Package ‘mistr’

November 23, 2020

**Type**  Package
**Title**  Mixture and Composite Distributions
**Version**  0.0.4
**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**  7.0.2
**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**  2020-11-23 16:10:02 UTC

**R topics documented:**

- mistr-package .................................................. 3
- autoplot.comp_fit ........................................ 4
- autoplot.dist ............................................. 5
- betadist .................................................... 6
- binomdist .................................................. 6
- breakpoints ............................................... 7
- Burr ......................................................... 8
- burrdist ................................................... 9
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cauchydist</td>
<td>10</td>
</tr>
<tr>
<td>chisqdist</td>
<td>11</td>
</tr>
<tr>
<td>compdist</td>
<td>11</td>
</tr>
<tr>
<td>d.compdist</td>
<td>13</td>
</tr>
<tr>
<td>distribution</td>
<td>14</td>
</tr>
<tr>
<td>Distribution autoplot</td>
<td>15</td>
</tr>
<tr>
<td>Distribution summary</td>
<td>18</td>
</tr>
<tr>
<td>Distribution transformation</td>
<td>19</td>
</tr>
<tr>
<td>expdist</td>
<td>19</td>
</tr>
<tr>
<td>fdist</td>
<td>22</td>
</tr>
<tr>
<td>Frechet</td>
<td>23</td>
</tr>
<tr>
<td>frechetdist</td>
<td>24</td>
</tr>
<tr>
<td>gammadist</td>
<td>25</td>
</tr>
<tr>
<td>geomdist</td>
<td>26</td>
</tr>
<tr>
<td>get_opt</td>
<td>26</td>
</tr>
<tr>
<td>GNG_fit</td>
<td>27</td>
</tr>
<tr>
<td>GPD</td>
<td>28</td>
</tr>
<tr>
<td>GPDdist</td>
<td>29</td>
</tr>
<tr>
<td>Gumbel</td>
<td>30</td>
</tr>
<tr>
<td>gumbeldist</td>
<td>31</td>
</tr>
<tr>
<td>hyperdist</td>
<td>32</td>
</tr>
<tr>
<td>is.composite</td>
<td>33</td>
</tr>
<tr>
<td>is.contin</td>
<td>34</td>
</tr>
<tr>
<td>is.discrete</td>
<td>35</td>
</tr>
<tr>
<td>is.dist</td>
<td>35</td>
</tr>
<tr>
<td>is.mixture</td>
<td>35</td>
</tr>
<tr>
<td>is.standard</td>
<td>36</td>
</tr>
<tr>
<td>is.transformed</td>
<td>36</td>
</tr>
<tr>
<td>jumps</td>
<td>36</td>
</tr>
<tr>
<td>last_history</td>
<td>37</td>
</tr>
<tr>
<td>lnormdist</td>
<td>38</td>
</tr>
<tr>
<td>mistr_d_p_q_r</td>
<td>39</td>
</tr>
<tr>
<td>mistr_theme</td>
<td>39</td>
</tr>
<tr>
<td>mixdist</td>
<td>40</td>
</tr>
<tr>
<td>monot</td>
<td>41</td>
</tr>
<tr>
<td>multinomdist</td>
<td>42</td>
</tr>
<tr>
<td>nbinomdist</td>
<td>43</td>
</tr>
<tr>
<td>new_dist</td>
<td>43</td>
</tr>
<tr>
<td>normdist</td>
<td>44</td>
</tr>
<tr>
<td>p.compdist</td>
<td>46</td>
</tr>
<tr>
<td>parameters</td>
<td>46</td>
</tr>
<tr>
<td>Pareto</td>
<td>48</td>
</tr>
<tr>
<td>paretodist</td>
<td>49</td>
</tr>
<tr>
<td>plim.compdist</td>
<td>50</td>
</tr>
<tr>
<td>plot.comp_fit</td>
<td>51</td>
</tr>
<tr>
<td>plotgg</td>
<td>52</td>
</tr>
<tr>
<td>PNP_fit</td>
<td>53</td>
</tr>
<tr>
<td>poisdist</td>
<td>61</td>
</tr>
</tbody>
</table>
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>
Description

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

Usage

```r
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

A 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`
Description

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

Usage

```r
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

- ggpplot 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

```r
## 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(0)

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

## S3 method for class 'comp_fit'
breakpoints(0)

Arguments

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

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.
The Burr distribution function with shape1 parameter c and shape2 parameter k has density given by
\[ f(x) = \frac{ckx^{c-1}}{(1 + x^c)^{k+1}} \]
for \( x > 0 \). The cumulative distribution function is
\[ F(x) = 1 - (1 + x^c)^{-k} \]
on \( x > 0 \).

Value
dburr gives the density, pburr gives the distribution function, qburrr gives the quantile function, and rburr generates random deviates.
Invalid arguments will result in return value NaN, with a warning.

See Also
- `burrdist`

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

Arguments
- `shape1`: shape parameter, default: 2.
- `shape2`: shape parameter, default: 2.

Details
See Burr.
cauchydist

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**

```r
cchisqdist(df = 2)
```

**Arguments**

- `df` degrees of freedom parameter, default: 2.

**Details**

See Chisquare.

**Value**

Object of class chisqdist.

**See Also**

Chisquare

**Examples**

```r
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**

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

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

## Default S3 method:
```r
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 \((-\infty, \infty)\) 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 \(\infty\) 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)
**distribution**

**Arguments**

- **0** 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[suffix]` 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**

```r
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**

```r
distribution(O)
```

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

**Arguments**

- **0** an object for which the extraction of distribution is meaningful.
Value

Object representing the distribution.

Description

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

Usage

```r
## 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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 \leq 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).

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))
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

```r
## 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'
\( e_1 \cdot e_2 \)

## S3 method for class 'normdist'
\( e_1 + e_2 \)

## S3 method for class 'normdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'normdist'
\( \text{exp}(x) \)

## S3 method for class 'expdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'expdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'unifdist'
\( e_1 + e_2 \)

## S3 method for class 'unifdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'tdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'fdist'
\( e_1 \cdot e_2 \)

## S3 method for class 'betadist'
\( e_1 - e_2 = \text{NULL} \)

## S3 method for class 'binomdist'
\( e_1 - e_2 = \text{NULL} \)

## S3 method for class 'gammadist'
\( e_1 \cdot e_2 \)

## S3 method for class 'cauchydist'
\( e_1 + e_2 \)

## S3 method for class 'cauchydist'
\( e_1 \cdot e_2 \)

## S3 method for class 'cauchydist'
\( e_1 \cdot e_2 \)
## 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

```r
# 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**

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

**Arguments**

- **rate**: rate parameter, default: 1.
**fdist**

Details

See Exponential.

Value

Object of class expdist.

See Also

Exponential

Examples

```r
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

```r
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

```r
f <- fdist(2, 2)
d(f, c(2, 3, 4, NA))
r(f, 5)
```
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 are given as log(p), default: FALSE.
lower.tail logical; if TRUE, probabilities are P[X ≤ 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

\[ f(x) = \frac{\alpha}{sz^{(\alpha - 1)}} e^{-z^{\alpha}} \]

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.


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.
frechetdist

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 \hspace{1cm} Creates an Object Representing Gamma Distribution

**Description**

The function creates an object which represents the gamma distribution.

**Usage**

\begin{verbatim}
gammadist(shape = 2, rate, scale)
\end{verbatim}

**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**

\begin{verbatim}
G <- gammadist(shape = 2, scale = 3)
d(G, c(2, 3, 4, NA))
r(G, 5)
\end{verbatim}

geomdist \hspace{1cm} Creates an Object Representing Geometric Distribution

**Description**

The function creates an object which represents the geometric distribution.

**Usage**

\begin{verbatim}
geomdist(prob = 0.5)
\end{verbatim}

**Examples**

\begin{verbatim}
G <- geomdist(prob = 0.5)
d(G, c(2, 3, 4, NA))
\end{verbatim}
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

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")
**Fitting a GPD-Normal-GPD Model**

**Description**

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

**Usage**

```r
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.
## 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

```r
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.

### Examples

```r
## 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)
```
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 \( \mu \), scale parameter \( \sigma \) and shape parameter \( \xi \) has density given by

\[
f(x) = \frac{1}{\sigma} (1 + \xi z)^{-\left(1/\xi + 1\right)}
\]

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

\[
F(x) = 1 - (1 + \xi z)^{-\left(1/\xi + 1\right)}
\]

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 \( F(x) = 1 - e^{-z} \).


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)
pGPD(pGPD(seq(1, 5), 0, 1, 1), 0, 1, 1)
rGPD(5, 0, 1, 1)
**Gumbel**

**The Gumbel Distribution**

**Description**

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

**Usage**

\[
\begin{align*}
& \text{dgumbel}(x, \text{loc}, \text{scale}, \log = \text{FALSE}) \\
& \text{pgumbel}(q, \text{loc}, \text{scale}, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
& \text{qgumbel}(p, \text{loc}, \text{scale}, \text{lower.tail} = \text{TRUE}, \log.p = \text{FALSE}) \\
& \text{rgumbel}(n, \text{loc}, \text{scale})
\end{align*}
\]

**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**

\[
G \leftarrow \text{GPDdist}(0, 1, 0) \\
d(G, c(2, 3, 4, \text{NA})) \\
r(G, 5)
\]
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 \( \mu \) and scale parameter \( \beta \) has density given by

\[
f(x) = \frac{1}{\beta} e^{-\left(\frac{x-\mu}{\beta} + e^{-\frac{x-\mu}{\beta}}\right)}
\]

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

\[
F(x) = e^{\left(\frac{x-\mu}{\beta} - e^{-\frac{x-\mu}{\beta}}\right)}
\]

with \( z \) as stated above.


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**

```r
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**

```r
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**

```r
hyperdist(m = 10, n = 10, k = 5)
```
is.composite

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.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

0 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

0 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

0 an object to test.
jumps

<table>
<thead>
<tr>
<th>Probability mass points</th>
</tr>
</thead>
</table>

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

```r
jumps(O, interval)
```

## S3 method for class 'discrdist'

```r
jumps(O, interval)
```

## S3 method for class 'trans_discrdist'

```r
jumps(O, interval)
```

## S3 method for class 'contdist'

```r
jumps(O, interval)
```

## S3 method for class 'trans_contdist'

```r
jumps(O, interval)
```

## S3 method for class 'mixdist'

```r
jumps(O, interval)
```

## S3 method for class 'trans_mixdist'

```r
jumps(O, interval)
```

## S3 method for class 'compdist'

```r
jumps(O, interval)
```

## S3 method for class 'trans_compdist'

```r
jumps(O, interval)
```

Arguments

- **O** 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

```r
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

```r
last_history(O, t)
```

Arguments

- **O**
  - 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

```r
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**

```r
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**

```r
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.
`mixdist`  

Value

ggplot theme.

See Also

`theme`

---

### mixdist

**Creates an Object Representing Mixture Distribution**

**Description**

mixdist creates an object which represents the mixture distribution.

**Usage**

```r
mixdist(..., weights)
```

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

```r
## 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(O)

## S3 method for class 'trans_univdist'
monot(O)

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)
new_dist

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 environment. 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

```r
## 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**

```r
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**

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

---

**p.comdist**  
*Distribution Function*

**Description**

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

```r
## 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)

## 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

- `0`: 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 \( p[suffix] \) 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

```r
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 mstr distribution objects.

**Usage**

```r
parameters(O)
```

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

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

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

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

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

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

## S3 method for class 'comp_fit'
parameters(O)

**Arguments**

- `O` 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

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

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

---

**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 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 \( \alpha \) 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](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)
ppareto(pppareto(seq(1, 5), 1, 1), 1,1)
rpareto(5, 1, 1)

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)
Description

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

Usage

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

## S3 method for class 'trans_discrmixdist'  
plotgg(  
  x,  
  which = "all",  
  only_mix = FALSE,
plotgg

plotgg(x, which = "all", only_mix = FALSE, pp1 = 1000, pp2 = 1000, col = "+F9D607", xlim1 = q(x, c(0.01, 0.99)), ylim1 = 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", 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,
  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 'comdist'
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,
  xlab1 = NULL,
  ylab1 = NULL,
  main1 = "CDF",
  size1 = 1.6,
  alpha1 = 0.4,
  legend.position1 = "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.
number of points at which PDF is evaluated, default: 1000.
color used in plot, default: '#122e94'.
xlim of CDF plot, default: \(q(x, c(0.01, 0.99))\).
ylim of CDF plot, default: NULL.
xlim of PDF plot, default: xlim1.
ylim of PDF plot, default: NULL.
xlab of CDF plot, default: NULL.
ylab of CDF plot, default: NULL.
xlab of PDF plot, default: NULL.
ylab of PDF plot, default: NULL.
title of CDF plot, default: 'CDF'.
title of PDF plot, default: 'PDF'/PMF'.
size used in CDF plot.
size used in PDF plot.
alpha used in CDF plot.
alpha used in PDF plot.
col of additional segment if contained in the plot (composite and discrete distributions).
whether to plot only mixture/composite model and not also the components, default: FALSE.
legend.position used in CDF plot.
legend.position used in PDF plot.
width of the bars that are used to plot discrete mixtures, default: 0.25.
y coordinate for text annotation, default: -0.01.
ltty of additional segment if contained in the plot (composite and discrete distributions).
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
```r
## 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**

```r
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`
poisdist

Examples

```r
## 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

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

Arguments

- `lambda` mean parameter, default: 1.

Details

See `Poisson`.

Value

Object of class poisdist.

See Also

- `Poisson`

Examples

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

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

Usage

```r
## S3 method for class 'compdist'
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 \( q[\text{sufix}] \) 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

```r
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

```r
## 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 issue with the overwriting of `q()`.

See Also

`q`
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 misstr. The function makes use of the p[suffix] and q[suffix] 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 \( p - \text{sub} \).

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

Value

Vector of computed results.

See Also

- `set_opt`
Examples

```r
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

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

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

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

```r
## 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

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

### Usage

```r
## 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

\[ Q(x^+) = \inf x : p < P(X \leq x). \]

The function makes use of the p[sufix] and q[sufix] functions as pnorm, pbeta, qnorm, qbeta, and thus, if a new distribution 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'.
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",
  )

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

```r
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 = ylab,
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

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)
Description

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

Usage

```r
## S3 method for class 'compdist'
\r(0, n)

## S3 method for class 'mixdist'
\r(0, n)

\r(0, n)

## S3 method for class 'standist'
\r(0, n)

## S3 method for class 'hyperdist'
\r(0, n)

## S3 method for class 'wilcoxdist'
\r(0, n)

## S3 method for class 'trans_univdist'
\r(0, 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[suffix] 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

```r
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

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

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

```r
## S3 method for class 'GNG'
risk(
  model,
  alpha = 0.05,
  expectile = TRUE,
```
### 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 \( \tau \) and \( \alpha \) if

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

where \( \mu \) is mean, \( q(\cdot) \) is the quantile function and \( G(\alpha) = \int_{-\infty}^{\alpha} y dF(y) \).

### Value

List of class risk_measures.

### Examples

```r
## 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**

```r
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**

```r
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**

```r
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

```r
sudo_support(0)
```

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

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

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

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

## S3 method for class 'trans_univdist'
```r
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

```r
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

```r
## 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**

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

**Arguments**

- `df` degrees of freedom parameter, default: 2.

**Details**

See `TDist`.

**Value**

Object of class `tdist`.

**See Also**

`TDist`

**Examples**

```r
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**

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

- **0**
  - 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

```r
#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
```
\begin{verbatim}
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
\end{verbatim}

---

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

\begin{verbatim}
U <- unifdist(1, 5)
d(U, c(2, 3, 4, NA))
r(U, 5)
\end{verbatim}
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(O)

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

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

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

Arguments

O  
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)
**wilcoxdist**

*Arguments*

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

*Details*

See [Weibull](#).

*Value*

Object of class `weibulldist`.

*See Also*

- [Weibull](#)

*Examples*

```r
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*

```r
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`.  

---

*Description*

The function creates an object which represents the Wilcoxon distribution.

*Usage*

```r
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`.  

---

*Description*

The function creates an object which represents the Wilcoxon distribution.

*Usage*

```r
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

* datasets
  stocks, 77
*.GPDbist
  (Distribution_transformation), 19
*.cauchydist
  (Distribution_transformation), 19
*.expdist
  (Distribution_transformation), 19
*.frechetdist
  (Distribution_transformation), 19
*.gammadist
  (Distribution_transformation), 19
*.gumbeldist
  (Distribution_transformation), 19
*.lnormdist
  (Distribution_transformation), 19
*.normdist
  (Distribution_transformation), 19
*.paretodist
  (Distribution_transformation), 19
*.trans_univdist
  (Distribution_transformation), 19
*.unidist
  (Distribution_transformation), 19
*.univdist
  (Distribution_transformation), 19
*.weibulldist
  (Distribution_transformation), 19
+.GPDbist
  (Distribution_transformation), 19
+.cauchydist
  (Distribution_transformation), 19
+.frechetdist
  (Distribution_transformation), 19
+.gumbeldist
  (Distribution_transformation), 19
+.normdist
  (Distribution_transformation), 19
+.trans_univdist
  (Distribution_transformation), 19
+.unidist
  (Distribution_transformation), 19
+.univdist
  (Distribution_transformation), 19
-.betadist
  (Distribution_transformation), 19
-.binomdist
  (Distribution_transformation), 19
-.dist
  (Distribution_transformation), 19
/.dist
  (Distribution_transformation), 19
^cauchydist
  (Distribution_transformation), 19
^expdist
  (Distribution_transformation), 86
\simpleindex

\begin{itemize}
\item ^.fdist (Distribution_transformation), 19
\item ^.lnormdist
  (Distribution_transformation), 19
\item ^.tdist (Distribution_transformation), 19
\item ^.trans_univdist
  (Distribution_transformation), 19
\item ^.univdist
  (Distribution_transformation), 19
\item \texttt{approx}, 73
\item \texttt{autoplot.comp_fit}, 4
\item \texttt{autoplot.dist}, 5
\item \texttt{Beta}, 6
\item \texttt{betadist}, 6
\item \texttt{binomdist}, 6
\item \texttt{Binomial}, 7
\item \texttt{breakpoints}, 7, 48
\item \texttt{Burr}, 8, 9, 10
\item \texttt{burrdist}, 9, 9
\item \texttt{Cauchy}, 10
\item \texttt{cauchydist}, 10
\item \texttt{chisqdist}, 11
\item \texttt{Chisquare}, 11
\item \texttt{compdist}, 11, 42
\item \texttt{d}, 39
\item \texttt{d (d.compdist)}, 13
\item \texttt{d.compdist}, 13
\item \texttt{dburr (Burr)}, 8
\item \texttt{dfrechet (Frechet)}, 24
\item \texttt{dGPD (GPD)}, 29
\item \texttt{dgumbel (Gumbel)}, 31
\item \texttt{distribution}, 14
\item \texttt{Distributionautoplot}, 15, 52
\item \texttt{Distribution_summary}, 18
\item \texttt{Distribution_transformation}, 19
\item \texttt{dpareto (Pareto)}, 49
\item \texttt{exp.normdist}
  (Distribution_transformation), 19
\item \texttt{exp.univdist}
  (Distribution_transformation), 19
\item \texttt{expdist}, 22
\item \texttt{Exponential}, 23
\item \texttt{FDist}, 23
\item \texttt{fdist}, 23
\item \texttt{Frechet}, 24, 25
\item \texttt{frechetdist}, 25, 25
\item \texttt{GammaDist}, 26
\item \texttt{gammadist}, 26
\item \texttt{geomdist}, 26
\item \texttt{Geometric}, 27
\item \texttt{get_opt}, 27
\item \texttt{GNG\_fit}, 28
\item \texttt{GPD}, 29, 31
\item \texttt{GPDdist}, 30, 30
\item \texttt{Gumbel}, 31, 33
\item \texttt{gumbeldist}, 32, 33
\item \texttt{hyperdist}, 33
\item \texttt{Hypergeometric}, 34
\item \texttt{is.composite}, 34
\item \texttt{is.contin}, 35
\item \texttt{is.discrete}, 35
\item \texttt{is.dist}, 35
\item \texttt{is.mixture}, 36
\item \texttt{is.standard}, 36
\item \texttt{is.transformed}, 36
\item \texttt{jumps}, 37
\item \texttt{last_history}, 38
\item \texttt{lngdist}, 39
\item \texttt{log.lnormdist}
  (Distribution_transformation), 19
\item \texttt{log.trans_univdist}
  (Distribution_transformation), 19
\item \texttt{log.univdist}
  (Distribution_transformation), 19
\item \texttt{Lognormal}, 39
\end{itemize}
INDEX

mistr, 7, 14, 18, 27, 47, 48, 51, 63, 65, 66, 68, 74, 77, 78
mistr (mistr-package), 3
mistr-package, 3
mistr_d (mistr_d_p_q_r), 39
mistr_d_p_q_r, 39
mistr_p (mistr_d_p_q_r), 39
mistr_q (mistr_d_p_q_r), 39
mistr_r (mistr_d_p_q_r), 39
mistr_theme, 40
mixdist, 12, 41
mixtures, 66
mle2, 28, 29, 61
monot, 42
multinomdist, 43
Multinomial, 43

nbinomdist, 43
NegBinomial, 44
new_dist, 44
Normal, 46
normdist, 45, 46

p, 39
p (p.compdist), 46
p.compdist, 46
parameters, 8, 48
Pareto, 49, 50
paretodist, 50, 50
pburr (Burr), 8
pfrechet (Frechet), 24
pGPD (GPD), 29
pgumbel (Gumbel), 31
qlim (qlim.compdist), 51
qlim.compdist, 51
plot, 37
plot.comp_fit, 52
plot.compdist (Distribution_autoplot), 15
plot.contdiscr mixdist (Distribution_autoplot), 15
plot.contdist (Distribution_autoplot), 15
plot.contmixdist (Distribution_autoplot), 15
plot.compdist (Distribution_autoplot), 15
plot.discr dist (Distribution_autoplot), 15
plot.discrmixdist (Distribution_autoplot), 15
plottranscompdist

plot_contdiscr mixdist (Distribution_autoplot), 15
plot_contdist (Distribution_autoplot), 15
plot_contmixdist (Distribution_autoplot), 15
plot_transdiscr dist (Distribution_autoplot), 15
plot_transdiscr mixdist (Distribution_autoplot), 15
plotgg, 4, 5, 37, 53
PNP_fit, 61
poisdist, 62
Poisson, 62
ppareto (Pareto), 49
q, 39, 64, 65, 73
q (q.compdist), 63
q.compdist, 63
q.default, 63, 64, 65
q.mixdist, 63, 65
q.approxfun, 73
qburr (Burr), 8
qfrechet (Frechet), 24
qGPD (GPD), 29
qgumbel (Gumbel), 31
qlim, 68, 77
qlim (qlim.compdist), 66
qlim.compdist, 66
qlim.contdistmixdist (qlim.discrimmixdist), 67
qlim.contmixdist (qlim.discrimmixdist), 67
qlim.discrimmixdist, 67
qpareto (Pareto), 49
QQnorm (QQplot), 68
QQnormg (QQplotgg), 70
QQplot, 68
QQplotgg, 70
quantmod, 78

r, 39
r (r.compdist), 74
r.compdist, 74
rburr (Burr), 8
rfrechet (Frechet), 24
rGPD (GPD), 29
rgumbel (Gumbel), 31
INDEX

risk, 75
rpareto (Pareto), 49

set_opt, 27, 65, 68, 77
sqrt.dist
(Distribution_transformation), 19
stocks, 77
sudo_support, 78
summary.comp_fit, 79
summary.compdist
(Distribution_summary), 18
summary.mixdist (Distribution_summary), 18
summary.standist
(Distribution_summary), 18
summary.trans_compdist
(Distribution_summary), 18
summary.trans_mixdist
(Distribution_summary), 18
summary.trans_standist
(Distribution_summary), 18

TDist, 80
tdist, 80
theme, 41
trafo, 80

unifdist, 45, 82
Uniform, 82
uniroot, 65, 68
untrafo, 83

Weibull, 84
weibulldist, 83
weights, 8, 48
wilcoxdist, 84
Wilcoxon, 84, 85