Package 'mistr'

February 4, 2020

Type Package

Title Mixture and Composite Distributions

Version 0.0.3

Depends R (>= 3.0.1)

Maintainer Lukas Sablica <lsablica@wu.ac.at>

Description A flexible computational framework for mixture distributions with the focus on the composite models.

License GPL-3

Imports bbmle, stats, graphics, grDevices

Suggests ggplot2, grid, knitr, rmarkdown, pinp

LazyData true

Encoding UTF-8

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Lukas Sablica [aut, cre] (<https://orcid.org/0000-0001-9166-4563>),
Kurt Hornik [aut] (<https://orcid.org/0000-0003-4198-9911>)

Repository CRAN

Date/Publication 2020-02-04 10:40:02 UTC

R topics documented:

`mistr-package` *mistr: A Computational Framework for Univariate Mixture and Composite Distributions*

Description

A system offering object oriented handling of univariate distributions with focus on composite models.

Author(s)

Lukas Sablica, <lsablica@wu.ac.at>

Kurt Hornik, <Kurt.Hornik@wu.ac.at>

Maintainer: Lukas Sablica, <lsablica@wu.ac.at>

autoplot.comp_fit *Autoplot of Fitted Distributions Using ggplot2*

Description

The functions plot the CDF, PDF and QQ-plot of a fitted distribution object together with the empirical values.

Usage

```
autoplot.comp_fit(x, which = "all", layout = matrix(c(1, 2, 1, 3), nrow
  = 2), empir_color = "#F9D607", empir_alpha = 0.4, ...)
```

Arguments

x	distribution object.
which	whether to plot only CDF, PDF, qq or all three, default: 'all'.
layout	layout of plots, default: matrix(c(1, 2, 1, 3), nrow = 2).
empir_color	color of empirical data, default: '#F9D607'.
empir_alpha	alpha of empirical data, default: 0.4.
...	further arguments to be passed.

Value

ggplot object if `which = "cdf"` or `which = "pdf"` or `which = "qq"`. If all are plotted, the plots are merged using `multiplot()` function and a list with all plots is invisibly returned.

See Also

plotgg

autoplot.dist *Autoplot of Distributions Using ggplot2*

Description

The function `autoplot` plots the CDF and PDF of a given distribution object.

Usage

```
autoplot.dist(x, which = "all", ncols = 2, ...)
```

Arguments

x	distribution object.
which	whether to plot only CDF, PDF or both, default: 'all'.
ncols	in how many columns should the plots be merged, default: 2.
...	further arguments to be passed.

Details

The function is a wrapper of the internal plotting function plotgg. For more details see plotgg.

Value

ggplot object if which = "cdf" or which = "pdf". If both are plotted, the plots are merged using multiplot() function and a list with both plots is invisibly returned.

See Also

plotgg

Examples

```
## Not run:
N <- normdist()
autoplot(N)

# manipulating cdf plot
B <- binomdist(12, 0.5)
autoplot(-3*B, which = "cdf", xlim1 = c(-30, -10))
# manipulating pdf plot
autoplot(-3*B, which = "pdf", xlim2 = c(-30, -10))

## End(Not run)
```

---

betadist *Creates an Object Representing Beta Distribution*

Description

The function creates an object which represents the beta distribution.

Usage

```
betadist(shape1 = 2, shape2 = 2)
```

Arguments

shape1	shape parameter, default: 2.
shape2	shape parameter, default: 2.

Details

See Beta.

Value

Object of class betadist.

See Also

Beta

Examples

```
B <- betadist(2, 2)
d(B, c(2, 3, 4, NA))
r(B, 5)
```

---

binomdist *Creates an Object Representing Binomial Distribution.*

Description

The function creates an object which represents the binomial distribution.

Usage

```
binomdist(size = 10, prob = 0.5)
```

Arguments

size	size parameter, default: 10.
prob	probability parameter, default: 0.5.

Details

See Binomial.

Value

Object of class binomdist.

See Also

Binomial

Examples

```
B <- binomdist(10, 0.4)
d(B, c(2, 3, 4, NA))
r(B, 5)
```

breakpoints *Extract Model Breakpoints*

Description

breakpoints is a generic function which extracts breakpoints from mistr composite distribution objects.

Usage

```
breakpoints(O)

## S3 method for class 'compdist'
breakpoints(O)

## S3 method for class 'trans_compdist'
breakpoints(O)

## S3 method for class 'comp_fit'
breakpoints(O)
```

Arguments

O	an object for which the extraction of model breakpoints is meaningful.

Value

Vector of extracted breakpoints form object.

See Also

parameters, weights

Examples

```
N <- normdist(1, 3)
C <- cauchydist()

CC <- compdist(N, C, weights = c(0.5, 0.5), breakpoints = 1)
breakpoints(CC)
```

Burr *The Burr Distribution*

Description

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

Usage

```
dburr(x, shape1, shape2, log = FALSE)

pburr(q, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

qburr(p, shape1, shape2, lower.tail = TRUE, log.p = FALSE)

rburr(n, shape1, shape2)
```

Arguments

x, q	vector of quantiles.
shape1	shape parameter.
shape2	shape parameter.
log, log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
p	vector of probabilities.
n	number of observations.

Details

The Burr distribution function with shape1 parameter c and shape2 parameter k has density given by

$$f(x) = ckx^(c-1)/(1+x^c)^(k+1)$$

for $x > 0$. The cumulative distribution function is

$$F(x) = 1 - (1+x^c)^-k$$

on $x > 0$.

See https://en.wikipedia.org/wiki/Burr_distribution for more details.

Value

dburr gives the density, pburr gives the distribution function, qburr gives the quantile function, and rburr generates random deviates.

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

See Also

burrdist

Examples

```
dburr(seq(1, 5), 2, 2)
qburr(pburr(seq(1, 5), 2, 2), 2 ,2)
rburr(5, 2, 2)
```

---

burrdist *Creates an Object Representing Burr Distribution*

Description

The function creates an object which represents the Burr distribution.

Usage

```
burrdist(shape1 = 2, shape2 = 2)
```

Arguments

shape1	shape parameter, default: 2.
shape2	shape parameter, default: 2.

Details

See Burr.

Value

Object of class burrdist.

See Also

Burr

Examples

```
B <- burrdist(2, 2)
d(B, c(2, 3, 4, NA))
r(B, 5)
```

cauchydist *Creates an Object Representing Cauchy Distribution.*

Description

The function creates an object which represents the Cauchy distribution.

Usage

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

Arguments

location	location parameter, default: 0.
scale	scale parameter, default: 1.

Details

See Cauchy.

Value

Object of class cauchydist.

See Also

Cauchy

Examples

```
C <- cauchydist(0, 1)
d(C, c(2, 3, 4, NA))
r(C, 5)
```

chisqdist *Creates an Object Representing Chi-Squared Distribution*

Description

The function creates an object which represents the chi-squared distribution.

Usage

```
chisqdist(df = 2)
```

Arguments

df	degrees of freedom parameter, default: 2.

Details

See Chisquare.

Value

Object of class chisqdist.

See Also

Chisquare

Examples

```
C <- chisqdist(2)
d(C, c(2, 3, 4, NA))
r(C, 5)
```

---

compdist *Creates an Object Representing Composite Distribution*

Description

compdist creates an object which represents the composite distribution.

Usage

```
compdist(..., weights, breakpoints, break.spec, all.left = FALSE)

## S3 method for class 'dist'
compdist(..., weights, breakpoints, break.spec,
  all.left = FALSE)

## Default S3 method:
compdist(dist, params, weights, breakpoints, break.spec,
  all.left = FALSE, ...)
```

Arguments

...	distribution objects.
weights	vector of weights for the components.
breakpoints	vector of breakpoints for the composite models, first and last breakpoints ($-\infty$, ∞) are assumed to be given, and should not be specified.

break.spec	vector of breakpoints specifications with values "L" or "R", breakpoints specifications corresponding to $-\infty$ and ∞ should not be specified.
all.left	if TRUE, all break.spec are set to "L", default: FALSE.
dist	vector of distribution names.
params	list of parameters.

Details

A CDF of a composite distribution function is

$$F(A) = \sum w_i F_i(A|B_i)$$

, where w_i is the weight of the i-th component, $F_i()$ is the CDF of the i-th component and B_i is the interval specified by the breakpoints. Clearly, the composite models are a specific case of the mixture models, where the corresponding probability distribution functions are truncated to some disjoint support.

The objects can be specified in two ways, either the user may enter objects representing distributions or a vector of names and list of parameters. See the examples below.

The argument break.spec defines if the breakpoint should be included to the distribution to the right ("R") or to the left ("L") of the breakpoint. This feature is of course useful only in the case where at least one of the adjacent components is discrete. By default the intervals are left-closed (all break.spec values are "R").

The function permits to use the same breakpoint twice. This possibility allows to define a partition on a singleton, and hence to create a mass of probability. If this feature is used, the break.spec needs to be specified with "R" and "L", for the first and the second identical breakpoints, respectively, or not set at all.

Value

Object of class compdist.

See Also

mixdist

Examples

```
# using the objects
C <- compdist(normdist(1, 3), expdist(4), weights = c(0.7, 0.3), breakpoints = 2)
C

# using the names and parameters
C2 <- compdist(c("norm","exp"), list(c(mean = 1, sd = 3), c(rate = 4)),
              weights = c(0.7, 0.3), breakpoints = 2)
C2

# more complicated model where break.spec is useful
C3 <- compdist(-GPDdist(1,0.15,0.7), normdist(-1,1), binomdist(5,0.5),
              geomdist(0.3) + 2, weights = c(0.075, 0.425, 0.425, 0.075),
```

```
              breakpoints = c(-2.5, 0,3), break.spec = c("L", "R", "R"))
C3

# same breakpoint twice
C4 <-  compdist(-expdist(2),poisdist(),expdist(2),
                weights = c(0.25, 0.5, 0.25), breakpoints = c(0, 0))
C4
```

`d.compdist` *Density Function*

Description

d is a generic function that evaluates the density function of a distribution object at given values.

Usage

```
## S3 method for class 'compdist'
d(O, x, log = FALSE)

## S3 method for class 'trans_compdist'
d(O, x, log = FALSE)

## S3 method for class 'mixdist'
d(O, x, log = FALSE)

## S3 method for class 'trans_mixdist'
d(O, x, log = FALSE)

d(O, x, log = FALSE)

## S3 method for class 'standist'
d(O, x, log = FALSE)

## S3 method for class 'trans_contdist'
d(O, x, log = FALSE)

## S3 method for class 'trans_discrdist'
d(O, x, log = FALSE)
```

Arguments

O	distribution object.
x	vector of quantiles.
log	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.

Details

Methods of d function evaluates any offered distribution from the package mistr. The function makes use of the d[sufix] functions as dnorm or dbeta and thus, if a new distribution is added, these functions must be reachable through the search path.

Value

Vector of computed results.

Examples

```
N <- normdist(1, 3)
d(N, c(NA, 1, 3, 5))

C <- cauchydist()
M <- mixdist(N, C, weights = c(0.5, 0.5))
d(M, c(NA, 1, 3, 5))

CC <- compdist(N, C, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*C+5
d(CCC, c(NA, 1, 3, 5))
```

---

distribution *Extract Distribution of Fitted Model*

Description

distribution is a generic function which extracts the distribution with fitted parameters from fitted objects.

Usage

```
distribution(O)

## S3 method for class 'comp_fit'
distribution(O)
```

Arguments

O	an object for which the extraction of distribution is meaningful.

Value

Object representing the distribution.

`Distribution_autoplot` *Autoplot of Distributions*

Description

The functions plot the CDF and PDF of a given distribution object.

Usage

```
## S3 method for class 'compdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
       lty1 = 1, lty2 = 1, lwd1 = 2, lwd2 = 2, lty_abline = 3, mtext_cex = 1, ...)

## S3 method for class 'trans_compdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
       lty1 = 1, lty2 = 1, lwd1 = 2, lwd2 = 2, lty_abline = 3, mtext_cex = 1, ...)

## S3 method for class 'contdist'
plot(x, which = "all", pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = NULL, lty2 = NULL, lwd1 = NULL, lwd2 = NULL, ...)

## S3 method for class 'trans_contdist'
plot(x, which = "all", pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = NULL, lty2 = NULL, lwd1 = NULL, lwd2 = NULL, ...)

## S3 method for class 'discrdist'
plot(x, which = "all", col = "#122e94",
        xlim1 = q(x,c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PMF", type1 = NULL, type2 = NULL,
        lty1 = NULL, lty2 = NULL, lwd1 = NULL, lwd2 = NULL, ...)

## S3 method for class 'trans_discrdist'
plot(x, which = "all", col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
```

```
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PMF", type1 = "p", type2 = "p",
        lty1 = NULL, lty2 = NULL, lwd1 = NULL, lwd2 = NULL, ...)

## S3 method for class 'contmixdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 2, lwd2 = 2, ...)

## S3 method for class 'trans_contmixdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
        xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 2, lwd2 = 2, ...)

## S3 method for class 'discrmixdist'
plot(x, which = "all", only_mix = FALSE,
        pp1 = 1000, pp2 = 2 * (diff(xlim2)), col = "#122e94",
       xlim1 = q(x,c(0.01, 0.99)), ylim1 = c(0, 1), xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PMF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 3, lwd2 = 3, ...)

## S3 method for class 'trans_discrmixdist'
plot(x, which = "all", only_mix = FALSE,
        pp1 = 1000, pp2 = 2 * (diff(xlim2)), col = "#122e94",
       xlim1 = q(x,c(0.01, 0.99)), ylim1 = c(0, 1), xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PMF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 3, lwd2 = 3, ...)

## S3 method for class 'contdiscrmixdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
       xlim1 = q(x, c(0.01, 0.99)), ylim1 = c(0, 1), xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 2, lwd2 = 2, ...)

## S3 method for class 'trans_contdiscrmixdist'
plot(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#122e94",
       xlim1 = q(x, c(0.01, 0.99)), ylim1 = c(0, 1), xlim2 = xlim1, ylim2 = NULL,
        xlab1 = "x", ylab1 = "P(X \u2264 x)", xlab2 = "x", ylab2 = "P(X = x)",
        main1 = "CDF", main2 = "PDF", type1 = "l", type2 = "l",
        lty1 = 3, lty2 = 3, lwd1 = 2, lwd2 = 2, ...)
```

Arguments

x	distribution object.
which	whether to plot only CDF, PDF or both, default: 'all'.
only_mix	whether to plot only mixture/composite model and not also the components, default: FALSE.
pp1	number of points at which CDF is evaluated, default: 1000.
pp2	number of points at which PDF is evaluated, default: 1000.
col	color used in plot, default: '#122e94'.
xlim1	xlim of CDF plot, default: q(x, c(0.01, 0.99)).
ylim1	ylim of CDF plot, default: NULL.
xlim2	xlim of PDF plot, default: xlim1.
ylim2	ylim of PDF plot, default: NULL.
xlab1	xlab of CDF plot, default: 'x'.
ylab1	ylab of CDF plot, default: 'P(X <U+2264> x)'.
xlab2	xlab of PDF plot, default: 'x'.
ylab2	ylab of PDF plot, default: 'P(X = x)'.
main1	title of CDF plot, default: 'CDF'.
main2	title of PDF plot, default: 'PDF'/'PMF'.
type1	type of CDF plot.
type2	type of PDF plot.
lty1	lty used in CDF plot.
lty2	lty used in PDF plot.
lwd1	lwd used in CDF plot.
lwd2	lwd used in PDF plot.
lty_abline	lty of abline if ablines are part of plot (composite and discrete distributions).
mtext_cex	cex parameter for mtexts used in the plots of composite distributions, default: 1.
...	further arguments to be passed.

Examples

```
N <- normdist()
plot(N)

# manipulating cdf plot
B <- binomdist(12, 0.5)
plot(-3*B, which = "cdf", xlim1 = c(-30, -10))
# manipulating pdf plot
plot(-3*B, which = "pdf", xlim1 = c(-30, -10))
```

`Distribution_summary` *Displays a Useful Description of a Distribution Object*

Description

Displays a useful description of a distribution object from `mistr`.

Usage

```
## S3 method for class 'standist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)

## S3 method for class 'trans_standist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)

## S3 method for class 'mixdist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)

## S3 method for class 'trans_mixdist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)

## S3 method for class 'compdist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)

## S3 method for class 'trans_compdist'
summary(object, level = 1, space = 2,
  additional_list, truncation, ...)
```

Arguments

`object`	distribution object to summarize.
`level`	adds 3*(level-1) spaces before the print, default: 1.
`space`	number of blank lines between outputs, default: 2.
`additional_list, truncation, ...`	additional information that may be passed to summary.

Details

`summary` prints useful description of a distribution object. This feature might be useful when working with a more complicated distribution that contains mixture and composite distributions as components and the print function does not offer enough information.

Arguments `level`, `additional_list` and truncation are present for recursive usage that is done for more complicated models automatically by the function.

`Distribution_transformation`

Transformation of a Distribution Object

Description

The methods for arithmetic operators `+,-,*,/,^,log,exp,sqrt`, which perform a transformation of a given random variable.

Usage

```
## S3 method for class 'univdist'
e1 + e2 = NULL

## S3 method for class 'trans_univdist'
e1 + e2 = NULL

## S3 method for class 'univdist'
e1 * e2

## S3 method for class 'trans_univdist'
e1 * e2

## S3 method for class 'dist'
e1 / e2

## S3 method for class 'dist'
e1 - e2 = NULL

## S3 method for class 'dist'
sqrt(x)

## S3 method for class 'univdist'
log(x, base = exp(1))

## S3 method for class 'trans_univdist'
log(x, base = exp(1))

## S3 method for class 'univdist'
exp(x)

## S3 method for class 'trans_univdist'
exp(x)
```

```
## S3 method for class 'univdist'
e1 ^ e2

## S3 method for class 'trans_univdist'
e1 ^ e2

## S3 method for class 'normdist'
e1 + e2

## S3 method for class 'normdist'
e1 * e2

## S3 method for class 'normdist'
exp(x)

## S3 method for class 'expdist'
e1 * e2

## S3 method for class 'expdist'
e1 ^ e2

## S3 method for class 'unifdist'
e1 + e2

## S3 method for class 'unifdist'
e1 * e2

## S3 method for class 'tdist'
e1 ^ e2

## S3 method for class 'fdist'
e1 ^ e2

## S3 method for class 'betadist'
e1 - e2 = NULL

## S3 method for class 'binomdist'
e1 - e2 = NULL

## S3 method for class 'gammadist'
e1 * e2

## S3 method for class 'cauchydist'
e1 + e2

## S3 method for class 'cauchydist'
e1 * e2
```

```
## S3 method for class 'cauchydist'
e1 ^ e2

## S3 method for class 'lnormdist'
e1 * e2

## S3 method for class 'lnormdist'
log(x, base = exp(1))

## S3 method for class 'lnormdist'
e1 ^ e2

## S3 method for class 'weibulldist'
e1 * e2

## S3 method for class 'gumbeldist'
e1 + e2

## S3 method for class 'gumbeldist'
e1 * e2

## S3 method for class 'frechetdist'
e1 + e2

## S3 method for class 'frechetdist'
e1 * e2

## S3 method for class 'paretodist'
e1 * e2

## S3 method for class 'GPDdist'
e1 + e2

## S3 method for class 'GPDdist'
e1 * e2
```

Arguments

e1	distribution object or numeric of length one.
e2	distribution object or numeric of length one.
x	distribution object.
base	a positive number: the base with respect to which logarithms are computed.

Details

The offered arithmetic operators `+,-,*,/,^,log,exp,sqrt` create an object that represents transformed random variable.

The functions, using the expressions manipulation, prepare expressions for transformation, inverse transformation, derivative of the inverse transformation and print. These expressions are then used later when the distribution is evaluated.

The transformation framework also keeps track on history of the transformations and so is able to recognize some inverse transformations of previous transformations or update the last transformation. Additionally, the methods are able to recognize some invariant and direct transformations, and so rather change the parameters or distribution family than to loose this information.

Value

Object representing a transformed random variable.

Examples

```
# transformation
B  <- binomdist(10, 0.3)
B2 <- - 3*log(B)
B2

# invariant transformation
N  <- normdist(1, 3)
N2 <- - 3*N + 5
N2

# direct transformation
N3 <- exp(N2)
N3

# recognize inverse
B3 <- exp(B2/-3)
B3
# update
B4 <- B + 5
B4 + 3
```

`expdist` *Creates an Object Representing Exponential Distribution*

Description

The function creates an object which represents the exponential distribution.

Usage

```
expdist(rate = 1)
```

Arguments

`rate`	rate parameter, default: 1.

Details

See Exponential.

Value

Object of class expdist.

See Also

Exponential

Examples

```
E <- expdist(1)
d(E, c(2, 3, 4, NA))
r(E, 5)
```

---

fdist *Creates an Object Representing F Distribution*

Description

The function creates an object which represents the F distribution.

Usage

```
fdist(df1 = 2, df2 = 2)
```

Arguments

df1	degrees of freedom parameter, default: 2.
df2	degrees of freedom parameter, default: 2.

Details

See FDist.

Value

Object of class fdist.

See Also

FDist

Examples

```
f <- fdist(2, 2)
d(f, c(2, 3, 4, NA))
r(f, 5)
```

Frechet *The Frechet Distribution*

Description

Density, distribution function, quantile function and random generation for the Frechet distribution with location, scale and shape parameters.

Usage

```
dfrechet(x, loc = 0, scale = 1, shape = 1, log = FALSE)

pfrechet(q, loc = 0, scale = 1, shape = 1, lower.tail = TRUE,
  log.p = FALSE)

qfrechet(p, loc = 0, scale = 1, shape = 1, lower.tail = TRUE,
  log.p = FALSE)

rfrechet(n, loc = 0, scale = 1, shape = 1)
```

Arguments

x, q	vector of quantiles.
loc	location parameter.
scale	scale parameter.
shape	shape parameter.
log, log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
p	vector of probabilities.
n	number of observations.

Details

The Frechet distribution function with location parameter m, scale parameter s and shape parameter α has density given by

$$f(x) = \alpha/sz^{(} - \alpha - 1)e^{-}z^{-}\alpha$$

for $x > m$, where $z = (x - m)/s$. The cumulative distribution function is

$$F(x) = e^{-}z^{-}\alpha$$

for $x > m$, with z as stated above.

See https://en.wikipedia.org/wiki/Frechet_distribution for more details.

Value

`dfrechet` gives the density, `pfrechet` gives the distribution function, `qfrechet` gives the quantile function, and `rfrechet` generates random deviates.

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

See Also

`frechetdist`

Examples

```
dfrechet(seq(1, 5), 0, 1, 1)
qfrechet(pfrechet(seq(1, 5), 0, 1, 1), 0, 1, 1)
rfrechet(5, 0, 1, 1)
```

`frechetdist` *Creates an Object Representing Frechet Distribution*

Description

The function creates an object which represents the Frechet distribution.

Usage

```
frechetdist(loc = 0, scale = 1, shape = 1)
```

Arguments

`loc`	location parameter, default: 0.
`scale`	scale parameter, default: 1.
`shape`	shape parameter, default: 1.

Details

See `Frechet`.

Value

Object of class frechetdist.

See Also

`Frechet`

Examples

```
Fr <- frechetdist(0, 1, 2)
d(Fr, c(2, 3, 4, NA))
r(Fr, 5)
```

gammadist *Creates an Object Representing Gamma Distribution*

Description

The function creates an object which represents the gamma distribution.

Usage

```
gammadist(shape = 2, rate, scale)
```

Arguments

shape	shape parameter, default: 2.
rate	rate parameter, an alternative way to specify the scale.
scale	scale parameter.

Details

See GammaDist.

Value

Object of class gammadist.

See Also

GammaDist

Examples

```
G <- gammadist(shape = 2, scale = 3)
d(G, c(2, 3, 4, NA))
r(G, 5)
```

geomdist *Creates an Object Representing Geometric Distribution*

Description

The function creates an object which represents the geometric distribution.

Usage

```
geomdist(prob = 0.5)
```

Arguments

prob	probability parameter, default: 0.5.

Details

See Geometric.

Value

Object of class geomdist.

See Also

Geometric

Examples

```
G <- geomdist(0.5)
d(G, c(2, 3, 4, NA))
r(G, 5)
```

---

get_opt *Get Parameters*

Description

Function can be used to extract the parameters used in mistr.

Usage

```
get_opt(...)
```

Arguments

...	characteristic strings of desired parameters. Possible values "sub", "add", "tol".

Value

named vector with values.

See Also

set_opt

Examples

```
get_opt("sub", "tol")
```

GNG_fit *Fitting a GPD-Normal-GPD Model*

Description

GNG_fit is used to fit three components composite models with components GPD, normal and GPD.

Usage

```
GNG_fit(data, start = c(break1 = -0.02, break2 = 0.02, mean = 0, sd =
  0.0115, shape1 = 0.15, shape2 = 0.15), break_fix = FALSE,
  midd = mean(data), ...)
```

Arguments

data	vector of values to which the density is optimized.
start	named vector (break1, break2, mean, sd, shape1, shape2) of values that are used to start the optimization, default: c(break1 = -0.02, break2 = 0.02, mean = 0, sd = 0.0115, shape1 = 0.15,shape2 = 0.15).
break_fix	logical, fix the breakpoints at the values from start?, default: FALSE.
midd	split reals into two subintervals, the first breakpoint is then optimized on the left of midd and the second on the right, default: mean(data).
...	further arguments to be passed to the optimizer.

Details

The GNG model is the GPD-Normal-GPD model. This means that a $-X$ transformation of a GPD random variable will be used for the left tail, normal distribution for the center and again GPD for the right tail.

The code uses the maximum likelihood estimation technique to estimate the six parameters from the start vector (break1,break2,mean,sd,shape1,shape2). The other parameters (location and scale parameters of the GPD) are computed in each step such that the function is continuous. Weights are estimated in every step as a proportion of points that correspond to each of the truncated region. If the breakpoints are fixed (i.e. break_fix = TRUE), the weights are computed before the optimization procedure.

Optimization is handled by the mle2 function.

Value

A list of class comp_fit.

See Also

mle2

Examples

```
## Not run:
 GNG_fit(stocks$SAP)

 GNG_fit(stocks$MSFT)

 autoplot(GNG_fit(stocks$ADS))

 GNG_fit(stocks$GSPC, start = c(break1=-0.0075, break2=0.0075, mean=0,
         sd=0.0115, shape1=0.15, shape2=0.15), control = list(maxit = 20000))

 GNG_fit(stocks$DJI, start = c(break1=-0.0055, break2=0.0055, mean=-0.001,
         sd=0.0055,shape1=0.15,shape2=0.15), method = "CG",control = list(maxit = 1000))


## End(Not run)
```

GPD *The Generalized Pareto Distribution*

Description

Density, distribution function, quantile function and random generation for the generalized Pareto distribution with location, scale and shape parameters.

Usage

```
dGPD(x, loc = 0, scale = 1, shape = 0, log = FALSE)

pGPD(q, loc = 0, scale = 1, shape = 0, lower.tail = TRUE,
  log.p = FALSE)

qGPD(p, loc = 0, scale = 1, shape = 0, lower.tail = TRUE,
  log.p = FALSE)

rGPD(n, loc = 0, scale = 1, shape = 0)
```

Arguments

x, q	vector of quantiles.
loc	location parameter.
scale	scale parameter.
shape	shape parameter.
log, log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
p	vector of probabilities.
n	number of observations.

Details

The generalized Pareto distribution function with location parameter μ, scale parameter σ and shape parameter ξ has density given by

$$f(x) = 1/\sigma(1 + \xi z)^{-}(1/\xi + 1)$$

for $x \geq \mu$ and $\xi > 0$, or $\mu - \sigma/\xi \geq x \geq \mu$ and $\xi < 0$, where $z = (x - \mu)/\sigma$. In the case where $\xi = 0$, the density is equal to $f(x) = 1/\sigma e^{-}z$ for $x \geq \mu$. The cumulative distribution function is

$$F(x) = 1 - (1 + \xi z)^{(} - 1/\xi)$$

for $x \geq \mu$ and $\xi > 0$, or $\mu - \sigma/\xi \geq x \geq \mu$ and $\xi < 0$, with z as stated above. If $\xi = 0$ the CDF has form $F(x) = 1 - e^{-}z$.

See https://en.wikipedia.org/wiki/Generalized_Pareto_distribution for more details.

Value

`dGPD` gives the density, `pGPD` gives the distribution function, `qGPD` gives the quantile function, and `rGPD` generates random deviates.

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

See Also

`GPDdist`

Examples

```
dGPD(seq(1, 5), 0, 1, 1)
qGPD(pGPD(seq(1, 5), 0, 1, 1), 0, 1 ,1)
rGPD(5, 0, 1, 1)
```

GPDdist *Creates an Object Representing Generalized Pareto Distribution*

Description

The function creates an object which represents the generalized Pareto distribution.

Usage

```
GPDdist(loc = 0, scale = 1, shape = 0)
```

Arguments

`loc`	location parameter, default: 0.
`scale`	scale parameter, default: 1.
`shape`	shape parameter, default: 0.

Details

See GPD.

Value

Object of class GPDdist.

See Also

GPD

Examples

```
G <- GPDdist(0, 1, 0)
d(G, c(2, 3, 4, NA))
r(G, 5)
```

Gumbel *The Gumbel Distribution*

Description

Density, distribution function, quantile function and random generation for the Gumbel distribution with location and scale parameters.

Usage

```
dgumbel(x, loc, scale, log = FALSE)

pgumbel(q, loc, scale, lower.tail = TRUE, log.p = FALSE)

qgumbel(p, loc, scale, lower.tail = TRUE, log.p = FALSE)

rgumbel(n, loc, scale)
```

Arguments

x, q	vector of quantiles.
loc	location parameter.
scale	scale parameter.
log, log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
p	vector of probabilities.
n	number of observations.

Details

The Gumbel distribution function with location parameter μ and scale parameter β has density given by

$$f(x) = 1/\beta e^{-}(z + e^{-}z)$$

, where $z = (x - \mu)/\beta$. The cumulative distribution function is

$$F(x) = e^{(} - e^{z})$$

with z as stated above.

See https://en.wikipedia.org/wiki/Gumbel_distribution for more details.

Value

`dgumbel` gives the density, `pgumbel` gives the distribution function, `qgumbel` gives the quantile function, and `rgumbel` generates random deviates.

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

See Also

`gumbeldist`

Examples

```
dgumbel(seq(1, 5), 0, 1)
qgumbel(pgumbel(seq(1, 5), 0, 1), 0 ,1)
rgumbel(5, 0, 1)
```

---

gumbeldist *Creates an Object Representing Gumbel Distribution*

Description

The function creates an object which represents the Burr distribution.

Usage

```
gumbeldist(loc = 0, scale = 1)
```

Arguments

`loc`	location parameter, default: 0.
`scale`	scale parameter, default: 1.

Details

See `Gumbel`.

Value

Object of class gumbeldist.

See Also

Gumbel

Examples

```
G <- gumbeldist(1, 2)
d(G, c(2, 3, 4, NA))
r(G, 5)
```

---

hyperdist *Creates an Object Representing Hypergeometric Distribution*

Description

The function creates an object which represents the hypergeometric distribution.

Usage

```
hyperdist(m = 10, n = 10, k = 5)
```

Arguments

m	the number of white balls in the urn, default: 10.
n	the number of black balls in the urn, default: 10.
k	the number of balls drawn from the urn, default: 5.

Details

See Hypergeometric.

Value

Object of class hyperdist.

See Also

Hypergeometric

Examples

```
H <- hyperdist(0.5)
d(H, c(2, 3, 4, NA))
r(H, 5)
```

is.composite *Reports whether O is a Composite Distribution Object*

Description

Reports whether O is a composite distribution object.

Usage

```
is.composite(O)
```

Arguments

O	an object to test.

is.contin *Reports whether O is a Continuous Distribution Object*

Description

Reports whether O is a continuous distribution object.

Usage

```
is.contin(O)
```

Arguments

O	an object to test.

is.discrete *Reports whether O is a Discrete Distribution Object*

Description

Reports whether O is a discrete distribution object.

Usage

```
is.discrete(O)
```

Arguments

O	an object to test.

is.dist *Reports whether O is a Distribution Object*

Description

Reports whether O is a distribution object.

Usage

```
is.dist(O)
```

Arguments

O	an object to test.

is.mixture *Reports whether O is a Mixture Distribution Object*

Description

Reports whether O is a mixture distribution object.

Usage

```
is.mixture(O)
```

Arguments

O	an object to test.

is.standard *Reports whether O is a Standard Distribution Object*

Description

Reports whether O is a standard distribution object.

Usage

```
is.standard(O)
```

Arguments

O	an object to test.

is.transformed *Reports whether O is a Transformed Distribution Object*

Description

Reports whether O is a transformed distribution object.

Usage

```
is.transformed(O)
```

Arguments

O	an object to test.

jumps *Probability mass points*

Description

Function returns a vector of points where a mass of probability is present. These points are then used in plot and plotgg calls.

Usage

```
jumps(O, interval)

## S3 method for class 'discrdist'
jumps(O, interval)

## S3 method for class 'trans_discrdist'
jumps(O, interval)

## S3 method for class 'contdist'
jumps(O, interval)

## S3 method for class 'trans_contdist'
jumps(O, interval)

## S3 method for class 'mixdist'
jumps(O, interval)

## S3 method for class 'trans_mixdist'
jumps(O, interval)
```

```
## S3 method for class 'compdist'
jumps(0, interval)

## S3 method for class 'trans_compdist'
jumps(0, interval)
```

Arguments

0	distribution object.
interval	interval in which the support of discrete elements should be found.

Value

Vector of values.

Note

The function is designed in a way that it rather returns more than less. Thus it might return a value that is close to the interval but not in. This is for use of the package not a problem as jumps is internally used only in plots and quantile function of a mixture distribution where an additional value can not influence the output.

Examples

```
B <- binomdist(12, 0.4)
P <- poisdist(2)

I <- c(-7, 16.8)
jumps(B, I)
jumps(P, I)
```

---

last_history *Returns the Last Element from History List*

Description

Function returns the last element from history list.

Usage

```
last_history(0, t)
```

Arguments

0	transformed distribution object.
t	which characterization should be extracted.

Value

Expression if t is set to "expre", "iexpre", "oldprint" and "oldderiv". Numeric and string if t is equal to "value" and "operation", respectively.

Examples

```
B  <- binomdist(10, 0.3)
B2 <- -3*log(B)
last_history(B2, "value")
last_history(B2, "operation")
```

`lnormdist` *Creates an Object Representing Log Normal Distribution.*

Description

The function creates an object which represents the log normal distribution.

Usage

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

Arguments

`meanlog`	mean parameter, default: 0.
`sdlog`	standard deviation parameter, default: 1.

Details

See `Lognormal`.

Value

Object of class lnormdist.

See Also

`Lognormal`

Examples

```
L <- lnormdist(0, 1)
d(L, c(2, 3, 4, NA))
r(L, 5)
```

`mistr_d_p_q_r` *Mistr d/p/q/r Wrappers*

Description

The functions `mistr_d`, `mistr_p`, `mistr_q`, `mistr_r` are wrappers for `d`, `p`, `q` and `r`, respectively.

Usage

```
mistr_d(O, x, log = FALSE)

mistr_p(O, q, lower.tail = TRUE, log.p = FALSE)

mistr_q(O, p, lower.tail = TRUE, log.p = FALSE, ...)

mistr_r(O, n)
```

Arguments

`O`	distribution object.
`x, q`	vector of quantiles.
`log, log.p`	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
`lower.tail`	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
`p`	vector of probabilities.
`...`	further arguments to be passed.
`n`	number of observations.

Details

Wrappers are offered as a consequence of R-Studio in Windows OS where the `q()` calls in the console are caught and terminate the `R` session.

Value

Vector of computed results.

mistr_theme *Mistr Theme for Ggplot*

Description

Theme for plots that use ggplot2.

Usage

```
mistr_theme(grey = FALSE, blue = FALSE, legend.position = "right",
  ...)
```

Arguments

grey	logical, if TRUE grey palette is used, default: FALSE.
blue	logical, if TRUE blue palette is used, default: FALSE.
legend.position	position of legend, default: "right".
...	further arguments to be passed.

Value

ggplot theme.

See Also

theme

mixdist *Creates an Object Representing Mixture Distribution*

Description

mixdist creates an object which represents the mixture distribution.

Usage

```
mixdist(..., weights)

## S3 method for class 'dist'
mixdist(..., weights)

## Default S3 method:
mixdist(dist, params, weights, ...)
```

Arguments

...	distribution objects.
weights	vector of weights for the components.
dist	vector of distribution names.
params	list of parameters for each component.

Details

A CDF of a mixture distribution function is

$$F(A) = \sum w_i F_i(A)$$

, where w_i is the weight of the i-th component and $F_i()$ is the CDF of the i-th component.

The objects can be specified in two ways, either the user may enter distribution objects or a vector of names and list of parameters. See the examples below.

Value

Object of class mixdist.

See Also

compdist

Examples

```
# using the objects
M <- mixdist(normdist(1, 3), expdist(4), weights = c(0.7, 0.3))
M

# using the names and parameters
M2 <- mixdist(c("norm", "exp"), list(c(mean = 1, sd = 3), c(rate = 4)),
              weights = c(0.7, 0.3))
M2
```

---

monot *Monotonicity of Transformation*

Description

Function checks whether the transformation is increasing or decreasing.

Usage

```
monot(0)

## S3 method for class 'trans_univdist'
monot(0)
```

Arguments

`O`	transforms distribution object.

Value

1 for increasing and -1 for decreasing.

`multinomdist` *Creates an Object Representing Multinomial Distribution*

Description

The function creates an object which represents the multinomial distribution.

Usage

```
multinomdist(size = 10, prob = c(0.5, 0.5))
```

Arguments

`size`	size parameter, default: 10.
`prob`	probability parameter vector, default: c(0.5, 0.5).

Details

See `Multinomial`.

Value

Object of class multinomdist.

See Also

`Multinomial`

Examples

```
M <- multinomdist(10, c(0.5, 0.5))
d(M, c(7, 3))
r(M, 5)
```

nbinomdist *Creates an Object Representing Negative Binomial Distribution*

Description

The function creates an object which represents the negative binomial distribution.

Usage

```
nbinomdist(size = 10, prob, mu)
```

Arguments

size	size parameter, default: 10.
prob	probability parameter.
mu	alternative parametrization via mean, see NegBinomial.

Details

See NegBinomial.

Value

Object of class nbinomdist.

See Also

NegBinomial

Examples

```
N <- nbinomdist(10, 0.5)
d(N, c(2, 3, 4, NA))
r(N, 5)
```

new_dist *Creates New Distribution Object*

Description

The function creates distribution objects that satisfy the naming convention used in package mistr.

Usage

```
new_dist(name, from, to, by = NULL,
  parameters = mget(names(eval(quote(match.call()), parent)[-1]),
  parent), class = deparse(sys.calls()[[sys.nframe() - 1]][[1]]),
  parent = parent.frame())
```

Arguments

name	string containing the name of the distribution.
from	numeric representing where the support of distribution starts.
to	numeric representing where the support of distribution ends.
by	numeric representing the deterministic step between support values. If NULL: continuous distribution is assumed. If the value is specified: discrete distribution with specified step is assumed, default: NULL.
parameters	named list of parameters of the distribution, default: mget(names(eval(quote(match.call()),parent)[-1]),parent).
class	class of the distribution, this should be set in [name]dist convention (e.g. normdist, tdist), default: deparse(sys.calls()[[sys.nframe() - 1]][[1]]).
parent	parent environment, default: parent.frame().

Details

The function can be used in two ways. Either it can be called from the creator functions as for example normdist or unifdist, or directly from any function or enviroment. In the former, only arguments "name", "from" and "to" must be set. Other arguments will be filled according to the parent calls. If this function is called directly, the arguments "parameters" and "class" have to be specified also.

Value

distribution object.

Examples

```
## Not run:
# using creator function
unifdist <- function(min = 0, max = 1) {
   if (!is.numeric(min) || !is.numeric(max))   stop("Parameters must be a numeric")
   if (min >= max)   stop("min must be smaller than max.")
   new_dist(name = "Uniform", from = min, to = max)
}

#directly
U <- new_dist(name = "Uniform", from = 1, to = 6,
              parameters =  list(min = 1, max = 6), class = "unifdist")

## End(Not run)
```

normdist *Creates an Object Representing Normal Distribution*

Description

The function creates an object which represents the normal distribution.

Usage

```
normdist(mean = 0, sd = 1)
```

Arguments

mean	mean parameter, default: 0.
sd	standard deviation parameter, default: 1.

Details

See Normal.

Value

Object of class normdist.

See Also

Normal

Examples

```
N <- normdist(1, 5)
d(N, c(2, 3, 4, NA))
r(N, 5)
```

p.compdist *Distribution Function*

Description

p is a generic function that evaluates the distribution function of a distribution object at given values.

Usage

```
## S3 method for class 'compdist'
p(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'mixdist'
p(O, q, lower.tail = TRUE, log.p = FALSE)

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

## S3 method for class 'standist'
p(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'trans_univdist'
p(O, q, lower.tail = TRUE, log.p = FALSE)
```

Arguments

O	distribution object.
q	vector of quantiles.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.

Details

Methods of p function evaluates any offered distribution from the package mistr. The function makes use of the p[sufix] functions as pnorm or pbeta and thus, if a new distribution is added, these functions must be reachable through the search path.

Value

Vector of computed results.

Examples

```
N <- normdist(1,3)
p(N, c(NA,1,3,5))

C <- cauchydist()
M <- mixdist(N, C, weights = c(0.5, 0.5))
p(M, c(NA,1,3,5))

CC <- compdist(N, C, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*C+5
p(CCC, c(NA,1,3,5))
```

parameters *Extract Model Parameters*

Description

parameters is a generic function which extracts parameters from mistr distribution objects.

Usage

```
parameters(0)

## S3 method for class 'standist'
parameters(0)

## S3 method for class 'trans_standist'
parameters(0)

## S3 method for class 'mixdist'
parameters(0)

## S3 method for class 'trans_mixdist'
parameters(0)

## S3 method for class 'compdist'
parameters(0)

## S3 method for class 'trans_compdist'
parameters(0)

## S3 method for class 'comp_fit'
parameters(0)
```

Arguments

0	an object for which the extraction of model parameters is meaningful.

Value

Vector (for standard distributions) or list (in the case of mixture/composite distribution) of parameters extracted from the object.

For a fitted object of class comp_fit returns vector of fitted parameters.

See Also

weights, breakpoints

Examples

```
N <- normdist(1, 3)
parameters(N)

C <- cauchydist()
M <- mixdist(N, C, weights = c(0.5, 0.5))
parameters(M)
```

Pareto *The Pareto Distribution*

Description

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

Usage

```
dpareto(x, scale = 1, shape = 1, log = FALSE)

ppareto(q, scale = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)

qpareto(p, scale = 1, shape = 1, lower.tail = TRUE, log.p = FALSE)

rpareto(n, scale = 1, shape = 1)
```

Arguments

x, q	vector of quantiles.
scale	scale parameter.
shape	shape parameter.
log, log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
p	vector of probabilities.
n	number of observations.

Details

The Pareto distribution function with scale parameter s and shape parameter α has density given by

$$f(x) = \alpha s^{\alpha}/x^(\alpha + 1)$$

for $x \geq s$. The cumulative distribution function is

$$F(x) = 1 - (s/x)^{\alpha}$$

for $x \geq s$. See https://en.wikipedia.org/wiki/Pareto_distribution for more details.

Value

dpareto gives the density, ppareto gives the distribution function, qpareto gives the quantile function, and rpareto generates random deviates.

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

See Also

paretodist

Examples

```
dpareto(seq(1, 5), 1, 1)
qpareto(ppareto(seq(1, 5), 1, 1), 1 ,1)
rpareto(5, 1, 1)
```

---

paretodist *Creates an Object Representing Pareto Distribution*

Description

The function creates an object which represents the Pareto distribution.

Usage

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

Arguments

scale	scale parameter, default: 1.
shape	shape parameter, default: 1.

Details

See Pareto.

Value

Object of class paretodist.

See Also

Pareto

Examples

```
P <- paretodist(1, 1)
d(P, c(2, 3, 4, NA))
r(P, 5)
```

plim.compdist *Left-Hand Limit of Distribution Function*

Description

plim is a generic function that evaluates the left-hand limit of distribution function for a distribution object at given values.

Usage

```
## S3 method for class 'compdist'
plim(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'mixdist'
plim(O, q, lower.tail = TRUE, log.p = FALSE)

plim(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'discrdist'
plim(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'contdist'
plim(O, q, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'trans_univdist'
plim(O, q, lower.tail = TRUE, log.p = FALSE)
```

Arguments

O	distribution object.
q	vector of quantiles.
lower.tail	logical; if TRUE, probabilities are $P[X < x]$ otherwise, $P[X \geq x]$, default: TRUE.
log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.

Details

Methods of plim function evaluates the left-hand limit of any offered distribution from the package mistr. The left-hand limit is defined as $F(x-) = P(X < x)$. The function makes use of the p[sufix] and q[sufix] functions as pnorm or qbeta and thus, if a new distribution is added, these functions must be reachable through the search path.

Value

Vector of computed results.

Examples

```
B <- binomdist(10, 0.3)
plim(B, c(NA, 1, 3, 5))

P <- poisdist()
M <- mixdist(B, P, weights = c(0.5, 0.5))
plim(M, c(NA, 1, 3, 5))

CC <- compdist(B, P, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*CC+5
plim(CCC, c(NA, 1, 3, 5))
```

`plot.comp_fit` *Autoplot of Fitted Distributions*

Description

The function plots the CDF, PDF and QQ-plot of a fitted distribution object together with the empirical values.

Usage

```
## S3 method for class 'comp_fit'
plot(x, which = "all", layout = matrix(c(1, 2, 1,
  3), nrow = 2), empir_color = "#122e94", mtext_cex = sett, ...)
```

Arguments

`x`	distribution object.
`which`	whether to plot only CDF, PDF, qq or all three, default: 'all'.
`layout`	layout of plots, default: matrix(c(1, 2, 1, 3), nrow = 2).
`empir_color`	color of empirical data, default: '#122e94'.
`mtext_cex`	cex parameter for mtexts used in the plots.
`...`	further arguments to be passed.

See Also

`Distribution_autoplot`

plotgg *Autoplot of Distributions Using ggplot2*

Description

The function plotgg plots the CDF and PDF of a given distribution object.

Usage

```
plotgg(x, which = "all", ...)

## S3 method for class 'contdist'
plotgg(x, which = "all", pp1 = 1000, pp2 = 1000,
  col = "#F9D607", xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL,
  xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL, ylab1 = NULL,
  xlab2 = NULL, ylab2 = NULL, main1 = "CDF", main2 = "PDF",
  size1 = 1, size2 = 1, alpha1 = 0.7, alpha2 = 0.7, ...)

## S3 method for class 'trans_contdist'
plotgg(x, which = "all", pp1 = 1000,
  pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01, 0.99)),
  ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PDF", size1 = 1, size2 = 1, alpha1 = 0.7,
  alpha2 = 0.7, ...)

## S3 method for class 'discrdist'
plotgg(x, which = "all", col = "#F9D607",
  xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1,
  ylim2 = NULL, xlab1 = NULL, ylab1 = NULL, xlab2 = NULL,
  ylab2 = NULL, main1 = "CDF", main2 = "PMF", size1 = 3.3,
  size2 = 3.3, alpha1 = 0.9, alpha2 = 0.9, col_segment = "#b05e0b",
  ...)

## S3 method for class 'trans_discrdist'
plotgg(x, which = "all", col = "#F9D607",
  xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1,
  ylim2 = NULL, xlab1 = NULL, ylab1 = NULL, xlab2 = NULL,
  ylab2 = NULL, main1 = "CDF", main2 = "PMF", size1 = 3.3,
  size2 = 3.3, alpha1 = 0.9, alpha2 = 0.9, col_segment = "#b05e0b",
  ...)

## S3 method for class 'contmixdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01,
  0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
```

```
  main2 = "PDF", size1 = 1, size2 = 1, alpha1 = 0.4,
  alpha2 = 0.4, legend.position1 = "none", legend.position2 = "none",
  ...)

## S3 method for class 'trans_contmixdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01,
  0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PDF", size1 = 1, size2 = 1, alpha1 = 0.4,
  alpha2 = 0.4, legend.position1 = "none", legend.position2 = "none",
  ...)

## S3 method for class 'discrmixdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01, 0.99)),
  ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PMF", size1 = 1.6, size2 = 1.6, alpha1 = 0.4,
  alpha2 = 0.9, legend.position1 = "none", legend.position2 = "none",
  width = 0.25, ...)

## S3 method for class 'trans_discrmixdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01, 0.99)),
  ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PMF", size1 = 1.6, size2 = 1.6, alpha1 = 0.4,
  alpha2 = 0.9, legend.position1 = "none", legend.position2 = "none",
  width = 0.25, ...)

## S3 method for class 'contdiscrmixdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01,
  0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PDF", size1 = 1.6, size2 = 1.6, alpha1 = 0.4,
  alpha2 = 0.4, legend.position1 = "none", legend.position2 = "none",
  ...)

## S3 method for class 'trans_contdiscrmixdist'
plotgg(x, which = "all",
  only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "#F9D607",
  xlim1 = q(x, c(0.01, 0.99)), ylim1 = NULL, xlim2 = xlim1,
  ylim2 = NULL, xlab1 = NULL, ylab1 = NULL, xlab2 = NULL,
  ylab2 = NULL, main1 = "CDF", main2 = "PDF", size1 = 1.6,
  size2 = 1.6, alpha1 = 0.4, alpha2 = 0.4,
  legend.position1 = "none", legend.position2 = "none", ...)
```

```
## S3 method for class 'compdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01,
  0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PDF", size1 = 1.6, size2 = 1.6, alpha1 = 0.4,
  alpha2 = 0.4, legend.position1 = "none", legend.position2 = "none",
  text_ylim = -0.01, col_segment = "white", lty_segment = 3,
  lwd_segment = 1.8, ...)

## S3 method for class 'trans_compdist'
plotgg(x, which = "all", only_mix = FALSE,
  pp1 = 1000, pp2 = 1000, col = "#F9D607", xlim1 = q(x, c(0.01,
  0.99)), ylim1 = NULL, xlim2 = xlim1, ylim2 = NULL, xlab1 = NULL,
  ylab1 = NULL, xlab2 = NULL, ylab2 = NULL, main1 = "CDF",
  main2 = "PDF", size1 = 1.6, size2 = 1.6, alpha1 = 0.4,
  alpha2 = 0.4, legend.position1 = "none", legend.position2 = "none",
  text_ylim = -0.01, col_segment = "white", lty_segment = 3,
  lwd_segment = 1.8, ...)
```

Arguments

x	distribution object.
which	whether to plot only CDF, PDF or both, default: 'all'.
...	further arguments to be passed.
pp1	number of points at which CDF is evaluated, default: 1000.
pp2	number of points at which PDF is evaluated, default: 1000.
col	color used in plot, default: '#122e94'.
xlim1	xlim of CDF plot, default: q(x, c(0.01, 0.99)).
ylim1	ylim of CDF plot, default: NULL.
xlim2	xlim of PDF plot, default: xlim1.
ylim2	ylim of PDF plot, default: NULL.
xlab1	xlab of CDF plot, default: NULL.
ylab1	ylab of CDF plot, default: NULL.
xlab2	xlab of PDF plot, default: NULL.
ylab2	ylab of PDF plot, default: NULL.
main1	title of CDF plot, default: 'CDF'.
main2	title of PDF plot, default: 'PDF'/'PMF'.
size1	size used in CDF plot.
size2	size used in PDF plot.
alpha1	alpha used in CDF plot.
alpha2	alpha used in PDF plot.

col_segment — col of additional segment if contained in the plot (composite and discrete distributions).

only_mix — whether to plot only mixture/composite model and not also the components, default: FALSE.

legend.position1 — legend.position used in CDF plot.

legend.position2 — legend.position used in PDF plot.

width — width of the bars that are used to plot discrete mixtures, default: 0.25.

text_ylim — y coordinate for text annotation, default: -0.01.

lty_segment — lty of additional segment if contained in the plot (composite and discrete distributions).

lwd_segment — lwd of additional segment if contained in the plot (composite and discrete distributions).

Value

ggplot object if which = "cdf" or which = "pdf". If both are plotted, the plots are merged using `multiplot()` function and a list with both plots is invisibly returned.

Examples

```
## Not run:
N <- normdist()
autoplot(N)

# manipulating cdf plot
B <- binomdist(12, 0.5)
autoplot(-3*B, which = "cdf", xlim1 = c(-30, -10))
# manipulating pdf plot
autoplot(-3*B, which = "pdf", xlim2 = c(-30, -10))

## End(Not run)
```

PNP_fit *Fitting a Pareto-Normal-Pareto Model*

Description

`GNG_fit` is used to fit three components composite models with components Pareto, normal and Pareto.

Usage

```
PNP_fit(data, start = c(break1 = -0.02, break2 = 0.02, mean = 0, sd =
  0.012), ...)
```

Arguments

data	vector of values to which the density is optimized.
start	named vector (break1, break2, mean, sd) of values that are used to start the optimization, default: c(break1 = -0.02, break2 = 0.02, mean = 0, sd = 0.012).
...	further arguments to be passed to optimizer.

Details

The PNP model is the Pareto-Normal-Pareto model. This means that a $-X$ transformation of a Pareto random variable will be used for the left tail, normal distribution for the center and again Pareto for the right tail.

The code uses the maximum likelihood estimation technique to estimate the four parameters from the start vector (`break1,break2,mean,sd`). The other parameters (shape parameters of Pareto distribution) are computed in each step such that the function is continuous. Weights are estimated in every step as a proportion of points that correspond to each of the truncated region.

Optimization is handled by the `mle2` function.

Value

A list of class comp_fit.

See Also

`mle2`

Examples

```
## Not run:
 PNP_fit(stocks$SAP)

 PNP_fit(stocks$MSFT)

 autoplot(PNP_fit(stocks$ADS))

 PNP_fit(stocks$GSPC, method = "BFGS")

 PNP_fit(stocks$DJI, start = c(-0.01,0.01,0,0.008))


## End(Not run)
```

poisdist *Creates an Object Representing Poisson Distribution*

Description

The function creates an object which represents the Poisson distribution.

Usage

```
poisdist(lambda = 1)
```

Arguments

lambda	mean parameter, default: 1.

Details

See Poisson.

Value

Object of class poisdist.

See Also

Poisson

Examples

```
P <- poisdist(1)
d(P, c(2, 3, 4, NA))
r(P, 5)
```

q.compdist *Quantile Function*

Description

q is a generic function that evaluates the quantile function of a distribution object at given values.

Usage

```
## S3 method for class 'compdist'
q(O, p, lower.tail = TRUE, log.p = FALSE, ...)

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

## S3 method for class 'standist'
q(O, p, lower.tail = TRUE, log.p = FALSE, ...)

## S3 method for class 'trans_univdist'
q(O, p, lower.tail = TRUE, log.p = FALSE, ...)
```

Arguments

O	distribution object.
p	vector of probabilities.
lower.tail	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
log.p	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
...	further arguments to be passed.

Details

Methods of q function evaluates any offered distribution from package mistr. The function makes use of the q[sufix] functions as qnorm or qbeta and thus, if a new distribution is added, these functions must be reachable through the search path.

The mixture method q.mixdist and the default method q.default have its own help page.

Value

Vector of computed results.

Examples

```
N <- normdist(1, 3)
q(N, c(NA, 1, 3, 5))

C <- cauchydist()
CC <- compdist(N, C, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*C+5
q(CCC, c(NA, 1, 3, 5))
```

`q.default` *Terminate an R Session*

Description

The default method `q.default` terminates the current R session.

Usage

```
## Default S3 method:
q(O = save, p = status, lower.tail = runLast,
  log.p = FALSE, save = "default", status = 0, runLast = TRUE, ...)
```

Arguments

`O`	place holder for generic, by default set to save, default: save.
`p`	place holder for generic, by default set to status, default: status.
`lower.tail`	place holder for generic, by default set to runLast, default: runLast.
`log.p`	place holder for generic, default: FALSE.
`save`	a character string indicating whether the environment (workspace) should be saved, one of "no", "yes", "ask" or "default", default: 'default'.
`status`	the (numerical) error status to be returned to the operating system, where relevant. Conventionally 0 indicates successful completion, default: 0.
`runLast`	should .Last() be executed?, default: TRUE.
`...`	further arguments to be passed.

Details

This method is designed to quit R if the `q()` without a distribution is called. The reason for such an implementation is R-Studio in Linux and Mac systems, where the software calls `q()` (rather than `base::q()`) once the R-Studio window is closed. Such implementation solves the issued with the overwriting of `q()`.

See Also

`q`

`q.mixdist` *Quantile Function of a Mixture Model*

Description

`q.mixdist` is a method that evaluates the quantile function of a mixture distribution object at given values.

Usage

```
## S3 method for class 'mixdist'
q(O, p, lower.tail = TRUE, log.p = FALSE, ...)
```

Arguments

`O`	mixture distribution object.
`p`	vector of probabilities.
`lower.tail`	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
`log.p`	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.
`...`	further arguments to be passed.

Details

Methods of `q` function evaluates any offered distribution from the package `mistr`. The function makes use of the p[sufix] and q[sufix] functions as `pnorm` or `qbeta` and thus, if a new distribution is added, these functions must be reachable through the search path.

The values are numerically found using the `uniroot` function, while the starting intervals are found automatically. The option parameter `tol` specifies the tolerance for the `uniroot`. Options parameter `sub` is used to test whether the CDF at computed values minus `sub` is not the same and thus the given value is not an infimum. In such case, the root is found one more time for the value `p -sub`.

Other methods `q` and the default method `q.default` have its own help page.

Value

Vector of computed results.

See Also

`set_opt`

Examples

```
DM <- mixdist(3*binomdist(12, 0.4), -2*poisdist(2)+12, weights=c(0.5, 0.5))
y <- c(0.4, p(DM, c(5, 10, 15, 18)), 0.95)
x <- q(DM, y)
plot(DM, which = "cdf", only_mix=TRUE, xlim1 = c(0, 37))
points(x, y)
```

---

qlim.compdist *Right-Hand Limit of Quantile Function*

Description

`qlim` is a generic function that evaluates the right-hand limit of quantile function for a distribution object at given values.

Usage

```
## S3 method for class 'compdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)

qlim(O, p, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'discrdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'contdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'trans_univdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`O`	distribution object.
`p`	vector of probabilities.
`lower.tail`	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
`log.p`	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.

Details

Methods of `qlim` function evaluates the right-hand limit of any offered distribution object from the package `mistr`. The right-hand limit of a quantile function is defined as

$$Q(x+) = inf x : p < P(X \leq x).$$

The function makes use of the p[sufix] and q[sufix] functions as `pnorm`, `pbeta`, `qnorm`, `qbeta`, and thus, if a new distribution is added, these functions must be reachable through the search path.

Methods for `mixtures` have its own help page.

Value

Vector of computed results.

Examples

```
B <- binomdist(10, 0.3)
qlim(B, plim(B, c(NA, 1, 3, 5)))

P <- poisdist()
M <- mixdist(B, P, weights = c(0.5, 0.5))
qlim(M, plim(M, c(NA, 1, 3, 5)))

CC <- compdist(B, P, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*CC+5
qlim(CCC, plim(CCC, c(NA, 1, 3, 5)))
```

---

qlim.discrmixdist *Right-Hand Limit of Mixture Quantile Function*

Description

`qlim.mixdist` is a method that evaluates the right-hand limit of quantile function for a mixture distribution object at given values.

Usage

```
## S3 method for class 'discrmixdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'contdiscrmixdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)

## S3 method for class 'contmixdist'
qlim(O, p, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`O`	mixture distribution object.
`p`	vector of probabilities.
`lower.tail`	logical; if TRUE, probabilities are $P[X \leq x]$ otherwise, $P[X > x]$, default: TRUE.
`log.p`	logical; if TRUE, probabilities p are given as $log(p)$, default: FALSE.

Details

Methods of `qlim` function evaluates the right-hand limit of a quantile function for any offered distribution object from the package `mistr`. The right-hand limit of a quantile function is defined as

$$Q(x+) = infx : p < P(X \leq x).$$

The function makes use of the p[sufix] and q[sufix] functions as `pnorm`, `pbeta`, `qnorm`, `qbeta`, and thus, if a new distribution will be added, these functions must be reachable through the search path.

The values are numerically found using the `uniroot` function, while the starting intervals are found automatically. The option parameter `tol` specifies the tolerance for the `uniroot`. Options parameter `sub` is used to test whether the CDF at computed value minus `sub` is not the same and thus the given value is not an infimum. In such case, the root is found one more time for the value `p -sub`.

Other methods `qlim` have its own help page.

Value

Vector of computed results.

See Also

`set_opt`

Examples

```
# q() of a negative transformed random variable uses qlim()
DM <- mixdist(3*binomdist(12,0.4), -2*poisdist(2)+12, weights=c(0.5, 0.5))
y <- c(0.05, 0.4, p(-DM, c(-5, -10, -15, -18)), 0.95)
x <- q(-DM, y)
plot(-DM, which = "cdf", only_mix=TRUE, xlim1 = c(-37, 0))
points(x, y)
```

QQplot *Quantile-Quantile Plot*

Description

QQplot is a generic function that produces QQ plot of two datasets, distribution and dataset or two distributions.

Usage

```
QQplot(d1, d2, line = TRUE, col = "#122e94", line_col = "#f28df9",
  xlab = deparse(substitute(d1)), ylab, main = "Q-Q plot", lwd = 2,
  ...)

## Default S3 method:
QQplot(d1, d2, line = TRUE, col = "#122e94",
```

```
  line_col = "#f28df9", xlab = deparse(substitute(d1)),
  ylab = deparse(substitute(d2)), main = "Q-Q plot", lwd = 2, ...)

## S3 method for class 'dist'
QQplot(d1, d2, line = TRUE, col = "#122e94",
  line_col = "#f28df9", xlab = deparse(substitute(d1)), ylab = ylabe,
  main = "Q-Q plot", lwd = 2, CI = re, conf = 0.95, n = 100,
  CI_col = "grey80", ...)

QQnorm(d2, xlab = "Standard Normal", ylab = deparse(substitute(d2)),
  ...)
```

Arguments

d1	distribution object or dataset.
d2	distribution object or dataset.
line	if qqline should be included, default: TRUE.
col	color of points, default: '#122e94'.
line_col	color of qqline, default: '#f28df9'.
xlab	xlab, default: deparse(substitute(d1)).
ylab	ylab, default: deparse(substitute(d2)).
main	title, default: 'Q-Q plot'.
lwd	lwd of qqline, default: 2.
...	further arguments to be passed.
CI	if confidence bound should be included.
conf	confidence level for confidence bound, default: 0.95.
n	number of points at which quantile functions are evaluated if two distributions are compared, default: 100.
CI_col	color of confidence bound, default: 'grey80'.

Details

`QQplot` is able to compare any combination of dataset and distributions.

`QQnorm` is a wrapper around `QQplot`, where d1 is set to `normdist()`.

If quantiles of a continuous distribution are compared with a sample, a confidence bound for data is offered. This confidence "envelope" is based on the asymptotic results of the order statistics. For more details see https://en.wikipedia.org/wiki/Order_statistic.

Examples

```
# sample vs sample
QQplot(r(normdist(), 10000), r(tdist(df = 4), 10000))

# distribution vs sample
QQplot(normdist(), r(tdist(df = 4), 10000))
```

```
# distribution vs distribution
QQplot(normdist(), tdist(df = 4))
```

QQplotgg *Implementation of Quantile-Quantile Plot with ggplot2*

Description

QQplotgg is a generic function that produces QQ plot of two datasets, distribution and dataset or two distributions.

Usage

```
QQplotgg(d1, d2, line = TRUE, col = "#F9D607", line_col = "#f28df9",
  xlab = deparse(substitute(d1)), ylab, main = "Q-Q plot", alpha,
  lwd = 1, ...)

## Default S3 method:
QQplotgg(d1, d2, line = TRUE, col = "#F9D607",
  line_col = "#f28df9", xlab = deparse(substitute(d1)),
  ylab = deparse(substitute(d2)), main = "Q-Q plot", alpha = 0.5,
  lwd = 1, ...)

## S3 method for class 'dist'
QQplotgg(d1, d2, line = TRUE, col = "#F9D607",
  line_col = "#f28df9", xlab = deparse(substitute(d1)), ylab = ylabe,
  main = "Q-Q plot", alpha = 0.7, lwd = 1, CI = re,
  CI_alpha = 0.4, CI_col = line_col, conf = 0.95, n = 100, ...)

QQnormgg(d2, xlab = "Standard Normal", ylab = deparse(substitute(d2)),
  ...)
```

Arguments

d1	distribution object or dataset.
d2	distribution object or dataset.
line	if qqline should be included, default: TRUE.
col	color of points, default: '#F9D607'.
line_col	color of qqline, default: '#f28df9'.
xlab	xlab, default: deparse(substitute(d1)).
ylab	ylab. default: deparse(substitute(d2)).
main	title, default: 'Q-Q plot'.
alpha	alpha of points, default: 0.7.
lwd	lwd of qqline, default: 1.

...	further arguments to be passed.
CI	if confidence bound should be included.
CI_alpha	alpha of confidence bound, default: 0.4.
CI_col	color of confidence bound , default: line_col.
conf	confidence level for confidence bound, default: 0.95.
n	number of points at which quantile functions are evaluated if two distributions are compared, default: 100.

Details

QQplotgg is able to compare any combination of dataset and distributions.

QQnormgg is a wrapper around QQplotgg, where d1 is set to normdist().

If quantiles of a continuous distribution are compared with a sample, a confidence bound for data is offered. This confidence "envelope" is based on the asymptotic results of the order statistics. For more details see https://en.wikipedia.org/wiki/Order_statistic.

Value

ggplot object.

Examples

```
# sample vs sample
QQplotgg(r(normdist(), 10000), r(tdist(df = 4), 10000))

# distribution vs sample
QQplotgg(normdist(), r(tdist(df = 4), 10000))

# distribution vs distribution
QQplotgg(normdist(), tdist(df = 4))
```

---

q_approxfun *Quantile Function Approximation*

Description

q_approxfun is a generic function that for a given object generates function to approximate the quantile function.

Usage

```
q_approxfun(O, range = q(O, c(0.005, 0.995)), n = 1000)

## S3 method for class 'dist'
q_approxfun(O, range = q(O, c(0.005, 0.995)), n = 1000)
```

Arguments

O	distribution object.
range	interval on which the grid is defined, q(O, c(0.005, 0.995)).
n	number of points within the grid, default: 1000.

Details

Function q_approxfun generates a grid of values on which the CDF of the object is evaluated. The function returns a quantile function that uses approx and the values of the grid to approximate the quantiles. This function is designed mostly for the mixture distributions where the standard q method may be slow and thus allows to trade the accuracy for the speed.

The returned function takes the arguments p, lower.tail and log.p, see q.

Value

Function.

Examples

```
N <- normdist(1, 3)
N2 <- normdist(8, 3)

M <- mixdist(N, N2, weights = c(0.5, 0.5))
q_app <- q_approxfun(M)

q_app(c(.2, .5, .7))
q_app(c(.2, .5, .7), lower.tail = FALSE)
```

---

r.compdist *Random Generation*

Description

r is a generic function that generates random deviates of a distribution object.

Usage

```
## S3 method for class 'compdist'
r(O, n)

## S3 method for class 'mixdist'
r(O, n)

r(O, n)

## S3 method for class 'standist'
r(O, n)
```

```
## S3 method for class 'hyperdist'
r(O, n)

## S3 method for class 'wilcoxdist'
r(O, n)

## S3 method for class 'trans_univdist'
r(O, n)
```

Arguments

O	distribution object.
n	number of observations.

Details

Methods of `r` function generates random deviates of offered distribution from the package `mistr`. The function makes use of the r[sufix] functions as `rnorm` or `rbeta` and thus, if a new distribution is added, these functions must be reachable through the search path.

For more complicated composite distributions, where one of the components is a mixture distribution, the function performs a rejection sampling of mixture random numbers to improve the speed.

Value

Vector of computed results.

Examples

```
N <- normdist(1, 3)
r(N, 5)

C <- cauchydist()
M <- mixdist(N, C, weights = c(0.5, 0.5))
r(M, 5)

CC <- compdist(N, C, weights = c(0.5, 0.5), breakpoints = 1)
CCC <- 2*C+5
r(CCC, 5)
```

---

risk *Risk Measures of Fitted Objects*

Description

`risk` computes the VaR, ES and expectiles at a given level for fitted distribution.

Usage

```
risk(model, alpha, expectile = TRUE, plot = FALSE, ggplot = FALSE,
  text_ylim = -0.15, size = 1)

## S3 method for class 'PNP'
risk(model, alpha = 0.05, expectile = TRUE,
  plot = FALSE, ggplot = FALSE, text_ylim = -0.15, size = 1)

## S3 method for class 'GNG'
risk(model, alpha = 0.05, expectile = TRUE,
  plot = FALSE, ggplot = FALSE, text_ylim = -0.15, size = 1)
```

Arguments

model	output object of GNG_fit() or PNP_fit().
alpha	levels of risk measures.
expectile	logical, if also expectiles should be computed, default: TRUE.
plot	plot the results?, default: FALSE.
ggplot	plot the results with ggplot2?, default: FALSE.
text_ylim	y coordinate for annotation in ggplot2, default: -0.15.
size	size of the text indicating the risk measures in the plot, default: 1.

Details

VaR are computed using the q() call of the fitted distribution.

ES is computed directly (i.e. the integrals are precomputed, not numerically) as an integral of the quantile function.

Expectiles can be obtained as a unit-root solution of the identity between quantiles and expectiles. These are equivalent for corresponding τ and α if

$$\tau = (\alpha q(\alpha) - G(\alpha))/(\mu - 2G(\alpha) - (1-2\alpha)q(\alpha))$$

where μ is mean, $q()$ is the quantile function and $G(\alpha) = \int_{-\infty}^{q(\alpha)} ydF(y)$.

Value

List of class risk_measures.

Examples

```
## Not run:
 GNG <- GNG_fit(stocks$SAP)
 PNP <- PNP_fit(stocks$MSFT)

 risk(PNP, alpha = c(0.01,0.05,0.08,0.1))
 risk(GNG, alpha = c(0.01,0.05,0.08,0.1), plot = TRUE)

## End(Not run)
```

set_opt *Set Parameters*

Description

Function can be used to set the parameters used in mistr.

Usage

```
set_opt(...)
```

Arguments

...	arguments in tag = value form, or a list of tagged values.

Details

The function can set the values for:

sub parameter: small value that is used in mixture quantile function to test if the computed value is infimum, default: 1e-10.

add parameter: small value that is added to values that are in the image of CDF in qlim function, default: 1e-08.

tol parameter: tolerance for uniroot used in mixture quantile function, default: .Machine$double.eps^0.5.

Value

When parameters are set, their previous values are returned in an invisible named list.

Examples

```
a <- set_opt(sub = 1e-5, tol = 1e-10)
get_opt("sub", "tol")
set_opt(a)
```

stocks *Log-returns of Five Stocks*

Description

A dataset containing the log-returns of adjusted closing prices from 04.01.2007 to 30.10.2017. The dataset contains data of Microsoft, SAP, Adidas, S&P 500 (index) and Dow Jones Industrial Average (index).

Usage

```
stocks
```

Format

A data frame with 2726 rows and 5 variables:

MSFT Microsoft Corporation

SAP Systems, Applications & Products in Data Processing

ADS Adidas

GSPC S&P 500

DJI Dow Jones Industrial Average

Source

quantmod

---

sudo_support *Support Interval of Distribution Object*

Description

sudo_support is a generic function that returns the two boundary values of object's support.

Usage

```
sudo_support(0)

## S3 method for class 'discrdist'
sudo_support(0)

## S3 method for class 'contdist'
sudo_support(0)

## S3 method for class 'mixdist'
sudo_support(0)

## S3 method for class 'compdist'
sudo_support(0)

## S3 method for class 'trans_univdist'
sudo_support(0)
```

Arguments

0	distribution object.

Details

Methods of sudo_support function calculate the support's boundary values for any distribution in the package mistr. This technique is particularly useful when dealing with a transformed distribution.

Value

Named vector containing two values.

Examples

```
B  <- binomdist(10, 0.3)
sudo_support(B)

B2 <- -3*log(B)
sudo_support(B2)

sudo_support( mixdist(B2, normdist(), weights = c(0.5, 0.5)))
```

---

summary.comp_fit *Displays a Useful Description of a Fitted Object*

Description

Displays a useful description of a fitted object.

Usage

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

Arguments

object	distribution object to summarize.
...	additional arguments.

Value

Function returns summary of the fit, offered by bbmle package for class mle2-class.

See Also

mle2-class

tdist *Creates an Object Representing Student-t Distribution*

Description

The function creates an object which represents the Student-t distribution.

Usage

```
tdist(df = 2)
```

Arguments

df	degrees of freedom parameter, default: 2.

Details

See TDist.

Value

Object of class tdist.

See Also

TDist

Examples

```
t <- tdist(2)
d(t, c(2, 3, 4, NA))
r(t, 5)
```

trafo *Modifications of Transformations*

Description

The function modifies the given object and adds the transformation expressions.

Usage

```
trafo(O, type = "new", trans, invtrans, print, deriv, operation,
  value = 0)
```

Arguments

`O`	distribution object.
`type`	type of modification to be performed, default: 'new'.
`trans`	transformation expression.
`invtrans`	inverse transformation expression.
`print`	print expression.
`deriv`	derivative expression.
`operation`	string indicating which operation is performed.
`value`	numeric value used in operation, default: 0.

Details

`trafo` is the main function used in the transformation framework. The function offers four types of possible modifications. Note, that all expressions must use X to indicate the object in the transformation.

type = "init": Initializes the needed lists for transformations and adds the first expressions. This type should be used only with yet untransformed distributions as the first modification. All arguments must be set.

type = "new": Adds a new transformation to the current ones. This must be used only on already transformed distributions. All arguments must be set.

type = "update": Updates previous expression. This is useful when same transformation is used twice in a row. All arguments except operation must be set.

type = "go_back": Uses to history to reverse the previous transformation. Useful if inverse of previous transformation is evaluated. Only object and type must be specified.

Value

Transformed distribution object.

Examples

```
#init
P <- poisdist(5) ; x <- 5
P2 <- trafo(P, type = "init", trans = bquote(X + .(x)),
            invtrans = bquote(X - .(x)), print = bquote(X + .(x)),
            deriv = quote(1), operation = "+", value = x)
P2

#new
x = 3
P3 <- trafo(P2, type = "new", trans = bquote(.(x) * X),
            invtrans = bquote(X/.(x)), print = bquote(.(x) * X),
            deriv = bquote(1/.(x)), operation = "*", value = x)
P3

#update
```

```
x = 7
P4 <- trafo(P3, type = "update", trans = bquote(.(x) * X),
            invtrans = bquote(X/.(x)), print = bquote(.(x) * X),
            deriv = bquote(1/.(x)), value = x)
P4

#go_back
P5 <- trafo(P4, type = "go_back")
P5
```

`unifdist` *Creates an Object Representing Uniform Distribution*

Description

The function creates an object which represents the uniform distribution.

Usage

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

Arguments

`min`	minimum parameter, default: 0.
`max`	maximum parameter, default: 1.

Details

See `Uniform`.

Value

Object of class unifdist.

See Also

`Uniform`

Examples

```
U <- unifdist(1, 5)
d(U, c(2, 3, 4, NA))
r(U, 5)
```

untrafo *Untransformation of a Distribution Object*

Description

untrafo is a generic function that returns the untransformed random variable, if a transformed object is given.

Usage

```
untrafo(0)

## S3 method for class 'trans_standist'
untrafo(0)

## S3 method for class 'trans_mixdist'
untrafo(0)

## S3 method for class 'trans_compdist'
untrafo(0)
```

Arguments

0	transformed distribution object.

Value

Untransformed distribution object.

Examples

```
B  <- binomdist(10, 0.3)
B2 <- -3*log(B)
B2

untrafo(B2)
```

weibulldist *Creates an Object Representing Weibull Distribution*

Description

The function creates an object which represents the Weibull distribution.

Usage

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

Arguments

shape	shape parameter, default: 1.
scale	scale parameter, default: 1.

Details

See Weibull.

Value

Object of class weibulldist.

See Also

Weibull

Examples

```
W <- weibulldist(1, 1)
d(W, c(2, 3, 4, NA))
r(W, 5)
```

---

wilcoxdist *Creates an Object Representing Wilcoxon Distribution*

Description

The function creates an object which represents the Wilcoxon distribution.

Usage

```
wilcoxdist(m, n)
```

Arguments

m	number of observations in the first sample.
n	number of observations in the second sample.

Details

See Wilcoxon.

Value

Object of class wilcoxdist.

See Also

Wilcoxon

Examples

```
W <- wilcoxdist(20, 15)
d(W, c(2, 3, 4, NA))
r(W, 5)
```

Index