Pareto distribution using the iterative method (Clauset et al. 2009) that minimizes the Kolmogorov-Smirnov distance.

Usage

```
clauset.xmax(x, q = 1)
```

Arguments

`x`	data vector
`q`	Percentage of data to search over (starting from the smallest values)

Value

Returns a named list containing a

coefficients Named vector of coefficients

KS Minimum Kolmogorov-Smirnov distance

n Number of observations in the Inverse Pareto tail

coeff.evo Evolution of the Inverse Pareto shape parameter over the iterations

References

Clauset A, Shalizi CR, Newman ME (2009). "Power-law distributions in empirical data." *SIAM review*, **51**(4), 661–703.

Examples

```
## Determine cuttof from compostie InvPareto-Lognormal distribution using Clauset's method
dist <- c("invpareto", "lnorm")
coeff <- c(coeff1.k = 1.5, coeff2.meanlog = 1, coeff2.sdlog = 0.5)
x <- rcomposite(1e3, dist = dist, coeff = coeff)
out <- clauset.xmax(x = x)
out$coefficients
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))$coeff1

## Speed up method by considering values above certain quantile only
dist <- c("invpareto", "lnorm")
coeff <- c(coeff1.k = 1.5, coeff2.meanlog = 1, coeff2.sdlog = 0.5)
x <- rcomposite(1e3, dist = dist, coeff = coeff)
out <- clauset.xmax(x = x, q = 0.5)
out$coefficients
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))$coeff1
```

`clauset.xmin` *Pareto scale determination à la Clauset*

Description

This method determines the optimal scale parameter of the Pareto distribution using the iterative method (Clauset et al. 2009)that minimizes the Kolmogorov-Smirnov distance.

Usage

```
clauset.xmin(x, q = 0)
```

Arguments

x	data vector
q	Percentage of data to search over (starting from the largest values)

Value

Returns a named list containing a

coefficients Named vector of coefficients

KS Minimum Kolmogorov-Smirnov distance

n Number of observations in the Pareto tail

coeff.evo Evolution of the Pareto shape parameter over the iterations

References

Clauset A, Shalizi CR, Newman ME (2009). "Power-law distributions in empirical data." *SIAM review*, **51**(4), 661–703.

Examples

```
## Determine cuttof from compostie lognormal-Pareto distribution using Clauset's method
dist <- c("lnorm", "pareto")
coeff <- c(coeff1.meanlog = -0.5, coeff1.sdlog = 0.5, coeff2.k = 1.5)
x <- rcomposite(1e3, dist = dist, coeff = coeff)
out <- clauset.xmin(x = x)
out$coefficients
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))$coeff2

## Speed up method by considering values above certain quantile only
dist <- c("lnorm", "pareto")
coeff <- c(coeff1.meanlog = -0.5, coeff1.sdlog = 0.5, coeff2.k = 1.5)
x <- rcomposite(1e3, dist = dist, coeff = coeff)
out <- clauset.xmin(x = x, q = 0.5)
out$coefficients
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))$coeff2
```

`coeffcomposite` *Parametrise two-/three- composite distribution*

Description

Determines the weights and cutoffs of the three-composite distribution numerically applying te continuity- and differentiability condition.

Usage

```
coeffcomposite(dist, coeff, startc = c(1, 1))
```

Arguments

- `dist` — character vector denoting the distribution of the first-, second- (and third) component respectively. If only two components are provided, the distribution reduces to the two-component distribution.
- `coeff` — named numeric vector holding the coefficients of the first-, second- (and third) component, predeced by coeff1., coeff2. (and coeff3.), respectively. Coefficients for the last component do not have to be provided for the two-component distribution and will be disregarded.
- `startc` — starting values for the lower and upper cutoff, defaults to c(1,1).

Details

The continuity condition implies

$$\alpha_1 = \frac{m_2(c_1)M_1(c_1)}{m_1(c_1)[M_2(c_2) - M_2(c_1)]}, \qquad \alpha_2 = \frac{m_2(c_2)[1 - M_3(c_2)]}{m_3(c_2)[M_2(c_2) - M_2(c_1)]}$$

The differentiability condition implies

$$\frac{d}{dc_1} ln[\frac{m_1(c_1)}{m_2(c_1)}] = 0, \qquad \frac{d}{dc_2} ln[\frac{m_2(c_2)}{m_3(c_2)}] = 0$$

Value

Returns a named list containing a the separate distributions and their respective coefficients, as well as the cutoffs and weights of the composite distribution

Examples

```
# Three-composite distribution
dist <- c("invpareto", "lnorm", "pareto")
coeff <- c(coeff1.k = 1, coeff2.meanlog = -0.5, coeff2.sdlog = 0.5, coeff3.k = 1)
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))

# Two-composite distribution
dist <- c("lnorm", "pareto")
coeff <- c(coeff1.meanlog = -0.5, coeff1.sdlog = 0.5, coeff2.k = 1.5)
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))

dist <- c("invpareto", "lnorm")
coeff <- c(coeff1.k = 1.5, coeff2.meanlog = 2, coeff2.sdlog = 0.5)
coeffcomposite(dist = dist, coeff = coeff, startc = c(1, 1))
#'
```

combdist *Combined distributions*

Description

Density, distribution function, quantile function, raw moments and random generation for combined (empirical, single, composite and finite mixture) truncated or complete distributions.

Usage

```
dcombdist(
  x,
  dist,
  prior = c(1),
  coeff,
  log = FALSE,
  compress = TRUE,
  lowertrunc = 0,
  uppertrunc = Inf
)

pcombdist(
  q,
  dist,
  prior = 1,
  coeff,
  log.p = FALSE,
  lower.tail = TRUE,
  compress = TRUE,
  lowertrunc = NULL,
  uppertrunc = NULL
)

qcombdist(p, dist, prior, coeff, log.p = FALSE, lower.tail = TRUE)

mcombdist(
  r,
  truncation = NULL,
  dist,
  prior = 1,
  coeff,
  lower.tail = TRUE,
  compress = TRUE,
  uppertrunc = 0,
  lowertrunc = Inf
)
```

```
rcombdist(n, dist, prior, coeff, uppertrunc = NULL, lowertrunc = NULL)
```

Arguments

`x, q`	vector of quantiles
`dist`	character vector denoting the distribution(s).
`prior`	Numeric vector of prior coefficients, defaults to single vector with value one.
`coeff`	list of parameters for the distribution(s).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`compress`	Logical indicating whether return values from individual densities of finite mixtures should be gathered or not, defaults to TRUE.
`lowertrunc, uppertrunc`	lowertrunc- and uppertrunc truncation points, defaults to 0 and Inf respectively
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r \vert X \leq y])$, otherwise, $P[X > x]$ $(E[x^r \vert X > y])$
`p`	vector of probabilities
`r`	rth raw moment of the Pareto distribution
`truncation`	lower truncation parameter
`n`	number of observations

Value

dcombdist gives the density, pcombdist gives the distribution function, qcombdist gives the quantile function, mcombdist gives the rth moment of the distribution and rcombdist generates random deviates.

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

Examples

```
# Load necessary tools
data("fit_US_cities")
library(tidyverse)
x <- rcombdist(
  n = 25359, dist = "lnorm",
  prior = subset(fit_US_cities, (dist == "lnorm" & components == 5))$prior[[1]],
  coeff = subset(fit_US_cities, (dist == "lnorm" & components == 5))$coefficients[[1]]
) # Generate data from one of the fitted functions

# Evaluate functioning of dcomdist by calculating log likelihood for all distributions
loglike <- fit_US_cities %>%
  group_by(dist, components, np, n) %>%
  do(loglike = sum(dcombdist(dist = .[["dist"]], x = sort(x), prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]], log = TRUE))) %>%
  unnest(cols = loglike) %>%
```

```
  mutate(NLL = -loglike, AIC = 2 * np - 2 * (loglike), BIC = log(n) * np - 2 * (loglike)) %>%
  arrange(NLL)

# Evaluate functioning of mcombdist and pcombdist by calculating NMAD
#(equivalent to the Kolmogorov-Smirnov test statistic for the zeroth moment
#of the distribution) for all distributions
nmad <- fit_US_cities %>%
  group_by(dist, components, np, n) %>%
  do(
    KS = max(abs(pempirical(q = sort(x), data = x) - pcombdist(dist = .[["dist"]],
    q = sort(x), prior = .[["prior"]][[1]], coeff = .[["coefficients"]][[1]]))),
    nmad_0 = nmad_test(r = 0, dist = .[["dist"]], x = sort(x), prior = .[["prior"]][[1]],
    coeff = .[["coefficients"]][[1]], stat = "max"),
    nmad_1 = nmad_test(r = 1, dist = .[["dist"]], x = sort(x), prior = .[["prior"]][[1]],
    coeff = .[["coefficients"]][[1]], stat = "max")
  ) %>%
  unnest(cols = c(KS, nmad_0, nmad_1)) %>%
  arrange(nmad_0)

# Evaluate functioning of qcombdist pcombdist by calculating NMAD (equivalent to the Kolmogorov-
#Smirnov test statistic for the zeroth moment of the distribution) for all distributions
test <- fit_US_cities %>%
  group_by(dist, components, np, n) %>%
  do(out = qcombdist(pcombdist(2, dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]], log.p = TRUE),
    dist = .[["dist"]], prior = .[["prior"]][[1]], coeff = .[["coefficients"]][[1]],
    log.p = TRUE
  )) %>%
  unnest(cols = c(out))
```

`combdist.mle` *Combined distributions MLE*

Description

Maximum Likelihood estimation for combined (single, composite and finite mixture) truncated or complete distributions.

Usage

```
combdist.mle(
  x,
  dist,
  start = NULL,
  lower = NULL,
  upper = NULL,
  components = 1,
  nested = FALSE,
```

```
  steps = 1,
  lowertrunc = 0,
  uppertrunc = Inf,
  ...
)
```

Arguments

- `x` data vector
- `dist` character vector denoting the distribution(s).
- `start` named numeric vector holding the starting values for the coefficients.
- `lower, upper` Lower and upper bounds to the estimated coefficients, defaults to -Inf and Inf respectively.
- `components` number of components for a mixture distribution.
- `nested` logical indicating whether results should be returned in a nested list or a flat list form, defaults to FALSE.
- `steps` number of steps taken in stepflexmix, defaults to 1.
- `lowertrunc, uppertrunc` lowertrunc- and uppertrunc truncation points, defaults to 0 and Inf respectively
- `...` Additional arguments.

Value

Returns a named list containing a

dist Character vector denoting the distributions, separated by an underscore

components Nr. of combined distributions

prior Weights assigned to the respective component distributions

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

Examples

```
x <- rdoubleparetolognormal(1e3)
combdist.mle(x = x, dist = "doubleparetolognormal") # Double-Pareto Lognormal
combdist.mle(x = x, components = 2, dist = "lnorm", steps = 20) # FMM with 2 components
combdist.mle( x = x, dist = c("invpareto", "lnorm", "pareto"),
start = c(coeff1.k = 1, coeff2.meanlog = mean(log(x)), coeff2.sdlog = sd(log(x)), coeff3.k = 1),
lower = c(1e-10, -Inf, 1e-10, 1e-10), upper = c(Inf, Inf, Inf, Inf), nested = TRUE)
# composite distribution
```

`combdist_plt` *Combined coefficients of power-law transformed combined distribution*

Description

Coefficients of a power-law transformed combined distribution

Usage

```
combdist_plt(
  dist,
  prior = NULL,
  coeff,
  a = 1,
  b = 1,
  inv = FALSE,
  nested = FALSE
)
```

Arguments

`dist`	character vector denoting the distribution(s).
`prior`	Numeric vector of prior coefficients, defaults to single vector with value one.
`coeff`	list of parameters for the distribution(s).
`a, b`	constant and power of power-law transformation, defaults to 1 and 1 respectively.
`inv`	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.
`nested`	logical indicating whether results should be returned in a nested list or flat list, defaults to FALSE.

Value

Returns a nested or flat list containing

coefficients Named vector of coefficients

Examples

```
# Load necessary tools
data("fit_US_cities")
```

```
library(tidyverse)

## Comparing probabilites of power-law transformed transformed variables
prob <- fit_US_cities %>%
  filter(!(dist %in% c(
    "exp", "invpareto_exp_pareto", "exp_pareto", "invpareto_exp",
    "gamma", "invpareto_gamma_pareto", "gamma_pareto", "invpareto_gamma"
  ))) %>%
  group_by(dist, components, np, n) %>%
  do(prob = pcombdist(q = 1.1, dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]])) %>%
  unnest(cols = c(prob))
fit_US_cities_plt <- fit_US_cities %>%
  filter(!(dist %in% c(
    "exp", "invpareto_exp_pareto", "exp_pareto", "invpareto_exp",
    "gamma", "invpareto_gamma_pareto", "gamma_pareto", "invpareto_gamma"
  ))) %>%
  group_by(dist, components, np, n, convergence) %>%
  do(results = as_tibble(combdist_plt(dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]], a = 2, b = 0.5, nested = TRUE))) %>%
  unnest(cols = c(results))
prob$prob_plt <- fit_US_cities_plt %>%
  group_by(dist, components, np, n) %>%
  do(prob_plt = pcombdist(q = 2 * 1.1^0.5, dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]])) %>%
  unnest(cols = c(prob_plt)) %>%
  .$prob_plt
prob <- prob %>%
  mutate(check = abs(prob - prob_plt))

prob <- fit_US_cities %>%
  filter(!(dist %in% c(
    "exp", "invpareto_exp_pareto", "exp_pareto", "invpareto_exp",
    "gamma", "invpareto_gamma_pareto", "gamma_pareto", "invpareto_gamma"
  ))) %>%
  group_by(dist, components, np, n) %>%
  do(prob = pcombdist(q = 2 * 1.1^0.5, dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]])) %>%
  unnest(cols = c(prob))
fit_US_cities_plt <- fit_US_cities %>%
  filter(!(dist %in% c(
    "exp", "invpareto_exp_pareto", "exp_pareto", "invpareto_exp",
    "gamma", "invpareto_gamma_pareto", "gamma_pareto", "invpareto_gamma"
  ))) %>%
  group_by(dist, components, np, n, convergence) %>%
  do(results = as_tibble(combdist_plt(dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]], a = 2, b = 0.5, nested = TRUE, inv = TRUE))) %>%
  unnest(cols = c(results))
prob$prob_plt <- fit_US_cities_plt %>%
  group_by(dist, components, np, n) %>%
  do(prob_plt = pcombdist(q = 1.1, dist = .[["dist"]], prior = .[["prior"]][[1]],
  coeff = .[["coefficients"]][[1]])) %>%
```

```
  unnest(cols = c(prob_plt)) %>%
  .$prob_plt
prob <- prob %>%
  mutate(check = abs(prob - prob_plt))
```

---

composite *The two- or three-composite distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for the two- or three-composite distribution.

Usage

```
dcomposite(x, dist, coeff, startc = c(1, 1), log = FALSE)

pcomposite(q, dist, coeff, startc = c(1, 1), log.p = FALSE, lower.tail = TRUE)

qcomposite(p, dist, coeff, startc = c(1, 1), log.p = FALSE, lower.tail = TRUE)

mcomposite(
  r = 0,
  truncation = 0,
  dist,
  coeff,
  startc = c(1, 1),
  lower.tail = TRUE
)

rcomposite(n, dist, coeff, startc = c(1, 1))
```

Arguments

`x, q`	vector of quantiles
`dist`	character vector denoting the distribution of the first-, second- (and third) component respectively. If only two components are provided, the distribution reduces to the two-component distribution.
`coeff`	named numeric vector holding the coefficients of the first-, second- (and third) component, predeced by coeff1., coeff2. (and coeff3.), respectively. Coefficients for the last component do not have to be provided for the two-component distribution and will be disregarded.
`startc`	starting values for the lower and upper cutoff, defaults to c(1,1).
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).

`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ ($E[x^r \| X \leq y]$), otherwise, $P[X > x]$ ($E[x^r \| X > y]$)
`p`	vector of probabilities
`r`	rth raw moment of the Pareto distribution
`truncation`	lower truncation parameter
`n`	number of observations

Details

These derivations are based on the two-composite distribution proposed by (Bakar et al. 2015). Probability Distribution Function:

$$f(x) = \{ \begin{array}{ll} \frac{\alpha_1}{1+\alpha_1+\alpha_2}\frac{m_1(x)}{M_1(c_1)} & if \quad 0 < x \leq c_1 \\ \frac{1}{1+\alpha_1+\alpha_2}\frac{m_2(x)}{M_2(c_2)-M_2(c_1)} & if \quad c_1 < x \leq c_2 \\ \frac{\alpha_2}{1+\alpha_1+\alpha_2}\frac{m_3(x)}{1-M_3(c_2)} & if \quad c_2 < x < \infty \end{array}.$$

Cumulative Distribution Function:

$$\{ \begin{array}{ll} \frac{\alpha_1}{1+\alpha_1+\alpha_2}\frac{M_1(x)}{M_1(c_1)} & if \quad 0 < x \leq c_1 \\ \frac{\alpha_1}{1+\alpha_1+\alpha_2} + \frac{1}{1+\alpha_1+\alpha_2}\frac{M_2(x)-M_2(c_1)}{M_2(c_2)-M_2(c_1)} & if \quad c_1 < x \leq c_2 \\ \frac{1+\alpha_1}{1+\alpha_1+\alpha_2} + \frac{\alpha_2}{1+\alpha_1+\alpha_2}\frac{M_3(x)-M_3(c_2)}{1-M_3(c_2)} & if \quad c_2 < x < \infty \end{array}.$$

Quantile function

$$Q(p) = \{ \begin{array}{ll} Q_1(\frac{1+\alpha_1+\alpha_2}{\alpha_1}pM_1(c_1)) & if \quad 0 < x \leq \frac{\alpha_1}{1+\alpha_1+\alpha_2} \\ Q_2[((p - \frac{\alpha_1}{1+\alpha_1+\alpha_2})(1+\alpha_1+\alpha_2)(M_2(c_2)-M_2(c_1))) + M_2(c_1)] & if \quad \frac{\alpha_1}{1+\alpha_1+\alpha_2} < x \leq \frac{1+\alpha_1}{1+\alpha_1+\alpha_2} \\ Q_3[((p - \frac{1+\alpha_1}{1+\alpha_1+\alpha_2})(\frac{1+\alpha_1+\alpha_2}{\alpha_2})(1-M_3(c_2))) + M_3(c_2)] & if \quad \frac{1+\alpha_1}{1+\alpha_1+\alpha_2} < x < \infty \end{array}.$$

The lower y-bounded r-th raw moment of the distribution equals

$$\mu_y^r = \{ \begin{array}{ll} \frac{\alpha_1}{1+\alpha_1+\alpha_2}\frac{(\mu_1)_y^r-(\mu_1)_{c_1}^r}{M_1(c_1)} + \frac{1}{1+\alpha_1+\alpha_2}\frac{(\mu_2)_{c_1}^r-(\mu_2)_{c_2}^r}{M_2(c_2)-M_2(c_1)} + \frac{\alpha_2}{1+\alpha_1+\alpha_2}\frac{(\mu_3)_y^r}{1-M_3(c_2)} & if \quad 0 < y \leq c_2 \\ \frac{1}{1+\alpha_1+\alpha_2}\frac{(\mu_2)_y^r-(\mu_2)_{c_2}^r}{M_2(c_2)-M_2(c_1)} + \frac{\alpha_2}{1+\alpha_1+\alpha_2}\frac{(\mu_3)_{c_2}^r}{1-M_3(c_2)} & if \quad c_1 < y \leq c_2 \\ \frac{\alpha_2}{1+\alpha_1+\alpha_2}\frac{(\mu_3)_y^r}{1-M_3(c_2)} & if \quad c_2 < y < \infty \end{array}.$$

Value

dcomposite returns the density, pcomposite the distribution function, qcomposite the quantile function, mcomposite the rth moment of the distribution and rcomposite generates random deviates.

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

References

Bakar SA, Hamzah N, Maghsoudi M, Nadarajah S (2015). "Modeling loss data using composite models." *Insurance: Mathematics and Economics*, **61**, 146–154.

Examples

```
#' ## Three-component distribution
dist <- c("invpareto", "lnorm", "pareto")
coeff <- c(coeff2.meanlog = -0.5, coeff2.sdlog = 0.5, coeff3.k = 1.5, coeff1.k = 1.5)

# Compare density with the Double-Pareto Lognormal distribution
plot(x = seq(0, 5, length.out = 1e3), y = dcomposite(x = seq(0, 5, length.out = 1e3),
dist = dist, coeff = coeff))
lines(x = seq(0, 5, length.out = 1e3), y = ddoubleparetolognormal(x = seq(0, 5, length.out = 1e3)))

# Demonstration of log functionality for probability and quantile function
qcomposite(pcomposite(2, dist = dist, coeff = coeff, log.p = TRUE), dist = dist,
coeff = coeff, log.p = TRUE)

# The zeroth truncated moment is equivalent to the probability function
pcomposite(2, dist = dist, coeff = coeff)
mcomposite(truncation = 2, dist = dist, coeff = coeff)

# The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
coeff <- c(coeff2.meanlog = -0.5, coeff2.sdlog = 0.5, coeff3.k = 3, coeff1.k = 1.5)
x <- rcomposite(1e5, dist = dist, coeff = coeff)

mean(x)
mcomposite(r = 1, lower.tail = FALSE, dist = dist, coeff = coeff)

sum(x[x > quantile(x, 0.1)]) / length(x)
mcomposite(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE, dist = dist, coeff = coeff)

## Two-component distribution
dist <- c("lnorm", "pareto")
coeff <- coeff <- c(coeff2.k = 1.5, coeff1.meanlog = -0.5, coeff1.sdlog = 0.5)

# Compare density with the Right-Pareto Lognormal distribution
plot(x = seq(0, 5, length.out = 1e3), y = dcomposite(x = seq(0, 5, length.out = 1e3),
dist = dist, coeff = coeff))
lines(x = seq(0, 5, length.out = 1e3), y = drightparetolognormal(x = seq(0, 5, length.out = 1e3)))

# Demonstration of log functionality for probability and quantile function
qcomposite(pcomposite(2, dist = dist, coeff = coeff, log.p = TRUE), dist = dist,
coeff = coeff, log.p = TRUE)

# The zeroth truncated moment is equivalent to the probability function
pcomposite(2, dist = dist, coeff = coeff)
mcomposite(truncation = 2, dist = dist, coeff = coeff)

# The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
coeff <- c(coeff1.meanlog = -0.5, coeff1.sdlog = 0.5, coeff2.k = 3)
x <- rcomposite(1e5, dist = dist, coeff = coeff)
```

```
mean(x)
mcomposite(r = 1, lower.tail = FALSE, dist = dist, coeff = coeff)

sum(x[x > quantile(x, 0.1)]) / length(x)
mcomposite(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE, dist = dist, coeff = coeff)
```

---

composite.mle *Composite MLE*

Description

Maximum likelihood estimation of the parameters of the two-/three- composite distribution

Usage

```
composite.mle(x, dist, start, lower = NULL, upper = NULL)
```

Arguments

`x`	data vector
`dist`	character vector denoting the distribution of the first-, second- (and third) component respectively. If only two components are provided, the distribution reduces to the two-component distribution.
`start`	named numeric vector holding the coefficients of the first-, second- (and third) component, predeced by coeff1., coeff2. (and coeff3.), respectively. Coefficients for the last component do not have to be provided for the two-component distribution and will be disregarded.
`lower, upper`	Lower and upper bounds to the estimated coefficients, defaults to -Inf and Inf respectively.

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

cutoffs Cutoffs of the composite distribution

n Length of the fitted data vector

np Nr. of coefficients

components Nr. of components

Examples

```
dist <- c("invpareto", "lnorm", "pareto")
coeff <- c(
  coeff1.k = 1.5, coeff2.meanlog = -0.5,
  coeff2.sdlog = 0.5, coeff3.k = 1.5
)
lower <- c(1e-10, -Inf, 1e-10, 1e-10)
upper <- c(Inf, Inf, Inf, Inf)
x <- rcomposite(1e3, dist = dist, coeff = coeff)

composite.mle(x = x, dist = dist, start = coeff + 0.2, lower = lower, upper = upper)

#'
```

`composite_plt` *Composite coefficients after power-law transformation*

Description

Coefficients of a power-law transformed composite distribution

Usage

```
composite_plt(dist, coeff, a = 1, b = 1, inv = FALSE)
```

Arguments

`dist`	character vector denoting the distribution of the first-, second- (and third) component respectively. If only two components are provided, the distribution reduces to the two-component distribution.
`coeff`	named numeric vector holding the coefficients of the first-, second- (and third) component, predeced by coeff1., coeff2. (and coeff3.), respectively. Coefficients for the last component do not have to be provided for the two-component distribution and will be disregarded.
`a, b`	constant and power of power-law transformation, defaults to 1 and 1 respectively.
`inv`	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables dist <- c("invpareto", "lnorm", "pareto") coeff <- c(coeff2.meanlog = -0.5, coeff2.sdlog = 0.5, coeff3.k = 1.5, coeff1.k = 1.5)

pcomposite(3,dist=dist,coeff=coeff) newcoeff = composite_plt(dist=dist,coeff=coeff,a=5,b=7)$coefficients pcomposite(5*3^7,dist=dist,coeff=newcoeff)

pcomposite(5*0.9^3,dist=dist,coeff=coeff) newcoeff = composite_plt(dist=dist,coeff=coeff,a=5,b=3,inv=TRUE)$coefficients pcomposite(0.9,dist=dist,coeff=newcoeff)

---

doubleparetolognormal *The Double-Pareto Lognormal distribution*

Description

Density, distribution function, quantile function and random generation for the Double-Pareto Lognormal distribution.

Usage

```
ddoubleparetolognormal(
  x,
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  log = FALSE
)

pdoubleparetolognormal(
  q,
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

qdoubleparetolognormal(
  p,
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)
```

```
mdoubleparetolognormal(
  r = 0,
  truncation = 0,
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE
)

rdoubleparetolognormal(
  n,
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5
)
```

Arguments

`x, q`	vector of quantiles
`shape1, shape2, meanlog, sdlog`	Shapes, mean and variance of the Double-Pareto Lognormal distribution respectively, defaults to shape1=1.5, shape2=1.5, meanlog=-0.5, sdlog=0.5.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ ($E[x^r \vert X \leq y]$), otherwise, $P[X > x]$ ($E[x^r \vert X > y]$)
`p`	vector of probabilities
`r`	rth raw moment of the Pareto distribution
`truncation`	lower truncation parameter, defaults to xmin
`n`	number of observations

Details

Probability and Cumulative Distribution Function as provided by (Reed and Jorgensen 2004):

$f(x) = \frac{shape2 shape1}{shape2 + shape1}[x^{-shape2-1} e^{shape2 meanlog + \frac{shape2^2 sdlog^2}{2}} \Phi(\frac{lnx - meanlog - shape2 sdlog^2}{sdlog}) + x^{shape1-1} e^{-shape1 meanlog +$

$F_X(x) = \Phi(\frac{lnx - meanlog}{sdlog}) - \frac{1}{shape2 + shape1}[shape1 x^{-shape2} e^{shape2 meanlog + \frac{shape2^2 sdlog^2}{2}} \Phi(\frac{lnx - meanlog - shape2 sdlog^2}{sdlog}) - shape2 x^{shape1} e^{-shape1 meanlog + \frac{shape1^2 sdlog^2}{2}} \Phi^c(\frac{lnx - meanlog + shape1 sdlog^2}{sdlog})]$

The y-bounded r-th raw moment of the Double-Pareto Lognormal distribution equals:

$meanlog^r_y = -\frac{shape2 shape1}{shape2 + shape1} e^{shape2 meanlog + \frac{shape2^2 sdlog^2}{2}} \frac{y^{r - shape2}}{r - shape2} \Phi(\frac{lny - meanlog - shape2 sdlog^2}{sdlog})$
$- \frac{shape2 shape1}{shape2 + shape1} \frac{1}{r - shape2} e^{\frac{r^2 sdlog^2 + 2 meanlog r}{2}} \Phi^c(\frac{lny - r sdlog^2 - meanlog}{sdlog})$
$- \frac{shape2 shape1}{shape2 + shape1} e^{-shape1 meanlog + \frac{shape1^2 sdlog^2}{2}} \frac{y^{r + shape1}}{r + shape1} \Phi^c(\frac{lny - meanlog + shape1 sdlog^2}{sdlog})$
$+ \frac{shape2 shape1}{shape2 + shape1} \frac{1}{r + shape1} e^{\frac{r^2 sdlog^2 + 2 meanlog r}{2}} \Phi^c(\frac{lny - r sdlog^2 - meanlog}{sdlog}), \qquad shape2 > r$

Value

ddoubleparetolognormal returns the density, pdoubleparetolognormal the distribution function, qdoubleparetolognormal the quantile function, mdoubleparetolognormal the rth moment of the distribution and rdoubleparetolognormal generates random deviates.

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

References

Reed WJ, Jorgensen M (2004). "The Double Pareto-Lognormal Distribution–A New Parametric Model for Size Distributions." *Communications in Statistics - Theory and Methods*, **33**(8), 1733–1753.

Examples

```
## Double-Pareto Lognormal density
plot(x = seq(0, 5, length.out = 100), y = ddoubleparetolognormal(x = seq(0, 5, length.out = 100)))
plot(x = seq(0, 5, length.out = 100), y = ddoubleparetolognormal(x = seq(0, 5, length.out = 100),
 shape2 = 1))

## Double-Pareto Lognormal relates to the right-pareto Lognormal distribution if
#shape1 goes to infinity
pdoubleparetolognormal(q = 6, shape1 = 1e20, shape2 = 1.5, meanlog = -0.5, sdlog = 0.5)
prightparetolognormal(q = 6, shape2 = 1.5, meanlog = -0.5, sdlog = 0.5)

## Double-Pareto Lognormal relates to the left-pareto Lognormal distribution if
# shape2 goes to infinity
pdoubleparetolognormal(q = 6, shape1 = 1.5, shape2 = 1e20, meanlog = -0.5, sdlog = 0.5)
pleftparetolognormal(q = 6, shape1 = 1.5, meanlog = -0.5, sdlog = 0.5)

## Double-Pareto Lognormal relates to the Lognormal if both shape parameters go to infinity
pdoubleparetolognormal(q = 6, shape1 = 1e20, shape2 = 1e20, meanlog = -0.5, sdlog = 0.5)
plnorm(q = 6, meanlog = -0.5, sdlog = 0.5)

## Demonstration of log functionality for probability and quantile function
qdoubleparetolognormal(pdoubleparetolognormal(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
pdoubleparetolognormal(2)
mdoubleparetolognormal(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rdoubleparetolognormal(1e5, shape2 = 3)

mean(x)
mdoubleparetolognormal(r = 1, shape2 = 3, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mdoubleparetolognormal(r = 1, shape2 = 3, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

`doubleparetolognormal.mle`
Double-Pareto Lognormal MLE

Description

Maximum likelihood estimation of the parameters of the Double-Pareto Lognormal distribution.

Usage

```
doubleparetolognormal.mle(
  x,
  lower = c(1e-10, 1e-10, 1e-10),
  upper = c(Inf, Inf, Inf),
  start = NULL
)
```

Arguments

`x`	data vector
`lower, upper`	Upper and lower bounds for the estimation procedure on the parameters c(shape2,shape1,sdlog), defaults to c(1e-10,1e-10,1e-10) and c(Inf,Inf,Inf) respectively.
`start`	named vector with starting values, default to c(shape2=2,shape1=2,sdlog=sd(log(x)))

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

Examples

```
x <- rdoubleparetolognormal(1e3)

## Pareto fit with xmin set to the minium of the sample
doubleparetolognormal.mle(x = x)
```

doubleparetolognormal_plt *Double-Pareto Lognormal coefficients of power-law transformed Double-Pareto Lognormal*

Description

Coefficients of a power-law transformed Double-Pareto Lognormal distribution

Usage

```
doubleparetolognormal_plt(
  shape1 = 1.5,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  a = 1,
  b = 1,
  inv = FALSE
)
```

Arguments

shape1, shape2, meanlog, sdlog	Shapes, mean and variance of the Double-Pareto Lognormal distribution respectively.
a, b	constant and power of power-law transformation, defaults to 1 and 1 respectively.
inv	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable y is Double-Pareto Lognormal distributed with mean meanlog and standard deviation sdlog, then the power-law transformed variable

$$y = ax^b$$

is Double-Pareto Lognormal distributed with $shape1 * b$, $\frac{meanlog-log(a)}{b}$, $\frac{sdlog}{b}$, $shape2 * b$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables pdoubleparetolognormal(3,shape1 = 1.5, shape2 = 3, meanlog = -0.5, sdlog = 0.5) coeff = doubleparetolognormal_plt(shape1 = 1.5, shape2 = 3, meanlog = -0.5, sdlog = 0.5,a=5,b=7)$coefficients pdoubleparetolognormal(5*3^7,shape1=coeff[["shape1"

pdoubleparetolognormal(5*0.9^7,shape1 = 1.5, shape2 = 3, meanlog = -0.5, sdlog = 0.5) coeff = doubleparetolognormal_plt(shape1 = 1.5, shape2 = 3, meanlog = -0.5, sdlog = 0.5,a=5,b=7, inv=TRUE)$coefficients pdoubleparetolognormal(0.9,shape1=coeff[["shape1"]],shape2=coeff[["shape2"]],meanlog=coeff[[

`empirical` *The empirical distribution*

Description

Density, distribution function, quantile function, and raw moments for the empirical distribution.

Usage

```
dempirical(x, data, log = FALSE)

pempirical(q, data, log.p = FALSE, lower.tail = TRUE)

qempirical(p, data, lower.tail = TRUE, log.p = FALSE)

mempirical(r = 0, data, truncation = NULL, lower.tail = TRUE)
```

Arguments

`x, q`	vector of quantiles
`data`	data vector
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), moments are $E[x^r\|X \leq y]$, otherwise, $E[x^r\|X > y]$
`p`	vector of probabilities
`r`	rth raw moment of the Pareto distribution
`truncation`	lower truncation parameter, defaults to NULL.

Details

The density function is a standard Kernel density estimation for 1e6 equally spaced points. The cumulative Distribution Function:

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{x_i \leq x}$$

The y-bounded r-th raw moment of the empirical distribution equals:

$$\mu_y^r = \frac{1}{n}\sum_{i=1}^{n} I_{x_i \leq x} x^r$$

Value

dempirical returns the density, pempirical the distribution function, qempirical the quantile function, mempirical gives the rth moment of the distribution or a function that allows to evaluate the rth moment of the distribution if truncation is NULL..

Examples

```
#'
## Generate random sample to work with
x <- rlnorm(1e5, meanlog = -0.5, sdlog = 0.5)

## Empirical density
plot(x = seq(0, 5, length.out = 100), y = dempirical(x = seq(0, 5, length.out = 100), data = x))

# Compare empirical and parametric quantities
dlnorm(0.5, meanlog = -0.5, sdlog = 0.5)
dempirical(0.5, data = x)

plnorm(0.5, meanlog = -0.5, sdlog = 0.5)
pempirical(0.5, data = x)

qlnorm(0.5, meanlog = -0.5, sdlog = 0.5)
qempirical(0.5, data = x)

mlnorm(r = 0, truncation = 0.5, meanlog = -0.5, sdlog = 0.5)
mempirical(r = 0, truncation = 0.5, data = x)

mlnorm(r = 1, truncation = 0.5, meanlog = -0.5, sdlog = 0.5)
mempirical(r = 1, truncation = 0.5, data = x)

## Demonstration of log functionailty for probability and quantile function
quantile(x, 0.5, type = 1)
qempirical(p = pempirical(q = quantile(x, 0.5, type = 1), data = x, log.p = TRUE),
data = x, log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
pempirical(q = quantile(x, 0.5, type = 1), data = x)
```

```
mempirical(truncation = quantile(x, 0.5, type = 1), data = x)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
mean(x)
mempirical(r = 1, data = x, truncation = 0, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mempirical(r = 1, data = x, truncation = quantile(x, 0.1), lower.tail = FALSE)
#'
```

---

exp *The Exponential distribution*

Description

Raw moments for the exponential distribution.

Usage

```
mexp(r = 0, truncation = 0, rate = 1, lower.tail = TRUE)
```

Arguments

`r`	rth raw moment of the distribution, defaults to 1.
`truncation`	lower truncation parameter, defaults to 0.
`rate`	rate of the distribution with default values of 1.
`lower.tail`	logical; if TRUE (default), moments are $E[x^r \mid X \leq y]$, otherwise, $E[x^r \mid X > y]$

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{1}{s} e^{-\frac{\omega}{s}}, \qquad F_X(x) = 1 - e^{-\frac{\omega}{s}}$$

The y-bounded r-th raw moment of the distribution equals:

$$s^{\sigma_s - 1} \Gamma\left(\sigma_s + 1, \frac{y}{s}\right)$$

where $\Gamma(,)$ denotes the upper incomplete gamma function.

Value

Returns the truncated rth raw moment of the distribution.

Examples

```
## The zeroth truncated moment is equivalent to the probability function
pexp(2, rate = 1)
mexp(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rexp(1e5, rate = 1)
mean(x)
mexp(r = 1, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mexp(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

---

fit_US_cities — *Fitted distributions to the US Census 2000 city size distribution.*

Description

A dataset containing 52 distribution fits to the US Census 2000 city size distributions

Usage

```
fit_US_cities
```

Format

A data frame with 52 rows and 7 variables:

dist distribution

components number of components

prior list of prior weights for the individual distribution components of FMM

coefficients list of coefficients for the distributions

np Number of paramters

n Number of observations

convergence Logical indicating whether the fitting procedure converged

Source

http://doi.org/10.3886/E113328V1

`frechet` *The Fréchet distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for the Fréchet distribution.

Usage

```
dfrechet(x, shape = 1.5, scale = 0.5, log = FALSE)

pfrechet(q, shape = 1.5, scale = 0.5, log.p = FALSE, lower.tail = TRUE)

qfrechet(p, shape = 1.5, scale = 0.5, log.p = FALSE, lower.tail = TRUE)

mfrechet(r = 0, truncation = 0, shape = 1.5, scale = 0.5, lower.tail = TRUE)

rfrechet(n, shape = 1.5, scale = 0.5)
```

Arguments

`x, q`	vector of quantiles
`shape, scale`	Shape and scale of the Fréchet distribution, defaults to 1.5 and 0.5 respectively.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r\|X \leq y])$, otherwise, $P[X > x]$ $(E[x^r\|X > y])$
`p`	vector of probabilities
`r`	rth raw moment of the distribution
`truncation`	lower truncation parameter
`n`	number of observations

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{shape}{scale}\left(\frac{\omega}{scale}\right)^{-1-shape} e^{-\left(\frac{\omega}{scale}\right)^{-shape}}, \qquad F_X(x) = e^{-\left(\frac{\omega}{scale}\right)^{-shape}}$$

The y-bounded r-th raw moment of the Fréchet distribution equals:

$$\mu_y^r = scale^{\sigma_s - 1}\left[1 - \Gamma\left(1 - \frac{\sigma_s - 1}{shape}, \left(\frac{y}{scale}\right)^{-shape}\right)\right], \qquad shape > r$$

Value

dfrechet returns the density, pfrechet the distribution function, qfrechet the quantile function, mfrechet the rth moment of the distribution and rfrechet generates random deviates.

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

Examples

```
## Frechet density
plot(x = seq(0, 5, length.out = 100), y = dfrechet(x = seq(0, 5, length.out = 100),
shape = 1, scale = 1))
plot(x = seq(0, 5, length.out = 100), y = dfrechet(x = seq(0, 5, length.out = 100),
shape = 2, scale = 1))
plot(x = seq(0, 5, length.out = 100), y = dfrechet(x = seq(0, 5, length.out = 100),
shape = 3, scale = 1))
plot(x = seq(0, 5, length.out = 100), y = dfrechet(x = seq(0, 5, length.out = 100),
shape = 3, scale = 2))

## frechet is also called the inverse weibull distribution, which is available in the stats package
pfrechet(q = 5, shape = 2, scale = 1.5)
1 - pweibull(q = 1 / 5, shape = 2, scale = 1 / 1.5)

## Demonstration of log functionality for probability and quantile function
qfrechet(pfrechet(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
pfrechet(2)
mfrechet(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rfrechet(1e5, scale = 1)

mean(x)
mfrechet(r = 1, lower.tail = FALSE, scale = 1)

sum(x[x > quantile(x, 0.1)]) / length(x)
mfrechet(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE, scale = 1)
```

---

frechet.mle *Fréchet MLE*

Description

Maximum likelihood estimation of the coefficients of the Fréchet distribution

Usage

```
frechet.mle(
  x,
  weights = NULL,
  start = c(shape = 1.5, scale = 0.5),
  lower = c(1e-10, 1e-10),
  upper = c(Inf, Inf)
)
```

Arguments

`x`	data vector
`weights`	numeric vector for weighted MLE, should have the same length as data vector x
`start`	named vector with starting values, default to c(shape=1.5,scale=0.5)
`lower, upper`	Lower and upper bounds to the estimated shape parameter, defaults to 1e-10 and Inf respectively

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

x = rfrechet(1e3)

Pareto fit with xmin set to the minium of the sample frechet.mle(x=x)

`frechet_plt` *Fréchet coefficients after power-law transformation*

Description

Coefficients of a power-law transformed Fréchet distribution

Usage

```
frechet_plt(shape = 1.5, scale = 0.5, a = 1, b = 1, inv = FALSE)
```

Arguments

`shape, scale`	Scale and shape of the Fréchet distribution, defaults to 1.5 and 0.5 respectively.
`a, b`	constant and power of power-law transformation, defaults to 1 and 1 respectively.
`inv`	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable x is Fréchet distributed with scale shape and shape scale, then the power-law transformed variable

$$y = ax^b$$

is Fréchet distributed with scale $\left(\frac{scale}{a}\right)^{\frac{1}{b}}$ and shape $b * k$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables pfrechet(3,shape=2,scale=1) coeff = frechet_plt(shape=2,scale=1,a=5,b=7)$coefficients pfrechet(5*3^7,shape=coeff[["shape"]],scale=coeff[["scale"]])

pfrechet(5*0.8^7,shape=2,scale=1) coeff = frechet_plt(shape=2,scale=1,a=5,b=7,inv=TRUE)$coefficients pfrechet(0.8,shape=coeff[["shape"]],scale=coeff[["scale"]])

`gamma` *The Gamma distribution*

Description

Raw moments for the Gamma distribution.

Usage

```
mgamma(
  r = 0,
  truncation = 0,
  shape = 2,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE
)
```

Arguments

`r` rth raw moment of the distribution, defaults to 1.

`truncation` lower truncation parameter, defaults to 0.

`shape, rate, scale`
shape, rate and scale of the distribution with default values of 2 and 1 respectively.

`lower.tail` logical; if TRUE (default), moments are $E[x^r|X \le y]$, otherwise, $E[x^r|X > y]$

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{1}{s^k \Gamma(k)} \omega^{k-1} e^{-\frac{\omega}{s}}, \qquad F_X(x) = \frac{1}{\Gamma(k)} \gamma(k, \frac{\omega}{s})$$

,

where $\Gamma(x)$ stands for the upper incomplete gamma function function, while $\gamma(s, x)$ stands for the lower incomplete Gamma function with upper bound x.

The y-bounded r-th raw moment of the distribution equals:

$$\mu_y^r = \frac{s^r}{\Gamma(k)} \Gamma\left(r + k, \frac{y}{s}\right)$$

Value

Provides the truncated rth raw moment of the distribution.

The zeroth truncated moment is equivalent to the probability function pgamma(2,shape=2,rate=1) mgamma(truncation=2)

The (truncated) first moment is equivalent to the mean of a (truncated) random sample, #for large enough samples. x = rgamma(1e5,shape=2,rate=1) mean(x) mgamma(r=1,lower.tail=FALSE)

sum(x[x>quantile(x,0.1)])/length(x) mgamma(r=1,truncation=quantile(x,0.1),lower.tail=FALSE)

`invpareto` *The Inverse Pareto distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for the Pareto distribution.

Usage

```
dinvpareto(x, k = 1.5, xmax = 5, log = FALSE, na.rm = FALSE)

pinvpareto(
  q,
  k = 1.5,
  xmax = 5,
  lower.tail = TRUE,
  log.p = FALSE,
  log = FALSE,
  na.rm = FALSE
)

qinvpareto(p, k = 1.5, xmax = 5, lower.tail = TRUE, log.p = FALSE)

minvpareto(r = 0, truncation = 0, k = 1.5, xmax = 5, lower.tail = TRUE)

rinvpareto(n, k = 1.5, xmax = 5)
```

Arguments

`x, q`	vector of quantiles
`xmax, k`	Scale and shape of the Inverse Pareto distribution, defaults to 5 and 1.5 respectively.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`na.rm`	Removes values that fall outside the support of the distribution
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ ($E[x^r\|X \leq y]$), otherwise, $P[X > x]$ ($E[x^r\|X > y]$)
`p`	vector of probabilities
`r`	rth raw moment of the Inverse Pareto distribution
`truncation`	lower truncation parameter, defaults to xmin
`n`	number of observations

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{kx_{max}^{-k}}{x^{-k+1}}, \qquad F_X(x) = (\frac{x_{max}}{x})^{-k}$$

The y-bounded r-th raw moment of the Inverse Pareto distribution equals:

$$\mu_y^r = k\omega_{max}^{-k}\frac{\omega_{max}^{r+k} - y^{r+k}}{r+k}$$

Value

dinvpareto returns the density, pinvpareto the distribution function, qinvpareto the quantile function, minvpareto the rth moment of the distribution and rinvpareto generates random deviates.

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

Examples

```
## Inverse invpareto density
plot(x = seq(0, 5, length.out = 100), y = dinvpareto(x = seq(0, 5, length.out = 100)))

## Demonstration of log functionality for probability and quantile function
qinvpareto(pinvpareto(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
pinvpareto(2)
minvpareto(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rinvpareto(1e5)

mean(x)
minvpareto(r = 1, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
minvpareto(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

---

invpareto.mle *Inverse Pareto MLE*

Description

Maximum likelihood estimation of the Inverse Pareto shape parameter using the Hill estimator.

Usage

```
invpareto.mle(x, xmax = NULL, clauset = FALSE, q = 1)
```

Arguments

x	data vector
xmax	scale parameter of the Inverse Pareto distribution, set to max(x) if not provided
clauset	Indicator variable for calculating the scale parameter using the clauset method, overrides provided xmax
q	Percentage of data to search over (starting from the smallest values), dafults to 1.

Details

The Hill estimator equals

$$\hat{k} = -\frac{1}{\frac{1}{n}\sum_{i=1}^{n} log\frac{x_{max}}{x_i}}$$

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

Examples

```
x <- rinvpareto(1e3, k = 1.5, xmax = 5)

## Pareto fit with xmin set to the minium of the sample
invpareto.mle(x = x)

## Pareto fit with xmin set to its real value
invpareto.mle(x = x, xmax = 5)

## Pareto fit with xmin determined by the Clauset method
invpareto.mle(x = x, clauset = TRUE)
```

---

invpareto_plt *Inverse Pareto coefficients after power-law transformation*

Description

Coefficients of a power-law transformed Inverse Pareto distribution

Usage

```
invpareto_plt(xmax = 5, k = 1.5, a = 1, b = 1, inv = FALSE)
```

Arguments

`xmax, k`	Scale and shape of the Inverse Pareto distribution, defaults to 5 and 1.5 respectively.
`a, b`	constant and power of power-law transformation, defaults to 1 and 1 respectively.

inv logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable x is Inverse Pareto-distributed with scale xmin and shape k, then the power-law transformed variable

$$y = ax^b$$

is Inverse Pareto distributed with scale $(\frac{xmin}{a})^{\frac{1}{b}}$ and shape $b * k$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables pinvpareto(3,k=2,xmax=5) coeff = invpareto_plt(xmax=5,k=2,a=5,b=7)$coefficients pinvpareto(5*3^7,k=coeff[["k"]],xmax=coeff[["xmax"]])

pinvpareto(5*0.9^7,k=2,xmax=5) coeff = invpareto_plt(xmax=5,k=2,a=5,b=7, inv=TRUE)$coefficients pinvpareto(0.9,k=coeff[["k"]],xmax=coeff[["xmax"]])

`leftparetolognormal` *The Left-Pareto Lognormal distribution*

Description

Density, distribution function, quantile function and random generation for the Left-Pareto Lognormal distribution.

Usage

```
dleftparetolognormal(x, shape1 = 1.5, meanlog = -0.5, sdlog = 0.5, log = FALSE)

pleftparetolognormal(
  q,
  shape1 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

qleftparetolognormal(
  p,
```

```
  shape1 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

mleftparetolognormal(
  r = 0,
  truncation = 0,
  shape1 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE
)

rleftparetolognormal(n, shape1 = 1.5, meanlog = -0.5, sdlog = 0.5)
```

Arguments

`x, q`	vector of quantiles
`shape1, meanlog, sdlog`	Shape, mean and variance of the Left-Pareto Lognormal distribution respectively.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r \| X \leq y])$, otherwise, $P[X > x]$ $(E[x^r \| X > y])$
`p`	vector of probabilities
`r`	rth raw moment of the Pareto distribution
`truncation`	lower truncation parameter, defaults to xmin
`n`	number of observations

Details

Probability and Cumulative Distribution Function as provided by (Reed and Jorgensen 2004):

$f(x) = shape1\omega^{shape1-1}e^{-shape1meanlog+\frac{shape1^2sdlog^2}{2}}\Phi^c(\frac{ln\omega-meanlog+shape1sdlog^2}{sdlog})$,

$F_X(x) = \Phi(\frac{ln\omega-meanlog}{sdlog})-\omega^{shape1}e^{-shape1meanlog+\frac{shape1^2sdlog^2}{2}}\Phi^c(\frac{ln\omega-meanlog+shape1sdlog^2}{sdlog})$

The y-bounded r-th raw moment of the Let-Pareto Lognormal distribution equals:

$meanlog_y^r = -shape1e^{-shape1meanlog+\frac{shape1^2sdlog^2}{2}}\frac{y^{\sigma_s+shape1-1}}{\sigma_s+shape1-1}\Phi^c(\frac{lny-meanlog+shape1sdlog^2}{sdlog})$

$+\frac{shape1}{r+shape1}e^{\frac{r^2sdlog^2+2meanlogr}{2}}\Phi^c(\frac{lny-rsdlog^2+meanlog}{sdlog})$

Value

dleftparetolognormal gives the density, pleftparetolognormal gives the distribution function, qleftparetolognormal gives the quantile function, mleftparetolognormal gives the rth moment of the distribution and rleftparetolognormal generates random deviates.

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

References

Reed WJ, Jorgensen M (2004). "The Double Pareto-Lognormal Distribution–A New Parametric Model for Size Distributions." *Communications in Statistics - Theory and Methods*, **33**(8), 1733–1753.

Left-Pareto Lognormal density plot(x=seq(0,5,length.out=100),y=dleftparetolognormal(x=seq(0,5,length.out=100))) plot(x=seq(0,5,length.out=100),y=dleftparetolognormal(x=seq(0,5,length.out=100),shape1=1))

Left-Pareto Lognormal relates to the Lognormal if the shape parameter goes to infinity pleftparetolognormal(q=6,shape1=1 0.5,sdlog=0.5) plnorm(q=6,meanlog=-0.5,sdlog=0.5)

Demonstration of log functionality for probability and quantile function qleftparetolognormal(pleftparetolognormal(2,log.

The zeroth truncated moment is equivalent to the probability function pleftparetolognormal(2) mleftparetolognormal(truncation=2)

The (truncated) first moment is equivalent to the mean of a (truncated) random sample, #for large enough samples. x = rleftparetolognormal(1e5)

mean(x) mleftparetolognormal(r=1,lower.tail=FALSE)

sum(x[x>quantile(x,0.1)])/length(x) mleftparetolognormal(r=1,truncation=quantile(x,0.1),lower.tail=FALSE)

`leftparetolognormal.mle` *Left-Pareto Lognormal MLE*

Description

Maximum likelihood estimation of the parameters of the Left-Pareto Lognormal distribution.

Usage

```
leftparetolognormal.mle(
  x,
  lower = c(1e-10, 1e-10),
  upper = c(Inf, Inf),
  start = NULL
)
```

Arguments

`x`	data vector
`lower, upper`	Upper and lower bounds for the estimation procedure on the parameters c(shape1,sdlog), defaults to c(1e-10,1e-10) and c(Inf,Inf) respectively.
`start`	named vector with starting values, default to c(shape1=2,sdlog=sd(log(x)))

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

x = rleftparetolognormal(1e3)

Pareto fit with xmin set to the minium of the sample leftparetolognormal.mle(x=x)

`leftparetolognormal_plt` *Left-Pareto Lognormal coefficients of power-law transformed Left-Pareto Lognormal*

Description

Coefficients of a power-law transformed Left-Pareto Lognormal distribution

Usage

```
leftparetolognormal_plt(
  shape1 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  a = 1,
  b = 1,
  inv = FALSE
)
```

Arguments

`shape1, meanlog, sdlog`
Shapes, mean and variance of the Left-Pareto Lognormal distribution respectively.

`a, b` constant and power of power-law transformation, defaults to 1 and 1 respectively.

`inv` logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable y is Left-Pareto Lognormal distributed with mean meanlog and standard deviation sdlog, then the power-law transformed variable

$$y = ax^b$$

is Left-Pareto Lognormal distributed with $shape1 * b$, $\frac{meanlog - log(a)}{b}$, $\frac{sdlog}{b}$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Examples

```
## Comparing probabilites of power-law transformed transformed variables
pleftparetolognormal(3, shape1 = 1.5, meanlog = -0.5, sdlog = 0.5)
coeff <- leftparetolognormal_plt(shape1 = 1.5, meanlog = -0.5, sdlog = 0.5,
a = 5, b = 7)$coefficients
pleftparetolognormal(5 * 3^7, shape1 = coeff[["shape1"]], meanlog = coeff[["meanlog"]],
sdlog = coeff[["sdlog"]])

pleftparetolognormal(5 * 0.9^7, shape1 = 1.5, meanlog = -0.5, sdlog = 0.5)
coeff <- leftparetolognormal_plt(shape1 = 1.5, meanlog = -0.5, sdlog = 0.5, a = 5, b = 7,
inv = TRUE)$coefficients
pleftparetolognormal(0.9, shape1 = coeff[["shape1"]], meanlog = coeff[["meanlog"]],
 sdlog = coeff[["sdlog"]])
```

---

llr_vuong *Vuong's closeness test*

Description

Likelihood ratio test for model selection using the Kullback-Leibler information criterion (Vuong 1989)

Usage

```
llr_vuong(x, y, np.x, np.y, corr = c("none", "BIC", "AIC"))
```

Arguments

x, y	vector of log-likelihoods
np.x, np.y	Number of paremeters respectively
corr	type of correction for parameters, defaults to none.

Value

returns data frame with test statistic, p-value and character vector indicating the test outcome.

References

Vuong QH (1989). "Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses." *Econometrica*, **57**(2), 307–333.

Examples

```
x <- rlnorm(1e4, meanlog = -0.5, sdlog = 0.5)
pareto_fit <- combdist.mle(x = x, dist = "pareto")
pareto_loglike <- dcombdist(x = x, dist = "pareto", coeff = pareto_fit$coefficients, log = TRUE)
lnorm_fit <- combdist.mle(x = x, dist = "lnorm")
lnorm_loglike <- dcombdist(x = x, dist = "lnorm", coeff = lnorm_fit$coefficients, log = TRUE)

llr_vuong(x = pareto_loglike, y = lnorm_loglike, np.x = pareto_fit$np, np.y = lnorm_fit$np)

# BIC type parameter correction
llr_vuong(x = pareto_loglike, y = lnorm_loglike, np.x = pareto_fit$np, np.y = lnorm_fit$np,
corr = "BIC")

# AIC type parameter correction
llr_vuong(x = pareto_loglike, y = lnorm_loglike, np.x = pareto_fit$np, np.y = lnorm_fit$np,
corr = "AIC")
```

---

lnorm *The Lognormal distribution*

Description

Raw moments for the Lognormal distribution.

Usage

```
mlnorm(r = 0, truncation = 0, meanlog = -0.5, sdlog = 0.5, lower.tail = TRUE)
```

Arguments

r	rth raw moment of the distribution, defaults to 1.
truncation	lower truncation parameter, defaults to 0.
meanlog, sdlog	mean and standard deviation of the distribution on the log scale with default values of 0 and 1 respectively.
lower.tail	logical; if TRUE (default), moments are $E[x^r\|X \leq y]$, otherwise, $E[x^r\|X > y]$

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{1}{xVar\sqrt{2\pi}} e^{-(lnx-\mu)^2/2Var^2}, \qquad F_X(x) = \Phi(\frac{lnx-\mu}{Var})$$

The y-bounded r-th raw moment of the Lognormal distribution equals:

$$\mu_y^r = e^{\frac{r(rVar^2+2\mu)}{2}}[1 - \Phi(\frac{lny-(rVar^2+\mu)}{Var})]$$

Value

Provides the y-bounded, rth raw moment of the distribution.

Examples

```
## The zeroth truncated moment is equivalent to the probability function
plnorm(2, meanlog = -0.5, sdlog = 0.5)
mlnorm(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rlnorm(1e5, meanlog = -0.5, sdlog = 0.5)
mean(x)
mlnorm(r = 1, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mlnorm(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

---

lnorm_plt — *Log Normal coefficients of power-law transformed log normal*

Description

Coefficients of a power-law transformed log normal distribution

Usage

```
lnorm_plt(meanlog = 0, sdlog = 1, a = 1, b = 1, inv = FALSE)
```

Arguments

- `meanlog, sdlog` mean and standard deviation of the log normal distributed variable, defaults to 0 and 1 respectively.
- `a, b` constant and power of power-law transformation, defaults to 1 and 1 respectively.
- `inv` logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable y is log normally distributed with mean meanlog and standard deviation sdlog, then the power-law transformed variable

$$y = ax^b$$

is log normally distributed with mean $\frac{meanlog - ln(a)}{b}$ and standard deviation $\frac{sdlog}{b}$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables plnorm(3,meanlog=-0.5,sdlog=0.5) coeff = lnorm_plt(meanlog=-0.5,sdlog=0.5,a=5,b=7)$coefficients plnorm(5*3^7,meanlog=coeff[["meanlog"]]

plnorm(5*0.8^7,meanlog=-0.5,sdlog=0.5) coeff = lnorm_plt(meanlog=-0.5,sdlog=0.5,a=5,b=7,inv=TRUE)$coefficients plnorm(0.8,meanlog=coeff[["meanlog"]],sdlog=coeff[["sdlog"]])

Comparing the first moments and sample means of power-law transformed variables for large enough samples x = rlnorm(1e5,meanlog=-0.5,sdlog=0.5) coeff = lnorm_plt(meanlog=-0.5,sdlog=0.5,a=2,b=0.5)$coefficient y = rlnorm(1e5,meanlog=coeff[["meanlog"]],sdlog=coeff[["sdlog"]]) mean(2*x^0.5) mean(y) mlnorm(r=1,meanlog=coeff[["meanlog"]],sdlog=coeff[["sdlog"]],lower.tail=FALSE)

`nmad_test` *Normalized Absolute Deviation*

Description

Calculates the Normalized Absolute Deviation between the empirical moments and the moments of the provided distribution. Corresponds to the Kolmogorov-Smirnov test statistic for the zeroth moment.

Usage

```
nmad_test(
  x,
  r = 0,
  dist,
  prior = 1,
  coeff,
  stat = c("NULL", "max", "sum"),
  ...
)
```

Arguments

`x`	data vector
`r`	moment parameter
`dist`	character vector containing distribution
`prior`	named list of priors, defaults to 1
`coeff`	named list of coefficients
`stat`	character vector indicating which statistic should be calculated: none (NULL), the maximum deviation "max" or the sum of deviations "sum". Defaults to NULL.
`...`	Additional arguments can be passed to the parametric moment call.

Examples

```
x <- rlnorm(1e2, meanlog = -0.5, sdlog = 0.5)
nmad_test(x = x, r = 0, dist = "lnorm", coeff = c(meanlog = -0.5, sdlog = 0.5))
nmad_test(x = x, r = 0, dist = "lnorm", coeff = c(meanlog = -0.5, sdlog = 0.5), stat = "max")
nmad_test(x = x, r = 0, dist = "lnorm", coeff = c(meanlog = -0.5, sdlog = 0.5), stat = "sum")
```

---

pareto *The Pareto distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for the Pareto distribution.

Usage

```
dpareto(x, k = 2, xmin = 1, log = FALSE, na.rm = FALSE)

ppareto(q, k = 2, xmin = 1, lower.tail = TRUE, log.p = FALSE, na.rm = FALSE)
```

```
qpareto(p, k = 2, xmin = 1, lower.tail = TRUE, log.p = FALSE)

mpareto(r = 0, truncation = xmin, k = 2, xmin = 1, lower.tail = TRUE)

rpareto(n, k = 2, xmin = 1)
```

Arguments

x, q	vector of quantiles
xmin, k	Scale and shape of the Pareto distribution, defaults to 1 and 2 respectively.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
na.rm	Removes values that fall outside the support of the distribution
lower.tail	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ ($E[x^r \| X \leq y]$), otherwise, $P[X > x]$ ($E[x^r \| X > y]$)
p	vector of probabilities
r	rth raw moment of the Pareto distribution
truncation	lower truncation parameter, defaults to xmin
n	number of observations

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{kx_{min}^k}{x^{k+1}}, \qquad F_X(x) = 1 - (\frac{x_{min}}{x})^k$$

The y-bounded r-th raw moment of the Pareto distribution equals:

$$\mu_y^r = kx_{min}^k \frac{-y^{r-k}}{r-k}, \qquad k > r$$

Value

dpareto returns the density, ppareto the distribution function, qpareto the quantile function, mpareto the rth moment of the distribution and rpareto generates random deviates.

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

Examples

```
## Pareto density
plot(x = seq(1, 5, length.out = 100), y = dpareto(x = seq(1, 5, length.out = 100), k = 2, xmin = 1))

## Pareto relates to the exponential distribution available in the stats package
ppareto(q = 5, k = 2, xmin = 3)
pexp(q = log(5 / 3), rate = 2)
```

```
## Demonstration of log functionality for probability and quantile function
qpareto(ppareto(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
ppareto(2)
mpareto(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rpareto(1e5)

mean(x)
mpareto(r = 1, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mpareto(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

`pareto.mle` *Pareto MLE*

Description

Maximum likelihood estimation of the Pareto shape parameter using the Hill estimator.

Usage

```
pareto.mle(x, xmin = NULL, clauset = FALSE, q = 0, lower = 1e-10, upper = Inf)
```

Arguments

`x`	data vector
`xmin`	scale parameter of the Pareto distribution, set to min(x) if not provided
`clauset`	Indicator variable for calculating the scale parameter using the clauset method, overrides provided xmin
`q`	Percentage of data to search over (starting from the largest values), defaults to 0.
`lower, upper`	Lower and upper bounds to the estimated shape parameter, defaults to 1e-10 and Inf respectively

Details

The Hill estimator equals

$$\hat{k} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} log\frac{x_i}{x_{min}}}$$

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

Examples

```
x <- rpareto(1e3, k = 2, xmin = 2)

## Pareto fit with xmin set to the minium of the sample
pareto.mle(x = x)

## Pareto fit with xmin set to its real value
pareto.mle(x = x, xmin = 2)

## Pareto fit with xmin determined by the Clauset method
pareto.mle(x = x, clauset = TRUE)
```

---

pareto_plt *Pareto coefficients after power-law transformation*

Description

Coefficients of a power-law transformed Pareto distribution

Usage

```
pareto_plt(xmin = 1, k = 2, a = 1, b = 1, inv = FALSE)
```

Arguments

xmin, k	Scale and shape of the Pareto distribution, defaults to 1 and 2 respectively.
a, b	constant and power of power-law transformation, defaults to 1 and 1 respectively.
inv	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable x is Pareto-distributed with scale xmin and shape k, then the power-law transformed variable

$$y = ax^b$$

is Pareto distributed with scale $(\frac{xmin}{a})^{\frac{1}{b}}$ and shape $b * k$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Examples

```
## Comparing probabilites of power-law transformed transformed variables
ppareto(3, k = 2, xmin = 2)
coeff <- pareto_plt(xmin = 2, k = 2, a = 5, b = 7)$coefficients
ppareto(5 * 3^7, k = coeff[["k"]], xmin = coeff[["xmin"]])

ppareto(5 * 0.9^7, k = 2, xmin = 2)
coeff <- pareto_plt(xmin = 2, k = 2, a = 5, b = 7, inv = TRUE)$coefficients
ppareto(0.9, k = coeff[["k"]], xmin = coeff[["xmin"]])

## Comparing the first moments and sample means of power-law transformed variables for
#large enough samples
x <- rpareto(1e5, k = 2, xmin = 2)
coeff <- pareto_plt(xmin = 2, k = 2, a = 2, b = 0.5)$coefficients
y <- rpareto(1e5, k = coeff[["k"]], xmin = coeff[["xmin"]])
mean(2 * x^0.5)
mean(y)
mpareto(r = 1, k = coeff[["k"]], xmin = coeff[["xmin"]], lower.tail = FALSE)
```

`rightparetolognormal` *The Right-Pareto Lognormal distribution*

Description

Density, distribution function, quantile function and random generation for the Right-Pareto Lognormal distribution.

Usage

```
drightparetolognormal(
  x,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  log = FALSE
)

prightparetolognormal(
  q,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

qrightparetolognormal(
  p,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE,
  log.p = FALSE
)

mrightparetolognormal(
  r = 0,
  truncation = 0,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE
)

rrightparetolognormal(
  n,
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  lower.tail = TRUE
)
```

Arguments

`x, q` vector of quantiles

shape2, meanlog, sdlog
: Shape, mean and variance of the Right-Pareto Lognormal distribution respectively.

log, log.p
: logical; if TRUE, probabilities p are given as log(p).

lower.tail
: logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r|X \leq y])$, otherwise, $P[X > x]$ $(E[x^r|X > y])$

p
: vector of probabilities

r
: rth raw moment of the Pareto distribution

truncation
: lower truncation parameter, defaults to xmin

n
: number of observations

Details

Probability and Cumulative Distribution Function as provided by (Reed and Jorgensen 2004):

$f(x) = shape2\omega^{-shape2-1}e^{shape2meanlog+\frac{shape2^2sdlog^2}{2}}\Phi(\frac{lnx-meanlog-shape2sdlog^2}{sdlog}),$
$F_X(x) = \Phi(\frac{lnx-meanlog}{sdlog}) - \omega^{-shape2}e^{shape2meanlog+\frac{shape2^2sdlog^2}{2}}\Phi(\frac{lnx-meanlog-shape2sdlog^2}{sdlog})$

The y-bounded r-th raw moment of the Right-Pareto Lognormal distribution equals:

$meanlog_y^r = -shape2e^{shape2meanlog+\frac{shape2^2sdlog^2}{2}}\frac{y^{\sigma_s-shape2-1}}{\sigma_s-shape2-1}\Phi(\frac{lny-meanlog-shape2sdlog^2}{sdlog})$
$-\frac{shape2}{r-shape2}e^{\frac{r^2sdlog^2+2meanlogr}{2}}\Phi^c(\frac{lny-rsdlog^2+meanlog}{sdlog}), \qquad shape2 > r$

Value

drightparetolognormal gives the density, prightparetolognormal gives the distribution function, qrightparetolognormal gives the quantile function, mrightparetolognormal gives the rth moment of the distribution and rrightparetolognormal generates random deviates.

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

References

Reed WJ, Jorgensen M (2004). "The Double Pareto-Lognormal Distribution–A New Parametric Model for Size Distributions." *Communications in Statistics - Theory and Methods*, **33**(8), 1733–1753.

Examples

```
## Right-Pareto Lognormal density
plot(x = seq(0, 5, length.out = 100), y = drightparetolognormal(x = seq(0, 5, length.out = 100)))
plot(x = seq(0, 5, length.out = 100), y = drightparetolognormal(x = seq(0, 5, length.out = 100),
shape2 = 1))

## Right-Pareto Lognormal relates to the Lognormal if the shape parameter goes to infinity
prightparetolognormal(q = 6, shape2 = 1e20, meanlog = -0.5, sdlog = 0.5)
plnorm(q = 6, meanlog = -0.5, sdlog = 0.5)
```

```
## Demonstration of log functionality for probability and quantile function
qrightparetolognormal(prightparetolognormal(2, log.p = TRUE), log.p = TRUE)

## The zeroth truncated moment is equivalent to the probability function
prightparetolognormal(2)
mrightparetolognormal(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rrightparetolognormal(1e5, shape2 = 3)

mean(x)
mrightparetolognormal(r = 1, shape2 = 3, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mrightparetolognormal(r = 1, shape2 = 3, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

`rightparetolognormal.mle` *Right-Pareto Lognormal MLE*

Description

Maximum likelihood estimation of the parameters of the Right-Pareto Lognormal distribution.

Usage

```
rightparetolognormal.mle(
  x,
  lower = c(1e-10, 1e-10),
  upper = c(Inf, Inf),
  start = NULL
)
```

Arguments

`x`	data vector
`lower, upper`	Upper and lower bounds for the estimation procedure on the parameters c(shape2,sdlog), defaults to c(1e-10,1e-10) and c(Inf,Inf) respectively.
`start`	named vector with starting values, default to c(shape2=2,sdlog=sd(log(x)))

Value

Returns a named list containing a

coefficients Named vector of coefficients

convergence logical indicator of convergence

n Length of the fitted data vector

np Nr. of coefficients

Examples

```
x <- rrightparetolognormal(1e3)

## Pareto fit with xmin set to the minium of the sample
rightparetolognormal.mle(x = x)
```

---

rightparetolognormal_plt   *Right-Pareto Lognormal coefficients of power-law transformed Right-Pareto Lognormal*

Description

Coefficients of a power-law transformed Right-Pareto Lognormal distribution

Usage

```
rightparetolognormal_plt(
  shape2 = 1.5,
  meanlog = -0.5,
  sdlog = 0.5,
  a = 1,
  b = 1,
  inv = FALSE
)
```

Arguments

shape2, meanlog, sdlog	Shapes, mean and variance of the Right-Pareto Lognormal distribution respectively.
a, b	constant and power of power-law transformation, defaults to 1 and 1 respectively.
inv	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable y is Right-Pareto Lognormal distributed with mean meanlog and standard deviation sdlog, then the power-law transformed variable

$$y = ax^b$$

is Right-Pareto Lognormal distributed with $\frac{meanlog-log(a)}{b}, \frac{sdlog}{b}, shape2 * b$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables prightparetolognormal(3, shape2 = 3, meanlog = -0.5, sdlog = 0.5) coeff = rightparetolognormal_plt(shape2 = 3, meanlog = -0.5, sdlog = 0.5,a=5,b=7)$coefficients prightparetolognormal(5*3^7,shape2=coeff[["shape2"]],meanlog=coeff[["mean

prightparetolognormal(5*0.9^7,shape2 = 3, meanlog = -0.5, sdlog = 0.5) coeff = rightparetolognormal_plt(shape2 = 3, meanlog = -0.5, sdlog = 0.5,a=5,b=7, inv=TRUE)$coefficients prightparetolognormal(0.9,shape2=coeff[["shape2"]],meanlog=coeff[["meanlog"]],sdlog=coeff[["sdlog"]])

`truncdist` *Truncated distribution*

Description

Density, distribution function, quantile function, raw moments and random generation for a truncated distribution.

Usage

```
dtruncdist(
  x,
  dist = c("lnormtrunc"),
  coeff = list(meanlog = 0, sdlog = 1),
  lowertrunc = 0,
  uppertrunc = Inf,
  log = FALSE
)

ptruncdist(
  q,
  dist = c("lnormtrunc"),
  coeff = list(meanlog = 0, sdlog = 1),
  lowertrunc = 0,
  uppertrunc = Inf,
  log.p = FALSE,
  lower.tail = TRUE
)

qtruncdist(
  p,
  dist = c("lnormtrunc"),
  coeff = list(meanlog = 0, sdlog = 1),
  lowertrunc = 0,
  uppertrunc = Inf,
```

```
  lower.tail = TRUE,
  log.p = FALSE
)

mtruncdist(
  r,
  truncation = 0,
  dist = c("lnormtrunc"),
  coeff = list(meanlog = 0, sdlog = 1),
  lowertrunc = 0,
  uppertrunc = Inf,
  lower.tail = TRUE
)

rtruncdist(
  n,
  dist = c("lnormtrunc"),
  coeff = list(meanlog = 0, sdlog = 1),
  lowertrunc = 0,
  uppertrunc = Inf
)
```

Arguments

`x, q`	vector of quantiles
`dist`	distribution to be truncated, defaults to lnorm
`coeff`	list of parameters for the truncated distribution, defaults to list(meanlog=0,sdlog=1)
`lowertrunc, uppertrunc`	lowertrunc- and uppertrunc truncation points, defaults to 0 and Inf respectively
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities (moments) are $P[X \leq x]$ $(E[x^r \| X \leq y])$, otherwise, $P[X > x]$ $(E[x^r \| X > y])$
`p`	vector of probabilities
`r`	rth raw moment of the distribution
`truncation`	lowertrunc truncation parameter, defaults to 0.
`n`	number of observations

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{g(x)}{F(uppertrunc) - F(lowertrunc)}, \qquad F_X(x) = \frac{F(x) - F(lowertrunc)}{F(uppertrunc) - F(lowertrunc)}$$

Value

dtruncdist gives the density, ptruncdist gives the distribution function, qtruncdist gives the quantile function, mtruncdist gives the rth moment of the distribution and rtruncdist generates random deviates.

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

Examples

```
## Truncated lognormal density
plot(x = seq(0.5, 3, length.out = 100), y = dtruncdist(x = seq(0.5, 5, length.out = 100),
dist = "lnorm", coeff = list(meanlog = 0.5, sdlog = 0.5), lowertrunc = 0.5, uppertrunc = 5))
lines(x = seq(0, 6, length.out = 100), y = dlnorm(x = seq(0, 6, length.out = 100),
meanlog = 0.5, sdlog = 0.5))

# Compare quantities
dtruncdist(0.5)
dlnorm(0.5)
dtruncdist(0.5, lowertrunc = 0.5, uppertrunc = 3)

ptruncdist(2)
plnorm(2)
ptruncdist(2, lowertrunc = 0.5, uppertrunc = 3)

qtruncdist(0.25)
qlnorm(0.25)
qtruncdist(0.25, lowertrunc = 0.5, uppertrunc = 3)

mtruncdist(r = 0, truncation = 2)
mlnorm(r = 0, truncation = 2, meanlog = 0, sdlog = 1)
mtruncdist(r = 0, truncation = 2, lowertrunc = 0.5, uppertrunc = 3)

mtruncdist(r = 1, truncation = 2)
mlnorm(r = 1, truncation = 2, meanlog = 0, sdlog = 1)
mtruncdist(r = 1, truncation = 2, lowertrunc = 0.5, uppertrunc = 3)
```

---

weibull *The Weibull distribution*

Description

Raw moments for the Weibull distribution.

Usage

```
mweibull(r = 0, truncation = 0, shape = 2, scale = 1, lower.tail = TRUE)
```

Arguments

`r`	rth raw moment of the distribution, defaults to 1.
`truncation`	lower truncation parameter, defaults to 0.
`shape, scale`	shape and scale of the distribution with default values of 2 and 1 respectively.
`lower.tail`	logical; if TRUE (default), moments are $E[x^r\|X \leq y]$, otherwise, $E[x^r\|X > y]$

Details

Probability and Cumulative Distribution Function:

$$f(x) = \frac{shape}{scale}(\frac{\omega}{scale})^{shape-1}e^{-(\frac{\omega}{scale})^shape}, \qquad F_X(x) = 1 - e^{-(\frac{\omega}{scale})^shape}$$

The y-bounded r-th raw moment of the distribution equals:

$$\mu_y^r = scale^r\Gamma(\frac{r}{shape} + 1, (\frac{y}{scale})^shape)$$

where $\Gamma(,)$ denotes the upper incomplete gamma function.

Value

returns the truncated rth raw moment of the distribution.

Examples

```
## The zeroth truncated moment is equivalent to the probability function
pweibull(2, shape = 2, scale = 1)
mweibull(truncation = 2)

## The (truncated) first moment is equivalent to the mean of a (truncated) random sample,
#for large enough samples.
x <- rweibull(1e5, shape = 2, scale = 1)
mean(x)
mweibull(r = 1, lower.tail = FALSE)

sum(x[x > quantile(x, 0.1)]) / length(x)
mweibull(r = 1, truncation = quantile(x, 0.1), lower.tail = FALSE)
```

`weibull_plt` *Weibull coefficients of power-law transformed Weibull*

Description

Coefficients of a power-law transformed Weibull distribution

Usage

```
weibull_plt(scale = 1, shape = 2, a = 1, b = 1, inv = FALSE)
```

Arguments

`shape, scale`	shape and scale of the distribution with default values of 2 and 1 respectively.
`a, b`	constant and power of power-law transformation, defaults to 1 and 1 respectively.
`inv`	logical indicating whether coefficients of the outcome variable of the power-law transformation should be returned (FALSE) or whether coefficients of the input variable being power-law transformed should be returned (TRUE). Defaults to FALSE.

Details

If the random variable y is Weibull distributed with mean meanlog and standard deviation sdlog, then the power-law transformed variable

$$y = ax^b$$

is Weibull distributed with scale $(\frac{scale}{a})^{\frac{1}{b}}$ and shape $b * shape$.

Value

Returns a named list containing

coefficients Named vector of coefficients

Comparing probabilites of power-law transformed transformed variables pweibull(3,shape=2,scale=1) coeff = weibull_plt(shape=2,scale=1,a=5,b=7)$coefficients pweibull(5*3^7,shape=coeff[["shape"]],scale=coeff[["scale"]])

pweibull(5*0.8^7,shape=2,scale=1) coeff = weibull_plt(shape=2,scale=1,a=5,b=7,inv=TRUE)$coefficients pweibull(0.8,shape=coeff[["shape"]],scale=coeff[["scale"]])

Comparing the first moments and sample means of power-law transformed variables for large enough samples x = rweibull(1e5,shape=2,scale=1) coeff = weibull_plt(shape=2,scale=1,a=2,b=0.5)$coefficients y = rweibull(1e5,shape=coeff[["shape"]],scale=coeff[["scale"]]) mean(2*x^0.5) mean(y) mweibull(r=1,shape=coeff[["shape'

Index