Package ‘fitdistrplus’

September 10, 2018

Title Help to Fit of a Parametric Distribution to Non-Censored or Censored Data

Version 1.0-11

Description Extends the fitdistr() function (of the MASS package) with several functions to help the fit of a parametric distribution to non-censored or censored data. Censored data may contain left censored, right censored and interval censored values, with several lower and upper bounds. In addition to maximum likelihood estimation (MLE), the package provides moment matching (MME), quantile matching (QME) and maximum goodness-of-fit estimation (MGE) methods (available only for non-censored data). Weighted versions of MLE, MME and QME are available. See e.g. Casella & Berger (2002). Statistical inference. Pacific Grove.

Depends R (>= 3.2.0), MASS, grDevices, survival, methods, npsurv

Imports stats

Suggests actuar, rgenoud, mc2d, gamlss.dist, knitr, knitcitations, ggplot2, GeneralizedHyperbolic

VignetteBuilder knitr

BuildVignettes true

License GPL (>= 2)

Encoding UTF-8


Contact Marie-Laure Delignette-Muller <marielaure.delignettemuller at vetagro-sup.fr> or Christophe Dutang <christophe.dutang at ensimag.fr>

Author Marie-Laure Delignette-Muller [aut], Christophe Dutang [aut], Regis Pouillot [ctb], Jean-Baptiste Denis [ctb], Aurelie Siberchicot [aut, cre]

Maintainer Aurelie Siberchicot <aurelie.siberchicot@univ-lyon1.fr>
R topics documented:

fitdistplus-package .................................................. 3
bootdist ................................................................. 4
bootdistcens ........................................................... 7
CIcdfplot ................................................................. 11
danish ................................................................. 14
dataFAQ ................................................................. 16
descdist ................................................................. 16
detectbound ............................................................. 19
dosulfan ................................................................. 20
fitdist ................................................................. 22
fitdistcens ............................................................. 32
fluazinam ............................................................... 37
gofstat ................................................................. 39
graphcomp ............................................................. 42
graphcompccens ....................................................... 48
groundbeef ............................................................ 51
logLikplot .............................................................. 52
logLiksurface ........................................................ 54
mgedist ................................................................. 57
mledist ................................................................. 60
mmmedist .............................................................. 65
plotdist ............................................................... 69
plotdistcens .......................................................... 71
prefit ................................................................. 74
qmedist ................................................................. 76
quantiles ............................................................... 79
salinity ................................................................. 82
smokedfish ............................................................ 84
toxocara ............................................................... 85

Index 86
Description

The idea of this package emerged in 2008 from a collaboration between JB Denis, R Pouillot and ML Delignette who at this time worked in the area of quantitative risk assessment. The implementation of this package was a part of a more general project named "Risk assessment with R" gathering different packages and hosted in R-forge.

The fitdistrplus package was first written by ML Delignette-Muller and made available in CRAN on 2009 and presented at the 2009 useR conference in Rennes. A few months after, C Dutang joined the project by starting to participate to the implementation of the fitdistrplus package. The package has also been presented at 2011 useR conference, 2eme rencontres R in 2013.

Three vignettes are available within the package:
- a general overview of the package published in the Journal of Statistical Software,
- a html document answering the most Frequently Asked Questions,
- a html document presenting a benchmark of optimization algorithms when finding parameters.

The fitdistrplus package is a general package that aims at helping the fit of univariate parametric distributions to censored or non-censored data. The two main functions are fitdist for fit on non-censored data and fitdistcens for fit on censored data.

The choice of candidate distributions to fit may be helped using functions descdist and plotdist for non-censored data and plotdistcens for censored data).

Using functions fitdist and fitdistcens, different methods can be used to estimate the distribution parameters:
- maximum likelihood estimation by default (mledist),
- moment matching estimation (mmedist),
- quantile matching estimation (qmedist),
- maximum goodness-of-fit estimation (mgmedist).

For classical distributions initial values are automatically calculated if not provided by the user. Graphical functions plotdist and plotdistcens can be used to help a manual calibration of initial values for parameters of non-classical distributions. Function prefit is proposed to help the definition of good starting values in the special case of constrained parameters. In the case where maximum likelihood is chosen as the estimation method, function llplot enables to visualize log-likelihood contours.

The goodness-of-fit of fitted distributions (a single fit or multiple fits) can be explored using different graphical functions (cdfcomp, denscomp, qqcomp and ppcomp for non-censored data and cdfcompcens for censored data). Goodness-of-fit statistics are also provided for non-censored data using function gofstat.

Bootstrap is proposed to quantify the uncertainty on parameter estimates (functions bootdist and bootdistcens) and also to quantify the uncertainty on CDF or quantiles estimated from the fitted distribution (quantile and Cldfplot).
Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

bootdist

Bootstrap simulation of uncertainty for non-censored data

Description

Uses parametric or nonparametric bootstrap resampling in order to simulate uncertainty in the parameters of the distribution fitted to non-censored data.

Usage

bootdist(f, bootmethod = "param", niter = 1001, silent = TRUE, 
        parallel = c("no", "snow", "multicore"), ncpus) 

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

## S3 method for class 'bootdist'
plot(x, main = "Bootstrapped values of parameters", enhance = FALSE, 
     trueval = NULL, rampcol = NULL, nbgrid = 100, nbcol = 100, ...)

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

Arguments

f

An object of class "fitdist", output of the fitdist function.

bootmethod

A character string coding for the type of resampling : "param" for a parametric resampling and "nonparam" for a nonparametric resampling of data.

niter

The number of samples drawn by bootstrap.

silent

A logical to remove or show warnings and errors when bootstraping.

parallel

The type of parallel operation to be used, "snow" or "multicore" (the second one not being available on Windows), or "no" if no parallel operation.

ncpus

Number of processes to be used in parallel operation : typically one would fix it to the number of available CPUs.

x

An object of class "bootdist".

object

An object of class "bootdist".

main

an overall title for the plot: see title, default to "Bootstrapped values of parameters".

enhance

a logical to get an enhanced plot.

trueval

when relevant, a numeric vector with the true value of parameters (for backfitting purposes).

rampcol

colors to interpolate; must be a valid argument to colorRampPalette().
bootdist

### nbgrid
Number of grid points in each direction. Can be scalar or a length-2 integer vector.

### nbcol
an integer argument, the required number of colors

... Further arguments to be passed to generic methods

### Details

Samples are drawn by parametric bootstrap (resampling from the distribution fitted by `fitdist`) or nonparametric bootstrap (resampling with replacement from the data set). On each bootstrap sample the function `mledist` (or `mmedist`, `qmedist`, `mgedist`) according to the component `f$method` of the object of class "fitdist") is used to estimate bootstrapped values of parameters. When that function fails to converge, NA values are returned. Medians and 2.5 and 97.5 percentiles are computed by removing NA values. The medians and the 95 percent confidence intervals of parameters (2.5 and 97.5 percentiles) are printed in the summary. If inferior to the whole number of iterations, the number of iterations for which the function converges is also printed in the summary.

By default (when `enhance`=FALSE), the plot of an object of class "bootdist" consists in a scatterplot or a matrix of scatterplots of the bootstrapped values of parameters. It uses the function `stripchart` when the fitted distribution is characterized by only one parameter, the function `plot` when there are two parameters and the function `pairs` in other cases. In these last cases, it provides a representation of the joint uncertainty distribution of the fitted parameters.

When `enhance`=TRUE, a personalized plot version of `pairs` is used where upper graphs are scatterplots and lower graphs are heatmap image using `image` based on a kernel based estimator for the 2D density function (using `kde2d` from MASS package). Arguments `rampcol`, `nbgrid`, `nbcol` can be used to customize the plots. Default values are `rampcol=c("green", "yellow", "orange", "red")`, `nbcol=100` (see `colorRampPalette()`), `nbgrid=100` (see `kde2d`). In addition, when fitting parameters on simulated datasets for backtesting purposes, an additional argument `trueval` can be used to plot a cross at the true value.

It is possible to accelerate the bootstrap using parallelization. We recommend you to use `parallel = "multicore"`, or `parallel = "snow"` if you work on Windows, and to fix `ncpus` to the number of available processors.

### Value

`bootdist` returns an object of class "bootdist", a list with 6 components,

- `estim` a data frame containing the bootstrapped values of parameters.
- `converg` a vector containing the codes for convergence obtained if an iterative method is used to estimate parameters on each bootstrapped data set (and 0 if a closed formula is used).
- `method` A character string coding for the type of resampling : "param" for a parametric resampling and "nonparam" for a nonparametric resampling.
- `nbboot` The number of samples drawn by bootstrap.
- `CI` bootstrap medians and 95 percent confidence percentile intervals of parameters.
- `fitpart` The object of class "fitdist" on which the bootstrap procedure was applied.
print  The print of a "bootdist" object shows the bootstrap parameter estimates. If inferior to the whole number of bootstrap iterations, the number of iterations for which the estimation converges is also printed.

summary  The summary provides the median and 2.5 and 97.5 percentiles of each parameter. If inferior to the whole number of bootstrap iterations, the number of iterations for which the estimation converges is also printed in the summary.

plot  The plot shows the bootstrap estimates with stripchart function for univariate parameters and plot function for multivariate parameters.

Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References

See Also
See fitdistrplus for an overview of the package. fitdist, mledist, qmedist, mmedist, mgdist, quantile.bootdist for another generic function to calculate quantiles from the fitted distribution and its bootstrap results and CICdfplot for adding confidence intervals on quantiles to a CDF plot of the fitted distribution.

Examples
# We choose a low number of bootstrap replicates in order to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.

# (1) Fit of a gamma distribution to serving size data
# using default method (maximum likelihood estimation)
# followed by parametric bootstrap
#
data(groundbeef)
x1 <- groundbeef$serving
f1 <- fitdist(x1, "gamma")
b1 <- bootdist(f1, niter=51)
print(b1)
plot(b1)
plot(b1, enhance=TRUE)
summary(b1)
quantile(b1)
CICdfplot(b1, CI.output = "quantile")

# (2) non parametric bootstrap on the same fit
# (3) Fit of a normal distribution on acute toxicity values of endosulfan in log10 for
# nonarthropod invertebrates, using maximum likelihood estimation
# to estimate what is called a species sensitivity distribution
# (SSD) in ecotoxicology, followed by estimation of the 5 percent quantile value of
# the fitted distribution, what is called the 5 percent hazardous concentration (HCS)
# in ecotoxicology, with its two-sided 95 percent confidence interval calculated by
# parametric bootstrap
#
data(endosulfan)
ATV <- subset(endosulfan, group == "NonArthroInvert")$ATV
log10ATV <- log10(subset(endosulfan, group == "NonArthroInvert")$ATV)
fln <- fitdist(log10ATV, "norm")
bln <- bootdist(fln, bootmethod = "param", niter=51)
quantile(bln, probs = c(0.05, 0.1, 0.2))

# (4) comparison of sequential and parallel versions of bootstrap
# to be tried with a greater number of iterations (1001 or more)
#
## Not run:
niter <- 1001
data(groundbeef)
x1 <- groundbeef$serving
f1 <- fitdist(x1, "gamma")

# sequential version
ptm <- proc.time()
summary(bootdist(f1, niter = niter))
proc.time() - ptm

# parallel version using snow
require(snow)
ptm <- proc.time()
summary(bootdist(f1, niter = niter, parallel = "snow", ncpus = 4))
proc.time() - ptm

# parallel version using multicore (not available on Windows)
ptm <- proc.time()
summary(bootdist(f1, niter = niter, parallel = "multicore", ncpus = 4))
proc.time() - ptm

## End(Not run)
Description

Uses nonparametric bootstrap resampling in order to simulate uncertainty in the parameters of the distribution fitted to censored data.

Usage

```R
bootdistcens(f, niter = 1001, silent = TRUE,
    parallel = c("no", "snow", "multicore"), ncpus)
```

Arguments

- `f`: An object of class "fitdistcens", output of the `fitdistcens` function.
- `niter`: The number of samples drawn by bootstrap.
- `silent`: A logical to remove or show warnings and errors when bootstrapping.
- `parallel`: The type of parallel operation to be used, "snow" or "multicore" (the second one not being available on Windows), or "no" if no parallel operation.
- `ncpus`: Number of processes to be used in parallel operation : typically one would fix it to the number of available CPUs.
- `x`: An object of class "bootdistcens".
- `object`: An object of class "bootdistcens".
- `...`: Further arguments to be passed to generic methods.

Details

Samples are drawn by nonparametric bootstrap (resampling with replacement from the data set). On each bootstrap sample the function `mledist` is used to estimate bootstrapped values of parameters. When `mledist` fails to converge, NA values are returned. Medians and 2.5 and 97.5 percentiles are computed by removing NA values. The medians and the 95 percent confidence intervals of parameters (2.5 and 97.5 percentiles) are printed in the summary. If inferior to the whole number of iterations, the number of iterations for which `mledist` converges is also printed in the summary.

The plot of an object of class "bootdistcens" consists in a scatterplot or a matrix of scatterplots of the bootstrapped values of parameters. It uses the function `stripchart` when the fitted distribution is characterized by only one parameter, and the function `plot` in other cases. In these last cases, it provides a representation of the joint uncertainty distribution of the fitted parameters.

It is possible to accelerate the bootstrap using parallelization. We recommend you to use `parallel = "multicore"`, or `parallel = "snow"` if you work on Windows, and to fix `ncpus` to the number of available processors.
bootdistcens

Value

bootdistcens returns an object of class "bootdistcens", a list with 6 components,

- **estim**: a data frame containing the bootstrapped values of parameters.
- **converg**: a vector containing the codes for convergence of the iterative method used to estimate parameters on each bootstrapped data set.
- **method**: A character string coding for the type of resampling: in this case "nonparam" as it is the only available method for censored data.
- **nbboot**: The number of samples drawn by bootstrap.
- **CI**: bootstrap medians and 95 percent confidence percentile intervals of parameters.
- **fitpart**: The object of class "fitdistcens" on which the bootstrap procedure was applied.

Generic functions:

- **print**: The print of a "bootdistcens" object shows the bootstrap parameter estimates. If inferior to the whole number of bootstrap iterations, the number of iterations for which the estimation converges is also printed.
- **summary**: The summary provides the median and 2.5 and 97.5 percentiles of each parameter. If inferior to the whole number of bootstrap iterations, the number of iterations for which the estimation converges is also printed in the summary.
- **plot**: The plot shows the bootstrap estimates with the `stripchart` function for univariate parameters and `plot` function for multivariate parameters.

Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

See **fitdistrplus** for an overview of the package. **fitdistcens, mledist, quantile.bootdistcens** for another generic function to calculate quantiles from the fitted distribution and its bootstrap results and **Cicdfplot** for adding confidence intervals on quantiles to a CDF plot of the fitted distribution.
Examples

# We choose a low number of bootstrap replicates in order to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.

# (1) Fit of a normal distribution to fluazinam data in log10
# followed by nonparametric bootstrap and calculation of quantiles
# with 95 percent confidence intervals
#
# data(fluazinam)
# (d1 <- log10(fluazinam))
# f1 <- fitdistcens(d1, "norm")
# b1 <- bootdistcens(f1, niter = 101)
# summary(b1)
# plot(b1)
# quantile(b1)
# CICdfplot(b1, CI.output = "quantile")

# (2) Estimation of the mean of the normal distribution
# by maximum likelihood with the standard deviation fixed at 1
# using the argument fix.arg
# followed by nonparametric bootstrap
# and calculation of quantiles with 95 percent confidence intervals
#
# f1b <- fitdistcens(d1, "norm", start = list(mean = 1), fix.arg = list(sd = 1))
# b1b <- bootdistcens(f1b, niter = 101)
# summary(b1b)
# plot(b1b)
# quantile(b1b)

# (3) comparison of sequential and parallel versions of bootstrap
# to be tried with a greater number of iterations (1001 or more)
#
## Not run:
# niter <- 1001
# data(fluazinam)
# d1 <- log10(fluazinam)
# f1 <- fitdistcens(d1, "norm")

# sequential version
# ptm <- proc.time()
# summary(bootdistcens(f1, niter = niter))
# proc.time() - ptm

# parallel version using snow
# require(parallel)
# ptm <- proc.time()
# summary(bootdistcens(f1, niter = niter, parallel = "snow", ncpus = 4))
# proc.time() - ptm

# parallel version using multicore (not available on Windows)
```
ptm <- proc.time()
summary(bootdistcens(f1, niter = niter, parallel = "multicore", ncpus = 4))
proc.time() - ptm
```

```
## End(Not run)
```

---

**ciCDFplot**  
Empirical cumulative distribution function with pointwise confidence intervals on probabilities or on quantiles

### Description

ciCDFplot plots the empirical cumulative distribution function with the bootstrapped pointwise confidence intervals on probabilities or on quantiles.

### Usage

```r
ciCDFplot(b, CI.output = "two.sided", CI.type = "probability", CI.level = 0.95, CI.col = "red", CI.lty = 2, CI.fill = NULL, CI.only = FALSE, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab, datapch, datacol, fitlty, fitcol, horizontals = TRUE, verticals = FALSE, use.ppoints = TRUE, do.points = TRUE, a.ppoints = 0.5, linesQ = FALSE, ...)
```

### Arguments

- **b**  
  One "bootdist" object.
- **CI.output**  
  The quantity on which (bootstrapped) bootstrapped confidence intervals are computed: either "probability" or "quantile".
- **CI.type**  
  Type of confidence intervals: either "two.sided" or one-sided intervals ("less" or "greater").
- **CI.level**  
  The confidence level.
- **CI.col**  
  The color of the confidence intervals.
- **CI.lty**  
  The line type of the confidence intervals.
- **CI.fill**  
  A color to fill the confidence area. Default is NULL corresponding to no filling.
- **CI.only**  
  A logical whether to plot empirical and fitted distribution functions or only the confidence intervals. Default to FALSE.
- **xlim**  
  The x-limits of the plot.
- **ylim**  
  The y-limits of the plot.
- **xlogscale**  
  If TRUE, uses a logarithmic scale for the x-axis.
- **ylogscale**  
  If TRUE, uses a logarithmic scale for the y-axis.
- **main**  
  A main title for the plot, see also `title`.
- **xlab**  
  A label for the x-axis, defaults to a description of x.
CIcdfplot

**ylab**
A label for the $y$-axis, defaults to a description of $y$.

**datapch**
An integer specifying a symbol to be used in plotting data points, see also *points* (only for non censored data).

**datacol**
A specification of the color to be used in plotting data points.

**fitcol**
A (vector of) color(s) to plot fitted distributions. If there are fewer colors than fits they are recycled in the standard fashion.

**fitlty**
A (vector of) line type(s) to plot fitted distributions/densities. If there are fewer colors than fits they are recycled in the standard fashion. See also *par*.

**horizontals**
If TRUE, draws horizontal lines for the step empirical cdf function (only for non censored data). See also *plot.stepfun*.

**do.points**
logical; if TRUE, also draw points at the x-locations. Default is true (only for non censored data).

**verticals**
If TRUE, draws also vertical lines for the empirical cdf function. Only taken into account if horizontals=TRUE (only for non censored data).

**use.ppoints**
If TRUE, probability points of the empirical distribution are defined using function *ppoints* as $(1:n - a.ppoints)/(n - 2a.ppoints + 1)$ (only for non censored data). If FALSE, probability points are simply defined as $1:n / n$. This argument is ignored for discrete data.

**a.ppoints**
If use.ppoints=TRUE, this is passed to function *ppoints* (only for non censored data).

**lines01**
A logical to plot two horizontal lines at h=0 and h=1 for cdfcomp.

... Further graphical arguments passed to *matlines* or *polygon*, respectively when CI.fill=FALSE and CI.fill=TRUE.

### Details

CIcdfplot provides a plot of the empirical distribution using *cdfcomp* or *cdfcompcens*, with bootstrapped pointwise confidence intervals on probabilities (y values) or on quantiles (x values). Each interval is computed by evaluating the quantity of interest (probability associated to an x value or quantile associated to an y value) using all the bootstrapped values of parameters to get a bootstrapped sample of the quantity of interest and then by calculating percentiles on this sample to get a confidence interval (classically 2.5 and 97.5 percentiles for a 95 percent confidence level). If CI.fill != NULL, then the whole confidence area is filled by the color CI.fill thanks to the function *polygon*, otherwise only borders are drawn thanks to the function *matline*. Further graphical arguments can be passed to these functions using the three dots arguments ... .

### Author(s)

Christophe Dutang and Marie-Laure Delignette-Muller.

### References

See Also

See also \texttt{cdfcomp}, \texttt{cdfcompcens}, \texttt{bootdist} and \texttt{quantile}.

Examples

# We choose a low number of bootstrap replicates in order to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.

# (1) Fit of an exponential distribution
#

set.seed(123)
s1 <- rexp(50, 1)
f1 <- fitdist(s1, "exp")
b1 <- bootdist(f1, niter=11) # voluntarily low to decrease computation time

# plot 95 percent bilateral confidence intervals on y values (probabilities)
CICdfplot(b1, CI.level= 95/100, CI.output = "probability")

# plot of the previous intervals as a band
CICdfplot(b1, CI.level= 95/100, CI.output = "probability", 
CI.fill = "pink", CI.col = "red")

# plot of the previous intervals as a band without empirical and fitted dist. functions
CICdfplot(b1, CI.level= 95/100, CI.output = "probability", CI.only = TRUE, 
CI.fill = "pink", CI.col = "red")

# same plot without contours
CICdfplot(b1, CI.level= 95/100, CI.output = "probability", CI.only = TRUE, 
CI.fill = "pink", CI.col = "pink")

# plot 95 percent bilateral confidence intervals on x values (quantiles)
CICdfplot(b1, CI.level= 95/100, CI.output = "quantile")

# plot 95 percent unilateral confidence intervals on quantiles
CICdfplot(b1, CI.level= 95/100, CI.output = "quant", CI.type = "less", 
CI.fill = "grey08", CI.col = "black", CI.lty = 1)
CICdfplot(b1, CI.level= 95/100, CI.output = "quant", CI.type = "greater", 
CI.fill = "grey08", CI.col = "black", CI.lty = 1)

# (2) Fit of a normal distribution on acute toxicity log-transformed values of endosulfan for nonarthropod invertebrates, using maximum likelihood estimation to estimate what is called a species sensitivity distribution
# (SSD) in ecotoxicology, followed by estimation of the 5, 10 and 20 percent quantile values of the fitted distribution, which are called the 5, 10, 20 percent hazardous concentrations (HC5, HC10, HC20) in ecotoxicology, with their confidence intervals, from a small number of bootstrap iterations to satisfy CRAN running times constraint and plot of the band representing pointwise confidence intervals on any quantiles (any HCX values)
# For practical applications, we recommend to use at least niter=50 or niter=100.
#
data(endosulfan)
ATV <- subset(endosulfan, group == "NonArthroInvert")$ATV
log10ATV <- log10(subset(endosulfan, group == "NonArthroInvert")$ATV)
fln <- fitdist(log10ATV, "norm")
bln <- bootdist(fln, bootmethod = "param", niter=10)
quantile(bln, probs = c(0.05, 0.1, 0.2))
CICdfplot(bln, CI.output = "quantile", CI.fill = "lightblue", CI.col = "blue",
          xlim = c(1,5))

# (3) Same type of example as example (2) from ecotoxicology
# with censored data
#
data(salinity)
log10LC50 <- log10(salinity)
fln <- fitdistcens(log10LC50,"norm")
bln <- bootdistcens(fl1n, niter=1)
(HC5ln <- quantile(bln,probs = 0.05))
CICdfplot(bln, CI.output = "quantile", CI.fill = "lightblue", CI.col = "blue",
          xlab = "log10(LC50)",xlim=c(0.5,2),lines01 = TRUE)
# zoom around the HC5
CICdfplot(bln, CI.output = "quantile", CI.fill = "lightblue", CI.col = "blue",
          xlab = "log10(LC50)", lines01 = TRUE, xlim = c(0.8, 1.5), ylim = c(0, 1))
abline(h = 0.05, lty = 2) # line corresponding to a CDF of 5 percent

Danish reinsurance claim dataset

Description

The univariate dataset was collected at Copenhagen Reinsurance and comprise 2167 fire losses over the period 1980 to 1990. They have been adjusted for inflation to reflect 1985 values and are expressed in millions of Danish Krone.

The multivariate data set is the same data as above but the total claim has been divided into a building loss, a loss of contents and a loss of profits.

Usage

data(danishuni)
data(danishmulti)

Format

danishuni contains two columns:
Date The day of claim occurrence.
Loss The total loss amount in millions of Danish Krone (DKK).
danishmulti contains five columns:

Date The day of claim occurrence.
Building The loss amount (mDKK) of the building coverage.
Contents The loss amount (mDKK) of the contents coverage.
Profits The loss amount (mDKK) of the profit coverage.
Total The total loss amount (mDKK).

All columns are numeric except Date columns of class Date.

Source


References


Examples

```r
# (1) load of data
#
data(danishuni)

# (2) plot and description of data
#
plotdist(danishuni$Loss)

# (3) load of data
#
data(danishmulti)

# (4) plot and description of data
#
idx <- sample(1:NROW(danishmulti), 10)
barplot(danishmulti$Building[idx], col = "grey25",
ylim = c(0, max(danishmulti$Total[idx])), main = "Some claims of danish data set")
barplot(danishmulti$Content[idx], add = TRUE, col = "grey50", axes = FALSE)
barplot(danishmulti$Profits[idx], add = TRUE, col = "grey75", axes = FALSE)
legend("topleft", legend = c("Building", "Content", "Profits"),
fill = c("grey25", "grey50", "grey75"))
```
Datasets for the FAQ

Description

Datasets used in the FAQ vignette.

Usage

data(dataFAQlog1)
data(dataFAQscale1)
data(dataFAQscale2)

Format

dataFAQlog1 dataFAQscale1 dataFAQscale2 are vectors of numeric data.

Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

descdist

Description of an empirical distribution for non-censored data

Description

Computes descriptive parameters of an empirical distribution for non-censored data and provides a skewness-kurtosis plot.

Usage

descdist(data, discrete = FALSE, boot = NULL, method = "unbiased",
graph = TRUE, obs.col = "darkblue", obs.pch = 16, boot.col = "orange")

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

Arguments

data A numeric vector.
discrete If TRUE, the distribution is considered as discrete.
boot If not NULL, boot values of skewness and kurtosis are plotted from bootstrap samples of data. boot must be fixed in this case to an integer above 10.
method  "unbiased" for unbiased estimated values of statistics or "sample" for sample values.

graph  If FALSE, the skewness-kurtosis graph is not plotted.

obs.col  Color used for the observed point on the skewness-kurtosis graph.

obs.pch  plotting character used for the observed point on the skewness-kurtosis graph.

boot.col  Color used for bootstrap sample of points on the skewness-kurtosis graph.

x  An object of class "descdist".

...  Further arguments to be passed to generic functions

Details

Minimum, maximum, median, mean, sample sd, and sample (if method="sample") or by default unbiased estimations of skewness and Pearson's kurtosis values are printed (Sokal and Rohlf, 1995). A skewness-kurtosis plot such as the one proposed by Cullen and Frey (1999) is given for the empirical distribution. On this plot, values for common distributions are also displayed as tools to help the choice of distributions to fit to data. For some distributions (normal, uniform, logistic, exponential for example), there is only one possible value for the skewness and the kurtosis (for a normal distribution for example, skewness = 0 and kurtosis = 3), and the distribution is thus represented by a point on the plot. For other distributions, areas of possible values are represented, consisting in lines (gamma and lognormal distributions for example), or larger areas (beta distribution for example). The Weibull distribution is not represented on the graph but it is indicated on the legend that shapes close to lognormal and gamma distributions may be obtained with this distribution.

In order to take into account the uncertainty of the estimated values of kurtosis and skewness from data, the data set may be bootstrapped by fixing the argument boot to an integer above 10. boot values of skewness and kurtosis corresponding to the boot bootstrap samples are then computed and reported in blue color on the skewness-kurtosis plot.

If discrete is TRUE, the represented distributions are the Poisson, negative binomial distributions, and the normal distribution to which previous discrete distributions may converge. If discrete is FALSE, these are uniform, normal, logistic, lognormal, beta and gamma distributions.

Value

descdist returns a list with 7 components,

- min  the minimum value
- max  the maximum value
- median  the median value
- mean  the mean value
- sd  the standard deviation sample or estimated value
- skewness  the skewness sample or estimated value
- kurtosis  the kurtosis sample or estimated value
- method  the method specified in input ("unbiased" for unbiased estimated values of statistics or "sample" for sample values.)
Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References

See Also
plotdist

Examples
# (1) Description of a sample from a normal distribution
# with and without uncertainty on skewness and kurtosis estimated by bootstrap
#
# set.seed(1234)
x1 <- rnorm(100)
descdist(x1)
descdist(x1, boot=500)

# (2) Description of a sample from a beta distribution
# with uncertainty on skewness and kurtosis estimated by bootstrap
# with changing of default colors and plotting character for observed point
#
descdist(rbeta(100, shape1=0.05, shape2=1), boot=500, obs.col="blue", obs.pch = 15, boot.col="yellow")

# (3) Description of a sample from a gamma distribution
# with uncertainty on skewness and kurtosis estimated by bootstrap
# without plotting
#
descdist(rgamma(100, shape=2, rate=1), boot=500, graph=FALSE)

# (3) Description of a sample from a Poisson distribution
# with uncertainty on skewness and kurtosis estimated by bootstrap
#
descdist(rpois(100, lambda=2), discrete=TRUE, boot=500)

# (4) Description of serving size data
# with uncertainty on skewness and kurtosis estimated by bootstrap
#
data(groundbeef)
serving <- groundbeef$serving
descdist(serving, boot=500)
**detectedbound**

---

**Detect bounds for density function**

**Description**

Manual detection of bounds of parameter of a density function.

**Usage**

```r
detectbound(distname, vstart, obs, fix.arg=NULL, echo=FALSE)
```

**Arguments**

- `distname`: A character string "name" naming a distribution for which the corresponding density function \( dname \) must be classically defined.
- `vstart`: A named vector giving the initial values of parameters of the named distribution.
- `obs`: A numeric vector for non censored data.
- `fix.arg`: An optional named vector giving the values of fixed parameters of the named distribution. Default to `NULL`.
- `echo`: A logical to show some traces.

**Details**

This function manually tests the following bounds: -1, 0, and 1.

**Value**

`detectbound` returns a 2-row matrix with the lower bounds in the first row and the upper bounds in the second row.

**Author(s)**

Christophe Dutang and Marie-Laure Delignette-Muller.

**References**


**See Also**

`fitdist`
Examples

```r
# case where the density returns a Not-an-Numeric value.
detectbound("exp", c(rate=3), 1:10)
detectbound("binom", c(size=3, prob=1/2), 1:10)
detectbound("nbinom", c(size=3, prob=1/2), 1:10)
```

---

**endosulfan**

*Species Sensitivity Distribution (SSD) for endosulfan*

---

**Description**

Summary of 48- to 96-hour acute toxicity values (LC50 and EC50 values) for exposure of Australian and Non-Australian taxa to endosulfan.

**Usage**

data(endosulfan)

**Format**

dendosulfan is a data frame with 3 columns, named ATV for Acute Toxicity Value (geometric mean of LC50 or EC50 values in micrograms per liter), Australian (coding for Australian or another origin) and group (arthropods, fish or non-arthropod invertebrates).

**Source**


**Examples**

```r
# (1) load of data
#
data(endosulfan)

# (2) plot and description of data for non Australian fish in decimal logarithm
#
log10ATV <- log10(subset(endosulfan,(Australian == "no") & (group == "Fish")))$ATV
plotdist(log10ATV)
descdist(log10ATV,boot=1000)

# (3) fit of a normal and a logistic distribution to data in log10
# (classical distributions used for SSD)
# and visual comparison of the fits
#```
endosulfan

```r
fln <- fitdist(log10ATV,"norm")
summary(fl)

fll <- fitdist(log10ATV,"logis")
summary(fll)

cdfcomp(list(fl,fl),legendtext=c("normal","logistic"),
xlab="log10ATV")
denscomp(list(fl,fl),legendtext=c("normal","logistic"),
xlab="log10ATV")
qqcomp(list(fl,fl),legendtext=c("normal","logistic"))
ppcomp(list(fl,fl),legendtext=c("normal","logistic"))
gofig(list(fl,fl), fitnames = c("lognormal", "loglogistic"))

# (4) estimation of the 5 percent quantile value of
# logistic fitted distribution (5 percent hazardous concentration : HC5)
# with its two-sided 95 percent confidence interval calculated by
# parametric bootstrap
# with a small number of iterations to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.
# in log10(ATV)
bll <- bootdist(fl1,niter=101)
HC5l <- quantile(bll,nprobs = 0.05)
# in ATV
10^(HC5l$quantiles)
10^(HC5l$quantci)

# (5) estimation of the 5 percent quantile value of
# the fitted logistic distribution (5 percent hazardous concentration : HC5)
# with its one-sided 95 percent confidence interval (type "greater")
# calculated by
# nonparametric bootstrap
# with a small number of iterations to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.
# in log10(ATV)
bllnonpar <- bootdist(fll,niter=101,bootmethod = "nonparam")
HC5lgreater <- quantile(bllnonpar,nprobs = 0.05, CI.type="greater")
# in ATV
10^(HC5lgreater$quantiles)
10^(HC5lgreater$quantCI)

# (6) fit of a logistic distribution
# by minimizing the modified Anderson-Darling AD2L distance
# cf. ?mgdist for definition of this distance
#
flAD2L <- fitdist(log10ATV,"logis",method="mge",gof="AD2L")
summary(fllAD2L)
```
Fit of univariate distributions to non-censored data

Description

Fit of univariate distributions to non-censored data by maximum likelihood (mle), moment matching (mme), quantile matching (qme) or maximizing goodness-of-fit estimation (mge). The latter is also known as minimizing distance estimation. Generic methods are print, plot, summary, quantile, logLik, vcov and coef.

Usage

fitdist(data, distr, method = c("mle", "mme", "qme", "mge"),
         start=NULL, fix.arg=NULL, discrete, keepdata = TRUE, keepdata.nb=100, ...)

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

## S3 method for class 'fitdist'
plot(x, breaks="default", ...)

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

## S3 method for class 'fitdist'
logLik(object, ...)

## S3 method for class 'fitdist'
vcov(object, ...)

## S3 method for class 'fitdist'
coef(object, ...)

Arguments

data A numeric vector.
distr A character string "name" naming a distribution for which the corresponding density function dname, the corresponding distribution function pname and the corresponding quantile function qname must be defined, or directly the density function.
method A character string coding for the fitting method: "mle" for 'maximum likelihood estimation', "mme" for 'moment matching estimation', "qme" for 'quantile matching estimation' and "mge" for 'maximum goodness-of-fit estimation'.

start A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see the 'details' section of mledist). It may not be into account for closed-form formulas.

fix.arg An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated by this maximum likelihood procedure. The use of this argument is not possible if method="mme" and a closed-form formula is used.

keepdata a logical. If TRUE, dataset is returned, otherwise only a sample subset is returned.

keepdata.nb When keepdata=FALSE, the length (>1) of the subset returned.

discrete If TRUE, the distribution is considered as discrete. If discrete is missing, discrete is automatically set to TRUE when distr belongs to "binom", "nbinom", "geom", "hyper" or "pois" and to FALSE in the other cases. It is thus recommended to enter this argument when using another discrete distribution. This argument will not directly affect the results of the fit but will be passed to functions gofstat, plotdist and cdfcomp.

x An object of class "fitdist".

object An object of class "fitdist".

breaks If "default" the histogram is plotted with the function hist with its default breaks definition. Else breaks is passed to the function hist. This argument is not taken into account with discrete distributions: "binom", "nbinom", "geom", "hyper" and "pois".

... Further arguments to be passed to generic functions, or to one of the functions "mledist", "mmedist", "qmedist" or "mgedist" depending of the chosen method. See mledist, mmedist, qmedist, mgedist for details on parameter estimation.

Details

It is assumed that the distr argument specifies the distribution by the probability density function, the cumulative distribution function and the quantile function (d, p, q).

The four possible fitting methods are described below:

When method="mle" Maximum likelihood estimation consists in maximizing the log-likelihood. A numerical optimization is carried out in mledist via optim to find the best values (see mledist for details).

When method="mme" Moment matching estimation consists in equalizing theoretical and empirical moments. Estimated values of the distribution parameters are computed by a closed-form formula for the following distributions: "norm", "lnorm", "pois", "exp", "gamma", "nbinom", "geom", "beta", "unif" and "logis". Otherwise the theoretical and the empirical moments are matched numerically, by minimization of the sum of squared differences
between observed and theoretical moments. In this last case, further arguments are needed in the call to fitdist: order and memp (see mmedist for details).

When method = "qme" Quantile matching estimation consists in equalizing theoretical and empirical quantile. A numerical optimization is carried out in qmedist via optim to minimize of the sum of squared differences between observed and theoretical quantiles. The use of this method requires an additional argument probs, defined as the numeric vector of the probabilities for which the quantile(s) is(are) to be matched (see qmedist for details).

When method = "mge" Maximum goodness-of-fit estimation consists in maximizing a goodness-of-fit statistics. A numerical optimization is carried out in mgedist via optim to minimize the goodness-of-fit distance. The use of this method requires an additional argument gof coding for the goodness-of-fit distance chosen. One can use the classical Cramer-von Mises distance ("CvM"), the classical Kolmogorov-Smirnov distance ("KS"), the classical Anderson-Darling distance ("AD") which gives more weight to the tails of the distribution, or one of the variants of this last distance proposed by Luceno (2006) (see mgedist for more details). This method is not suitable for discrete distributions.

By default, direct optimization of the log-likelihood (or other criteria depending of the chosen method) is performed using optim, with the "Nelder-Mead" method for distributions characterized by more than one parameter and the "BFGS" method for distributions characterized by only one parameter. The optimization algorithm used in optim can be chosen or another optimization function can be specified using ... argument (see mledist for details). start may be omitted (i.e. NULL) for some classic distributions (see the 'details' section of mledist). Note that when errors are raised by optim, it's a good idea to start by adding traces during the optimization process by adding control=list(trace=1, REPORT=1) in ...argument.

Once the parameter(s) is(are) estimated, fitdist computes the log-likelihood for every estimation method and for maximum likelihood estimation the standard errors of the estimates calculated from the Hessian at the solution found by optim or by the user-supplied function passed to mledist.

By default (keepdata = TRUE), the object returned by fitdist contains the data vector given in input. When dealing with large datasets, we can remove the original dataset from the output by setting keepdata = FALSE. In such a case, only keepdata.nb points (at most) are kept by random subsampling keepdata.nb-2 points from the dataset and adding the minimum and the maximum. If combined with bootdist, and use with non-parametric bootstrap be aware that bootstrap is performed on the subset randomly selected in fitdist. Currently, the graphical comparisons of multiple fits is not available in this framework.

Weighted version of the estimation process is available for method = "mle", "mme", "qme" by using weights=.... See the corresponding man page for details. Weighted maximum GOF estimation (when method = "mge" ) is not allowed. It is not yet possible to take into account weighths in functions plotdist, plot.fitdist, cdfcomp, denscomp, ppcomp, qqcomp, gofstat and descdist (developments planned in the future).

NB: if your data values are particularly small or large, a scaling may be needed before the optimization process. See example (14) in this man page and examples (14,15) in the test file of the package. Please also take a look at the Rmpfr package available on CRAN for numerical accuracy issues.

Value

fitdist returns an object of class "fitdist", a list with the following components:

estimate the parameter estimates.
fitdist

method the character string coding for the fitting method: "mle" for 'maximum likelihood estimation', "mme" for 'matching moment estimation', "qme" for 'matching quantile estimation' and "mge" for 'maximum goodness-of-fit estimation'.

sd the estimated standard errors, NA if numerically not computable or NULL if not available.

cor the estimated correlation matrix, NA if numerically not computable or NULL if not available.

vcov the estimated variance-covariance matrix, NULL if not available.

loglik the log-likelihood.

aic the Akaike information criterion.

bic the the so-called BIC or SBC (Schwarz Bayesian criterion).

n the length of the data set.

data the data set.

distname the name of the distribution.

fix.arg the named list giving the values of parameters of the named distribution that must be kept fixed rather than estimated by maximum likelihood or NULL if there are no such parameters.

fix.arg.fun the function used to set the value of fix.arg or NULL.

dots the list of further arguments passed in ... to be used in bootdist in iterative calls to mledist, mmedist, qmedist, mgedist or NULL if no such arguments.

convergence an integer code for the convergence of optim/constrOptim defined as below or defined by the user in the user-supplied optimization function. 0 indicates successful convergence. 1 indicates that the iteration limit of optim has been reached. 10 indicates degeneracy of the Neelder-Mead simplex. 100 indicates that optim encountered an internal error.

discrete the input argument or the automatic definition by the function to be passed to functions gofstat, plotdist and cdfcomp.

weights the vector of weights used in the estimation process or NULL.

Generic functions:

print The print of a "fitdist" object shows few traces about the fitting method and the fitted distribution.

summary The summary provides the parameter estimates of the fitted distribution, the log-likelihood, AIC and BIC statistics and when the maximum likelihood is used, the standard errors of the parameter estimates and the correlation matrix between parameter estimates.

plot The plot of an object of class "fitdist" returned by fitdist uses the function plotdist. An object of class "fitdist" or a list of objects of class "fitdist" corresponding to various fits using the same data set may also be plotted using a cdf plot (function cdfcomp), a density plot (function denscomp), a density Q-Q plot (function qqcomp), or a P-P plot (function ppcomp).

logLik Extracts the estimated log-likelihood from the "fitdist" object.

vcov Extracts the estimated var-covariance matrix from the "fitdist" object (only available When method = "mle").

coef Extracts the fitted coefficients from the "fitdist" object.
Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

See *fitdistrplus* for an overview of the package. See *mledist, mmedist, qmedist, mgedist* for details on parameter estimation. See *gofstat* for goodness-of-fit statistics. See *plotdist, graphcomp, Ciddfplot* for graphs (with or without uncertainty and/or multiple fits). See *llplot* for (log-)likelihood plots in the neighborhood of the fitted value. See *bootdist* for bootstrap procedures and *fitdistcens* for censored-data fitting methods. See *optim* for base R optimization procedures. See *quantile.fitdistr*, another generic function, which calculates quantiles from the fitted distribution. See *quantile* for base R quantile computation.

Examples

```r
# (1) fit of a gamma distribution by maximum likelihood estimation
#

data(groundbeef)
serving <- groundbeef$serving
fitg <- fitdist(serving, "gamma")
summary(fitg)
plot(fitg)
plot(fitg, demp = TRUE)
plot(fitg, histo = FALSE, demp = TRUE)
cdfcomp(fitg, addlegend=FALSE)
denscomp(fitg, addlegend=FALSE)
ppcomp(fitg, addlegend=FALSE)
qqcomp(fitg, addlegend=FALSE)

# (2) use the moment matching estimation (using a closed formula)
#

fitgmmme <- fitdist(serving, "gamma", method="mme")
summary(fitgmmme)

# (3) Comparison of various fits
```
### fitdist

```r
#

fitW <- fitdist(serving, "weibull")
fitg <- fitdist(serving, "gamma")
fitln <- fitdist(serving, "lnorm")
summary(fitW)
summary(fitg)
summary(fitln)
cdfcomp(list(fitW, fitg, fitln), legendtext=c("Weibull", "gamma", "lognormal"))
denscomp(list(fitW, fitg, fitln), legendtext=c("Weibull", "gamma", "lognormal"))
qqcomp(list(fitW, fitg, fitln), legendtext=c("Weibull", "gamma", "lognormal"))
ppcomp(list(fitW, fitg, fitln), legendtext=c("Weibull", "gamma", "lognormal"))
gofstat(list(fitW, fitg, fitln), fitnames=c("Weibull", "gamma", "lognormal"))

# (4) defining your own distribution functions, here for the Gumbel distribution
# for other distributions, see the CRAN task view
# dedicated to probability distributions
#

dgumbel <- function(x, a, b) 1/b*exp((-a*x)/b)*exp(-exp((-a*x)/b))
pgumbel <- function(q, a, b) exp(-exp((-a-q)/b))
qgumbel <- function(p, a, b) a-b*log(-log(p))

fitgumbel <- fitdist(serving, "gumbel", start=list(a=10, b=10))
summary(fitgumbel)
plot(fitgumbel)

# (5) fit discrete distributions (Poisson and negative binomial)
#

data(toxocara)
number <- toxocara$number
fitp <- fitdist(number,"pois")
summary(fitp)
plot(fitp)
fitnb <- fitdist(number,"nbinom")
summary(fitnb)
plot(fitnb)
cdfcomp(list(fitp,fitnb))
gofstat(list(fitp,fitnb))

# (6) how to change the optimisation method?
#

data(groundbeef)
serving <- groundbeef$serving
fitdist(serving, "gamma", optim.method="Nelder-Mead")
fitdist(serving, "gamma", optim.method="BFGS")
fitdist(serving, "gamma", optim.method="SANN")

# (7) custom optimization function
#
```
## Not run:

### create the sample

```r
set.seed(1234)
mysample <- rexp(100, 5)
mystart <- list(rate=8)
```

```r
res1 <- fitdist(mysample, dexp, start= mystart, optim.method="Nelder-Mead")
```

# show the result

```r
summary(res1)
```

# the warning tell us to use optimise, because the Nelder-Mead is not adequate.

# to meet the standard 'fn' argument and specific name arguments, we wrap optimize,

```r
myoptimize <- function(fn, par, ...)
{
  res <- optimize(f=fn, ..., maximum=FALSE)
  # assume the optimization function minimize

  standardres <- c(res, convergence=0, value=res$objective,
                   par=res$minimum, hessian=NA)

  return(standardres)
}
```

# call fitdist with a 'custom' optimization function

```r
res2 <- fitdist(mysample, "exp", start=mystart, custom.optim=myoptimize,
                interval=c(0, 100))
```

# show the result

```r
summary(res2)
```

## End(Not run)

## (8) custom optimization function - another example with the genetic algorithm

## Not run:

### set a sample

```r
fit1 <- fitdist(serving, "gamma")
summary(fit1)
```

# wrap genoud function rgenoud package

```r
mygenoud <- function(fn, par, ...)
{
  require(rgenoud)
  res <- genoud(fn, starting.values=par, ...)
  standardres <- c(res, convergence=0)

  return(standardres)
}
```

# call fitdist with a 'custom' optimization function
fit2 <- fitdist(serving, "gamma", custom.optim=mygenoud, nvars=2,  
  Domains=cbind(c(0, 0), c(10, 10)), boundary.enforcement=1,  
  print.level=1, hessian=TRUE)

summary(fit2)

## End(Not run)

# (9) estimation of the standard deviation of a gamma distribution  
# by maximum likelihood with the shape fixed at 4 using the argument fix.arg  
#  
# data(groundbeef)  
serving <- groundbeef$serving  
fQc <- fitdist(serving, "gamma", start=list(rate=0.1), fix.arg=list(shape=4))  
summary(fQc)  
plot(fQc)

# (10) fit of a Weibull distribution to serving size data  
# by maximum likelihood estimation  
# or by quantile matching estimation (in this example  
# matching first and third quartiles)  
#  
# data(groundbeef)  
serving <- groundbeef$serving  
fWmle <- fitdist(serving, "weibull")  
summary(fWmle)  
plot(fWmle)  
gofstat(fWmle)

fWqme <- fitdist(serving, "weibull", method="qme", probs=c(0.25, 0.75))  
summary(fWqme)  
plot(fWqme)  
gofstat(fWqme)

# (11) Fit of a Pareto distribution by numerical moment matching estimation  
#  
## Not run:  
  require(actuar)  
  #simulate a sample  
x4 <- rpareto(1000, 6, 2)

  #empirical raw moment  
memp <- function(x, order) mean(x^order)

  #fit  
fP <- fitdist(x4, "pareto", method="mme", order=c(1, 2), memp="memp",  
    start=list(shape=10, scale=10), lower=1, upper=Inf)  
summary(fP)  
plot(fP)
## End(Not run)

# (12) Fit of a Weibull distribution to serving size data by maximum
# goodness-of-fit estimation using all the distances available
#
## Not run:
data(groundbeef)
serving <- groundbeef$serving
(f1 <- fitdist(serving, "weibull", method="mge", gof="CvM"))
(f2 <- fitdist(serving, "weibull", method="mge", gof="KS"))
(f3 <- fitdist(serving, "weibull", method="mge", gof="AD"))
(f4 <- fitdist(serving, "weibull", method="mge", gof="ADR"))
(f5 <- fitdist(serving, "weibull", method="mge", gof="ADL"))
(f6 <- fitdist(serving, "weibull", method="mge", gof="AD2R"))
(f7 <- fitdist(serving, "weibull", method="mge", gof="AD2L"))
(f8 <- fitdist(serving, "weibull", method="mge", gof="AD2"))
cdfcomp(list(f1, f2, f3, f4, f5, f6, f7, f8))
cdfcomp(list(f1, f2, f3, f4, f5, f6, f7, f8),
  xlogscale=TRUE, xlim=c(8, 250), verticals=TRUE)
denscomp(list(f1, f2, f3, f4, f5, f6, f7, f8))

## End(Not run)

# (13) Fit of a uniform distribution using maximum likelihood
# (a closed formula is used in this special case where the loglikelihood is not defined),
# or maximum goodness-of-fit with Cramer-von Mises or Kolmogorov-Smirnov distance
#
set.seed(1234)
u <- runif(50, min=5, max=10)
fuMLE <- fitdist(u, "unif", method="mle")
summary(fuMLE)
plot(fuMLE)
gofstat(fuMLE)

fuCvM <- fitdist(u, "unif", method="mge", gof="CvM")
summary(fuCvM)
plot(fuCvM)
gofstat(fuCvM)

fuKS <- fitdist(u, "unif", method="mge", gof="KS")
summary(fuKS)
plot(fuKS)
gofstat(fuKS)

# (14) scaling problem
# the simulated dataset (below) has particularly small values, hence without scaling (10^0),
# the optimization raises an error. The for loop shows how scaling by 10^i
# for i=1,\ldots,6 makes the fitting procedure work correctly.
set.seed(1234)
x2 <- rnorm(100, 1e-4, 2e-4)

for(i in 0:6)
cat(i, try(fitdist(x2*10^i, "cauchy", method="mle")$estimate, silent=TRUE), "\n")

# (15) Fit of a normal distribution on acute toxicity values of endosulfan in log10 for
# nonarthropod invertebrates, using maximum likelihood estimation
# to estimate what is called a species sensitivity distribution
# (SSD) in ecotoxicology, followed by estimation of the 5 percent quantile value of
# the fitted distribution (which is called the 5 percent hazardous concentration, HC5,
# in ecotoxicology) and estimation of other quantiles.
#
data(endosulfan)
ATV <- subset(endosulfan, group == "NonArthroInvert")$ATV
log10ATV <- log10(subset(endosulfan, group == "NonArthroInvert")$ATV)
fln <- fitdist(log10ATV, "norm")

quantile(fln, probs = 0.05)
quantile(fln, probs = c(0.05, 0.1, 0.2))

# (16) Fit of a triangular distribution using Cramer-von Mises or
# Kolmogorov-Smirnov distance
#
## Not run:
set.seed(1234)
require(mc2d)
t <- rtriang(100, min=5, mode=6, max=10)
fCvM <- fitdist(t, "triang", method="mge", start = list(min = 4, mode = 6, max = 9), gof="CvM")
fKS <- fitdist(t, "triang", method="mge", start = list(min = 4, mode = 6, max = 9), gof="KS")
cdfcomp(list(fCvM, fKS))

## End(Not run)

# (17) fit a non classical discrete distribution (the zero inflated Poisson distribution)
#
## Not run:
require(gamlss.dist)
set.seed(1234)
x <- rZIP(n = 30, mu = 5, sigma = 0.2)
plotdist(x, discrete = TRUE)
fitzip <- fitdist(x, "ZIP", start = list(mu = 4, sigma = 0.15), discrete = TRUE,
oxim.method = "L-BFGS-B", lower = c(0, 0), upper = c(Inf, 1))
summary(fitzip)
plot(fitzip)
fitp <- fitdist(x, "pois")
cdfcomp(list(fitzip, fitp))
gofstat(list(fitzip, fitp))

## End(Not run)
# (18) examples with distributions in actuar (predefined starting values)
#
## Not run:
require(actuar)
x <- c(2.3, 0.1, 2.7, 2.2, 0.4, 1.6, 0.2, 1., 2., 3.2, 0.8, 1.2, 3.7, 14.,
      21.4, 7.1, 1., 0.7, 12.6, 3.2, 7.3, 9.4, 0.6, 2.5, 6.7, 3., 63.,
      6., 1.6, 10.1, 1.2, 1.5, 1.2, 30., 3.2, 3.5, 1.2, 0.2, 1.9, 0.7, 17.,
      2.8, 4.8, 1.3, 3.7, 0.2, 1.8, 2.6, 5.9, 2.6, 6.3, 1.4, 0.8)
#log logistic
ft_llogis <- fitdist(x, 'llogis')

x <- c(0.3837053, 0.8576858, 0.3552237, 0.6226119, 0.4783756, 0.3139799, 0.4051403,
       0.4537631, 0.4711057, 0.5647414, 0.6479617, 0.7134207, 0.5259464, 0.5949068,
       0.3509200, 0.3783077, 0.5226465, 1.0241043, 0.4384580, 1.3341520)
#inverse weibull
ft_iw <- fitdist(x, 'invweibull')

## End(Not run)

---

## fitdistcens

### Fitting of univariate distributions to censored data

#### Description
Fits a univariate distribution to censored data by maximum likelihood.

#### Usage

```r
fitdistcens(censdata = NULL, distr = NULL, start = NULL, fix.arg = NULL,
keepdata = TRUE, keepdata.nb = 100, ...)
```

#### Methods

- `print(x, ...)`
- `plot(x, ...)`
- `summary(object, ...)`
- `logLik(object, ...)`
- `vcov(object, ...)`
## Arguments

censdata  A dataframe of two columns respectively named `left` and `right`, describing each observed value as an interval. The `left` column contains either `NA` for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The `right` column contains either `NA` for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.

distr  A character string "name" naming a distribution, for which the corresponding density function `dname` and the corresponding distribution function `pname` must be defined, or directly the density function.

start  A named list giving the initial values of parameters of the named distribution. This argument may be omitted for some distributions for which reasonable starting values are computed (see the 'details' section of `mledist`).

fix.arg  An optional named list giving the values of parameters of the named distribution that must be kept fixed rather than estimated by maximum likelihood.

x  an object of class "fitdistcens".

object  an object of class "fitdistcens".

keepdata  a logical. If TRUE, dataset is returned, otherwise only a sample subset is returned.

keepdata.nb  When `keepdata`=FALSE, the length of the subset returned.

...  further arguments to be passed to generic functions, to the function `plotdistcens` in order to control the type of ecdf-plot used for censored data, or to the function `mledist` in order to control the optimization method.

## Details

Maximum likelihood estimations of the distribution parameters are computed using the function `mledist`. By default direct optimization of the log-likelihood is performed using `optim`, with the "Nelder-Mead" method for distributions characterized by more than one parameter and the "BFGS" method for distributions characterized by only one parameter. The algorithm used in `optim` can be chosen or another optimization function can be specified using ...argument (see `mledist` for details). `start` may be omitted (i.e. `NULL`) for some classic distributions (see the 'details' section of `mledist`). Note that when errors are raised by `optim`, it's a good idea to start by adding traces during the optimization process by adding `control=list(trace=1, REPORT=1)` in ...argument.

The function is not able to fit a uniform distribution. With the parameter estimates, the function returns the log-likelihood and the standard errors of the estimates calculated from the Hessian at the solution found by `optim` or by the user-supplied function passed to `mledist`.

By default (keepdata = TRUE), the object returned by `fitdist` contains the data vector given in input. When dealing with large datasets, we can remove the original dataset from the output by setting keepdata = FALSE. In such a case, only keepdata.nb points (at most) are kept by random
subsampling keepdata. nb-4 points from the dataset and adding the component-wise minimum and maximum. If combined with \texttt{bootdistcens}, be aware that bootstrap is performed on the subset randomly selected in \texttt{fitdistcens}. Currently, the graphical comparisons of multiple fits is not available in this framework.

Weighted version of the estimation process is available for \texttt{method = "mle"} by using weights=... See the corresponding man page for details. It is not yet possible to take into account weights in functions \texttt{plotdistcens}, \texttt{plot.fitdistcens} and \texttt{cdfcompcens} (developments planned in the future).

**Value**

\texttt{fitdistcens} returns an object of class "\texttt{fitdistcens}". a list with the following components:

- \texttt{estimate} the parameter estimates.
- \texttt{method} the character string coding for the fitting method: only "\texttt{mle}" for 'maximum likelihood estimation'.
- \texttt{sd} the estimated standard errors.
- \texttt{cor} the estimated correlation matrix, \texttt{NA} if numerically not computable or \texttt{NULL} if not available.
- \texttt{vcov} the estimated variance-covariance matrix, \texttt{NULL} if not available.
- \texttt{loglik} the log-likelihood.
- \texttt{aic} the Akaike information criterion.
- \texttt{bic} the the so-called BIC or SBC (Schwarz Bayesian criterion).
- \texttt{censdata} the censored data set.
- \texttt{distname} the name of the distribution.
- \texttt{fix.arg} the named list giving the values of parameters of the named distribution that must be kept fixed rather than estimated by maximum likelihood or \texttt{NULL} if there are no such parameters.
- \texttt{fix.arg.fun} the function used to set the value of \texttt{fix.arg} or \texttt{NULL}.
- \texttt{dots} the list of further arguments passed in... to be used in \texttt{bootdistcens} to control the optimization method used in iterative calls to \texttt{mledist} or \texttt{NULL} if no such arguments.
- \texttt{convergence} an integer code for the convergence of \texttt{optim/constrOptim} defined as below or defined by the user in the user-supplied optimization function. \texttt{0} indicates successful convergence. \texttt{1} indicates that the iteration limit of \texttt{optim} has been reached. \texttt{10} indicates degeneracy of the Nealder-Mead simplex. \texttt{100} indicates that \texttt{optim} encountered an internal error.
- \texttt{discrete} always \texttt{FALSE}.
- \texttt{weights} the vector of weights used in the estimation process or \texttt{NULL}.

Generic functions:

\texttt{print} The print of a "\texttt{fitdist}" object shows few traces about the fitting method and the fitted distribution.
**fitdistscens**

summary The summary provides the parameter estimates of the fitted distribution, the log-likelihood, AIC and BIC statistics, the standard errors of the parameter estimates and the correlation matrix between parameter estimates.

plot The plot of an object of class "fitdistscens" returned by fitdistscens uses the function `plotdistscens`.

logLik Extracts the estimated log-likelihood from the "fitdistscens" object.

vcov Extracts the estimated var-covariance matrix from the "fitdistscens" object (only available When method = "mle").

coeff Extracts the fitted coefficients from the "fitdistscens" object.

**Author(s)**

Marie-Laure Delignette-Muller and Christophe Dutang.

**References**


**See Also**

See *fitdistrplus* for an overview of the package. plotdistscens, optim, mledist, fitdist and quantile.fdistscens for another generic function to calculate quantiles from the fitted distribution.

**Examples**

```r
# (1) Fit of a lognormal distribution to bacterial contamination data
#
data(smokedfish)
fitsf <- fitdistscens(smokedfish, "lnorm")
summary(fitsf)
# default plot using the Wang technique (see ?plotdiscens for details)
plot(fitsf)
# plot using the Turnbull algorithm (see ?plotdiscens for details)
# with confidence intervals for the empirical distribution
plot(fitsf, NPMLE = TRUE, NPMLE.method = "Turnbull", Turnbull.confint = TRUE)
# basic plot using intervals and points (see ?plotdiscens for details)
plot(fitsf, NPMLE = FALSE)
# plot of the same fit using the Turnbull algorithm in logscale
cdfcompdistscens(fitsf, main="bacterial contamination fits", xlab="bacterial concentration (CFU/g)", ylab="F", addlegend = FALSE, lines01 = TRUE, xlogscale = TRUE, xlim = c(1e-2,1e2))
```

```r
# zoom on large values of F
cdfcompdistscens(fitsf, main="bacterial contamination fits", xlab="bacterial concentration (CFU/g)", ylab="F", addlegend = FALSE, lines01 = TRUE, xlogscale = TRUE, xlim = c(1e-2,1e2), ylim=c(0.4,1))
```
# (2) Fit of a normal distribution on acute toxicity values
# of fluazinam (in decimal logarithm) for
# macroinvertebrates and zooplankton, using maximum likelihood estimation
# to estimate what is called a species sensitivity distribution
# (SSD) in ecotoxicology
#
data(fluazinam)
log10EC50 <- log10(fluazinam)
fln <- fitdistcens(log10EC50, "norm")
fln
summary(fln)
plot(fln)

# (3) defining your own distribution functions, here for the Gumbel distribution
# for other distributions, see the CRAN task view dedicated to
# probability distributions
#
dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
qgumbel <- function(p,a,b) a+b*log(-log(p))
fg <- fitdistcens(log10EC50, "gumbel", start=list(a=1,b=1))
summary(fg)
plot(fg)

# (4) comparison of fits of various distributions
#
fll <- fitdistcens(log10EC50, "logis")
summary(fll)
cdfcompcens(list(fl, fl1, fg), legendtext=c("normal", "logistic", "gumbel"),
xlab = "log10(EC50)")

# (5) how to change the optimisation method?
#
fitdistcens(log10EC50, "logis", optim.method="Nelder-Mead")
fitdistcens(log10EC50, "logis", optim.method="BFGS")
fitdistcens(log10EC50, "logis", optim.method="SANN")

# (6) custom optimisation function - example with the genetic algorithm
#
## Not run:

#wrap genoud function rgenoud package
mygenoud <- function(fn, par, ...)
{
  require(rgenoud)
  res <- genoud(fn, starting.values=par, ...)
  standardres <- c(res, convergence=0)
}
return(standardres)
}

# call fitdistcens with a 'custom' optimization function
fit.with.genoud <- fitdistcens(log10EC50, "logis", custom.optim=mygenoud, nvars=2,
  Domains=cbind(c(0, 0), c(5, 5)), boundary.enforcement=1,
  print.level=1, hessian=TRUE)

summary(fit.with.genoud)

## End(Not run)

# (7) estimation of the mean of a normal distribution
# by maximum likelihood with the standard deviation fixed at 1 using the argument fix.arg
# flnb <- fitdistcens(log10EC50, "norm", start = list(mean = 1), fix.arg = list(sd = 1))

# (8) Fit of a lognormal distribution on acute toxicity values of fluazinam for
# macroinvertebrates and zooplancton, using maximum likelihood estimation
# to estimate what is called a species sensitivity distribution
# (SSD) in ecotoxicology, followed by estimation of the 5 percent quantile value of
# the fitted distribution (which is called the 5 percent hazardous concentration, HC5,
# in ecotoxicology) and estimation of other quantiles.

data(fluazinam)
log10EC50 <- log10(fluazinam)
fln <- fitdistcens(log10EC50, "norm")

quantile(fln, probs = 0.05)
quantile(fln, probs = c(0.05, 0.1, 0.2))

# (9) Fit of a lognormal distribution on 72-hour acute salinity tolerance (LC50 values)
# of riverine macro-invertebrates using maximum likelihood estimation

data(salinity)
log10LC50 <- log10(salinity)
fln <- fitdistcens(log10LC50, "norm")
plot(fln)

---

**fluazinam**

*Species-Sensitivity Distribution (SSD) for Fluazinam*

**Description**

48-hour acute toxicity values (EC50 values) for exposure of macroinvertebrates and zooplancton to fluazinam.
Usage

data(fluazinam)

Format

fluazinam is a data frame with 2 columns named left and right, describing each observed EC50 value (in micrograms per liter) as an interval. The left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.

Source


Examples

```r
# (1) load of data
#
data(fluazinam)

# (2) plot of data using Turnbull cdf plot
#
log10EC50 <- log10(fluazinam)
plotdistcens(log10EC50)

# (3) fit of a lognormal and a logistic distribution to data
# (classical distributions used for species sensitivity distributions, SSD, in ecotoxicology)
# and visual comparison of the fits using Turnbull cdf plot
#
fln <- fitdistcens(log10EC50,"norm")
summary(fln)

fll <- fitdistcens(log10EC50,"logis")
summary(fll)

cdfcompcens(list(fln,fll),legendtext=c("normal","logistic"),
  xlab = "log10(EC50)"

# (4) estimation of the 5 percent quantile value of
# the normal fitted distribution (5 percent hazardous concentration : HC5)
# with its two-sided 95 percent confidence interval calculated by
# non parametric bootstrap
# with a small number of iterations to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.
#
# in log10(EC50)
```
gofstat <- bootdistcens(fln, niter=101)
HC5ln <- quantile(bln, probs = 0.05)
# in EC50
10^(HC5ln$quantiles)
10^(HC5ln$quantCI)

# (5) estimation of the HC5 value
# with its one-sided 95 percent confidence interval (type "greater")
# in log10(EC50)
HC5lnb <- quantile(bln,probs = 0.05,CI.type="greater")

# in LC50
10^(HC5lnb$quantiles)
10^(HC5lnb$quantCI)

---

**gofstat**

**Goodness-of-fit statistics**

**Description**

Computes goodness-of-fit statistics for parametric distributions fitted to a same non-censored data set.

**Usage**

gofstat(f, chisqbreaks, meancount, discrete, fitnames=NULL)

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

**Arguments**

- **f**
  - An object of class "fitdist", output of the function fitdist, or a list of "fitdist" objects.

- **chisqbreaks**
  - A numeric vector defining the breaks of the cells used to compute the chi-squared statistic. If omitted, these breaks are automatically computed from the data in order to reach roughly the same number of observations per cell, roughly equal to the argument meancount, or slightly more if there are some ties.

- **meancount**
  - The mean number of observations per cell expected for the definition of the breaks of the cells used to compute the chi-squared statistic. This argument will not be taken into account if the breaks are directly defined in the argument chisqbreaks. If chisqbreaks and meancount are both omitted, meancount is fixed in order to obtain roughly \((4n)^{2/5}\) cells with \(n\) the length of the dataset.
discrete If TRUE, only the Chi-squared statistic and information criteria are computed. If missing, discrete is passed from the first object of class "fitdist" of the list f.

fitnames A vector defining the names of the fits.

x An object of class "fitdist".

... Further arguments to be passed to generic functions.

Details

Goodness-of-fit statistics are computed. The Chi-squared statistic is computed using cells defined by the argument chisqbreaks or cells automatically defined from data, in order to reach roughly the same number of observations per cell, roughly equal to the argument meancount, or slightly more if there are some ties. The choice to define cells from the empirical distribution (data), and not from the theoretical distribution, was done to enable the comparison of Chi-squared values obtained with different distributions fitted on a same data set. If chisqbreaks and meancount are both omitted, meancount is fixed in order to obtain roughly \( \left( \frac{4n}{n} \right)^{2/5} \) cells, with \( n \) the length of the data set (Vose, 2000). The Chi-squared statistic is not computed if the program fails to define enough cells due to a too small dataset. When the Chi-squared statistic is computed, and if the degree of freedom (nb of cells - nb of parameters - 1) of the corresponding distribution is strictly positive, the p-value of the Chi-squared test is returned.

For continuous distributions, Kolmogorov-Smirnov, Cramer-von Mises and Anderson-Darling and statistics are also computed, as defined by Stephens (1986).

An approximate Kolmogorov-Smirnov test is performed by assuming the distribution parameters known. The critical value defined by Stephens (1986) for a completely specified distribution is used to reject or not the distribution at the significance level 0.05. Because of this approximation, the result of the test (decision of rejection of the distribution or not) is returned only for data sets with more than 30 observations. Note that this approximate test may be too conservative.

For data sets with more than 5 observations and for distributions for which the test is described by Stephens (1986) for maximum likelihood estimations ("exp", "cauchy", "gamma" and "weibull"), the Cramer-von Mises and Anderson-darling tests are performed as described by Stephens (1986). Those tests take into account the fact that the parameters are not known but estimated from the data by maximum likelihood. The result is the decision to reject or not the distribution at the significance level 0.05. Those tests are available only for maximum likelihood estimations.

Only recommended statistics are automatically printed, i.e. Cramer-von Mises, Anderson-Darling and Kolmogorov statistics for continuous distributions and Chi-squared statistics for discrete ones ("binom", "nbinom", "geom", "hyper" and "pois").

Results of the tests are not printed but stored in the output of the function.

Value
gof.stat returns an object of class "gofstat.fitdist" with following components,

chisq a named vector with the Chi-squared statistics or NULL if not computed

chisqbreaks common breaks used to define cells in the Chi-squared statistic

chisqpvalue a named vector with the p-values of the Chi-squared statistic or NULL if not computed
chisqdf  a named vector with the degrees of freedom of the Chi-squared distribution or
        NULL if not computed
chisqtable  a table with observed and theoretical counts used for the Chi-squared calcula-
            tions

cvm  a named vector of the Cramer-von Mises statistics or "not computed" if not computed

cvmtest  a named vector of the decisions of the Cramer-von Mises test or "not computed"
        if not computed

ad  a named vector with the Anderson-Darling statistics or "not computed" if not computed

adtest  a named vector with the decisions of the Anderson-Darling test or "not computed"
        if not computed

ks  a named vector with the Kolmogorov-Smirnov statistic or "not computed" if not computed

kstest  a named vector with the decisions of the Kolmogorov-Smirnov test or "not computed"
        if not computed

aic  a named vector with the values of the Akaike’s Information Criterion.

bic  a named vector with the values of the Bayesian Information Criterion.

discrete  the input argument or the automatic definition by the function from the first
            object of class "fitdist" of the list in input.

nbfit  Number of fits in argument.

Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References

Cullen AC and Frey HC (1999), Probabilistic techniques in exposure assessment. Plenum Press,
        USA, pp. 81-155.

        and Stephens MA, eds), Marcel Dekker, New York, pp. 97-194.

        435-446.

Vose D (2000), Risk analysis, a quantitative guide. John Wiley & Sons Ltd, Chischester, England,
        pp. 99-143.

        Journal of Statistical Software, 64(4), 1-34.

See Also

fitdist.
Examples

# (1) fit of two distributions to the serving size data
# by maximum likelihood estimation
# and comparison of goodness-of-fit statistics
#

data(groundbeef)
serving <- groundbeef$serving
(fitg <- fitdist(serving, "gamma"))
gofstat(fitg)
(fitln <- fitdist(serving, "lnorm"))
gofstat(fitln)

gofstat(list(fitg, fitln))

# (2) fit of two discrete distributions to toxocara data
# and comparison of goodness-of-fit statistics
#

data(toxocara)
number <- toxocara$number

fitp <- fitdist(number,"pois")
summary(fitp)
plot(fitp)

fitnb <- fitdist(number,"nbinom")
summary(fitnb)
plot(fitnb)

gofstat(list(fitp, fitnb), fitnames = c("Poisson","negbin"))

# (3) Use of Chi-squared results in addition to
# recommended statistics for continuous distributions
#
set.seed(1234)
x4 <- rweibull(n=1000,shape=2,scale=1)
# fit of the good distribution
f4 <- fitdist(x4,"weibull")

# fit of a bad distribution
f4b <- fitdist(x4,"cauchy")

gofstat(list(f4,f4b), fitnames=c("Weibull", "Cauchy"))
**graphcomp**

Graphical comparison of multiple fitted distributions (for non-censored data)

**Description**

`cdfcomp` plots the empirical cumulative distribution against fitted distribution functions, `denscomp` plots the histogram against fitted density functions, `qqcomp` plots theoretical quantiles against empirical ones, `ppcomp` plots theoretical probabilities against empirical ones. Only `cdfcomp` is able to plot fits of a discrete distribution.

**Usage**

```r
cdfcomp(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab, datapch, datacol, fitlty, fitcol, addlegend = TRUE, legendtext, xlegend = "bottomright", ylegend = NULL, horizontal = TRUE, verticals = FALSE, do.points = TRUE, use.ppoints = TRUE, a.points = 0.5, lines01 = FALSE, discrete, add = FALSE, plotstyle = "graphics", fitnbpts = 101, ...)
```

```r
denscomp(ft, xlim, ylim, probability = TRUE, main, xlab, ylab, datacol, fitlty, fitcol, addlegend = TRUE, legendtext, xlegend = "topright", ylegend = NULL, demp = FALSE, dempcol = "black", plotstyle = "graphics", discrete, fitnbpts = 101, fittype="l", ...)
```

```r
qqcomp(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab, fitpch, fitcol, addlegend = TRUE, legendtext, xlegend = "bottomright", ylegend = NULL, use.ppoints = TRUE, a.points = 0.5, line01 = TRUE, line01col = "black", line01lty = 1, ynoise = TRUE, plotstyle = "graphics", ...)
```

```r
ppcomp(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab, fitpch, fitcol, addlegend = TRUE, legendtext, xlegend = "bottomright", ylegend = NULL, use.ppoints = TRUE, a.points = 0.5, line01 = TRUE, line01col = "black", line01lty = 1, ynoise = TRUE, plotstyle = "graphics", ...)
```

**Arguments**

- `ft` One "fitdist" object or a list of objects of class "fitdist".
- `xlim` The $x$-limits of the plot.
- `ylim` The $y$-limits of the plot.
- `xlogscale` If TRUE, uses a logarithmic scale for the $x$-axis.
- `ylogscale` If TRUE, uses a logarithmic scale for the $y$-axis.
- `main` A main title for the plot. See also `title`.
- `xlab` A label for the $x$-axis, defaults to a description of $x$.
- `ylab` A label for the $y$-axis, defaults to a description of $y$.
- `datapch` An integer specifying a symbol to be used in plotting data points. See also `par`.
graphcomp

**datacol**
A specification of the color to be used in plotting data points. See also `par`.

**fitcol**
A (vector of) color(s) to plot fitted distributions. If there are fewer colors than fits they are recycled in the standard fashion. See also `par`.

**fitlty**
A (vector of) line type(s) to plot fitted distributions/densities. If there are fewer colors than fits they are recycled in the standard fashion. See also `par`.

**fitpch**
A (vector of) line type(s) to plot fitted quantiles/probabilities. If there are fewer colors than fits they are recycled in the standard fashion. See also `par`.

**fittype**
The type of plot for fitted probabilities in the case of discrete distributions: possible types are "p" for points, "l" for lines and "o" for both overplotted (as in `plot.default`). `fittype` is not used for non-discrete distributions.

**fitnbpts**
A numeric for the number of points to compute fitted probabilities or cumulative probabilities. Default to 101.

**addlegend**
If `true`, a legend is added to the plot.

**legendtext**
A character or expression vector of length $\geq 1$ to appear in the legend. See also `legend`.

**xlegend, ylegend**
The $x$ and $y$ coordinates to be used to position the legend. They can be specified by keyword. If `plotstyle = "graphics"`, see `xy.coords` and `legend`. If `plotstyle = "ggplot"`, the `xlegend` keyword must be one of top, bottom, left, or right. See also `guide_legend` in ggplot2

**horizontals**
If `TRUE`, draws horizontal lines for the step empirical cumulative distribution function (ecdf). See also `plot.stepfun`.

**do.points**
If `TRUE` (by default), draws points at the x-locations. For large dataset ($n > 1\times 10^{4}$), `do.points` is ignored and no point is drawn.

**verticals**
If `TRUE`, draws vertical lines for the empirical cumulative distribution function (ecdf). Only taken into account if `horizontals=TRUE`.

**use.ppoints**
If `TRUE`, probability points of the empirical distribution are defined using function `ppoints` as $(1:n - a.ppoints)/(n - 2a.ppoints + 1)$. If FALSE, probability points are simply defined as $1:n/n$. This argument is ignored for discrete data.

**a.ppoints**
If `use.ppoints=TRUE`, this is passed to the `ppoints` function.

**lines01**
A logical to plot two horizontal lines at $h=0$ and $h=1$ for `cdfcomp`.

**line01**
A logical to plot an horizontal line $y = x$ for `qqcomp` and `ppcomp`.

**line01col, line01lty**
Color and line type for `line01`. See also `par`.

**demp**
A logical to add the empirical density on the plot, using the `density` function.

**dempcol**
A color for the empirical density in case it is added on the plot (`demp=TRUE`).

**ynoise**
A logical to add a small noise when plotting empirical quantiles/probabilities for `qqcomp` and `ppcomp`.

**probability**
A logical to use the probability scale for `denscomp`. See also `hist`.

**discrete**
If TRUE, the distributions are considered discrete. When missing, `discrete` is set to `TRUE` if at least one object of the list `ft` is discrete.
add If TRUE, adds to an already existing plot. If FALSE, starts a new plot. This parameter is not available when plotstyle = "ggplot".

plotstyle "graphics" or "ggplot". If "graphics", the display is built with graphics functions. If "ggplot", a graphic object output is created with ggplot2 functions.

... Further graphical arguments passed to graphical functions used in cdfcomp, denscomp, pcomp and qcomp when plotstyle = "graphics". When plotstyle = "ggplot", these arguments are only used by the histogram plot (hist) in the denscomp function. When plotstyle = "ggplot", the graphical output can be customized with relevant ggplot2 functions after you store your output.

Details
cdfcomp provides a plot of the empirical distribution and each fitted distribution in cdf, by default using the Hazen's rule for the empirical distribution, with probability points defined as \((1:n - 0.5)/n\). If discrete is TRUE, probability points are always defined as \((1:n)/n\). For large dataset \((n > 1e4)\), no point is drawn but the line for ecdf is drawn instead. Note that when horizontals, verticals and do.points are FALSE, no empirical point is drawn, only the fitted cdf is shown.

denscomp provides a density plot of each fitted distribution with the histogram of the data for continuous data. When discrete=TRUE, distributions are considered as discrete, no histogram is plotted but demp is forced to TRUE and fitted and empirical probabilities are plotted either with vertical lines fittype="l", with single points fittype="p" or both lines and points fittype="o".

ppcomp provides a plot of the probabilities of each fitted distribution (x-axis) against the empirical probabilities (y-axis) by default defined as \((1:n - 0.5)/n\) (data are assumed continuous). For large dataset \((n > 1e4)\), lines are drawn instead of pointss and customized with the fitpch parameter.

qqcomp provides a plot of the quantiles of each theoretical distribution (x-axis) against the empirical quantiles of the data (y-axis), by default defining probability points as \((1:n - 0.5)/n\) for theoretical quantile calculation (data are assumed continuous). For large dataset \((n > 1e4)\), lines are drawn instead of points and customized with the fitpch parameter.

By default a legend is added to these plots. Many graphical arguments are optional, dedicated to personalize the plots, and fixed to default values if omitted.

Author(s)
Christophe Dutang, Marie-Laure Delignette-Muller and Aurelie Siberchicot.

References

See Also
See plot, legend, ppoints, plot.stepfun for classic plotting functions. See CIdcfplot and plotdist for other plot functions of fitdistrplus.
Examples

# (1) Plot various distributions fitted to serving size data
#
data(groundbeef)
serving <- groundbeef$serving
fitW <- fitdist(serving, "weibull")
fitln <- fitdist(serving, "lnorm")
fitg <- fitdist(serving, "gamma")

cdfcomp(list(fitW, fitln, fitg), horizontals = FALSE)
cdfcomp(list(fitW, fitln, fitg), horizontals = TRUE)
cdfcomp(list(fitW, fitln, fitg), horizontals = TRUE, verticals = TRUE, datacol = "purple")
cdfcomp(list(fitW, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
    main = "ground beef fits", xlab = "serving sizes (g)",
    ylab = "F", xlim = c(0, 250), xlegend = "center", lines01 = TRUE)
denscomp(list(fitW, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
    main = "ground beef fits", xlab = "serving sizes (g)",
    xlim = c(0, 250), xlegend = "topright")
denscomp(list(fitW, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
    main = "ground beef fits", xlegend = "bottomright", line01 = TRUE)
ppcomp(list(fitW, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
    main = "ground beef fits", xlegend = "bottomright", line01 = TRUE,
    xlim = c(0, 300), ylim = c(0, 300), fitpch = 16)

# (2) Plot lognormal distributions fitted by
# maximum goodness-of-fit estimation
# using various distances (data plotted in log scale)
#
data(endosulfan)
ATV <- subset(endosulfan, group == "NonArthroInvert")$ATV
flnMGEKS <- fitdist(ATV, "lnorm", method = "mge", gof = "KS")
flnMGEAD <- fitdist(ATV, "lnorm", method = "mge", gof = "AD")
flnMGEADL <- fitdist(ATV, "lnorm", method = "mge", gof = "ADL")
flnMGEAD2L <- fitdist(ATV, "lnorm", method = "mge", gof = "AD2L")

cdfcomp(list(flnMGEKS, flnMGEAD, flnMGEADL, flnMGEAD2L),
    xlogscale = TRUE, main = "Fits of a lognormal dist. using various GOF dist.",
    legendtext = c("MGE KS", "MGE AD", "MGE ADL", "MGE AD2L"),
    verticals = TRUE, xlim = c(10, 100000))
qqcomp(list(flnMGEKS, flnMGEAD, flnMGEADL, flnMGEAD2L),
    main = "Fits of a lognormal dist. using various GOF dist.",
    legendtext = c("MGE KS", "MGE AD", "MGE ADL", "MGE AD2L"),
    xlogscale = TRUE, ylogscale = TRUE)
ppcomp(list(flnMGEKS, flnMGEAD, flnMGEADL, flnMGEAD2L),
    main = "Fits of a lognormal dist. using various GOF dist.",
    legendtext = c("MGE KS", "MGE AD", "MGE ADL", "MGE AD2L"))

# (3) Plot normal and logistic distributions fitted by
# maximum likelihood estimation
# using various plotting positions in cdf plots
#
data(endsosulfan)
log10ATV <- log10(subset(endsosulfan, group == "NonArthroInvert")$ATV)
fln <- fitdist(log10ATV, "norm")
fl1 <- fitdist(log10ATV, "logis")

# default plot using Hazen plotting position: (1:n - 0.5)/n
cdfcomp(list(fln, fl1), legendtext = c("normal", "logistic"), xlab = "log10ATV")

# plot using mean plotting position (named also Gumbel plotting position)
# (1:n)/(n + 1)
cdfcomp(list(fln, fl1), legendtext = c("normal", "logistic"), xlab = "log10ATV",
use.ppoints = TRUE, a.ppoints = 0)

# plot using basic plotting position: (1:n)/n
cdfcomp(list(fln, fl1), legendtext = c("normal", "logistic"), xlab = "log10ATV",
use.ppoints = FALSE)

# (4) Comparison of fits of two distributions fitted to discrete data
#
data(toxocara)
number <- toxocara$number
fitp <- fitdist(number, "pois")
fitnb <- fitdist(number, "nbinom")
cdfcomp(list(fitp, fitnb), legendtext = c("Poisson", "negative binomial"))
denscomp(list(fitp, fitnb), demp = TRUE, legendtext = c("Poisson", "negative binomial"))
denscomp(list(fitp, fitnb), demp = TRUE, fitype = "l", dempcol = "black",
legendtext = c("Poisson", "negative binomial"))
denscomp(list(fitp, fitnb), demp = TRUE, fitype = "p", dempcol = "black",
legendtext = c("Poisson", "negative binomial"))
denscomp(list(fitp, fitnb), demp = TRUE, fitype = "o", dempcol = "black",
legendtext = c("Poisson", "negative binomial"))

# (5) Customizing of graphical output and use of ggplot2
#
data(groundbeef)
serving <- groundbeef$serving
fitW <- fitdist(serving, "weibull")
fitln <- fitdist(serving, "lnorm")
fitg <- fitdist(serving, "gamma")
if (requireNamespace("ggplot2", quietly = TRUE)) {
denscomp(list(fitW, fitln, fitg), plotstyle = "ggplot")
cdfcomp(list(fitW, fitln, fitg), plotstyle = "ggplot")
qqcomp(list(fitW, fitln, fitg), plotstyle = "ggplot")
ppcomp(list(fitW, fitln, fitg), plotstyle = "ggplot")
}

# customizing graphical output with graphics
denscomp(list(fitW, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
main = "ground beef fits", xlab = "serving sizes (g)", xlim = c(0, 250),
xlegend = "topright", addlegend = FALSE)
# customizing graphical output with ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
  dcomp <- denscomp(list(fitw, fitln, fitg), legendtext = c("Weibull", "lognormal", "gamma"),
    xlab = "serving sizes (g)", xlim = c(0, 250),
    xlegend = "topright", plotstyle = "ggplot", breaks = 20, addlegend = FALSE)
  dcomp + ggplot2::theme_minimal() + ggplot2::ggtitle("Ground beef fits")
}

---

**graphcompcens**

*Graphical comparison of multiple fitted distributions for censored data*

## Description

cdfcompcens plots the empirical cumulative distribution against fitted distribution functions, qqcompcens plots theoretical quantiles against empirical ones, ppcompcens plots theoretical probabilities against empirical ones.

## Usage

cdfcompcens(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab,
  datacol, fillrect, fitlty, fitcol, addlegend = TRUE, legendtext,
  xlegend = "bottomright", ylegend = NULL, lines01 = FALSE,
  Turnbull.confint = FALSE, NPMLE.method = "Wang", add = FALSE,
  plotstyle = "graphics", ...)

qqcompcens(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab,
  fillrect, fitcol, addlegend = TRUE, legendtext, xlegend = "bottomright",
  ylegend = NULL, line01 = TRUE, line01col = "black", line01lty = 1,
  ynoise = TRUE, plotstyle = "graphics", ...)

ppcompcens(ft, xlim, ylim, xlogscale = FALSE, ylogscale = FALSE, main, xlab, ylab,
  fillrect, fitcol, addlegend = TRUE, legendtext, xlegend = "bottomright",
  ylegend = NULL, line01 = TRUE, line01col = "black", line01lty = 1,
  ynoise = TRUE, plotstyle = "graphics", ...)

## Arguments

- **ft**
  One "fitdistcens" object or a list of objects of class "fitdistcens".
- **xlim**
  The x-limits of the plot.
- **ylim**
  The y-limits of the plot.
- **xlogscale**
  If TRUE, uses a logarithmic scale for the x-axis.
- **ylogscale**
  If TRUE, uses a logarithmic scale for the y-axis.
- **main**
  A main title for the plot, see also title.
- **xlab**
  A label for the x-axis, defaults to a description of x.
- **ylab**
  A label for the y-axis, defaults to a description of y.
datacol  A specification of the color to be used in plotting data points.

fillrect  A specification of the color to be used for filling rectangles of non uniqueness of the empirical cumulative distribution (only used if NPMLE.method is equal to "Wang" in cdfcompcens). Fix it to NA if you do not want to fill the rectangles.

fitcol  A (vector of) color(s) to plot fitted distributions. If there are fewer colors than fits they are recycled in the standard fashion.

fitlty  A (vector of) line type(s) to plot fitted distributions. If there are fewer colors than fits they are recycled in the standard fashion. See also par.

addlegend  If TRUE, a legend is added to the plot.

legendtext  A character or expression vector of length ≥ 1 to appear in the legend, see also legend.

xlegend, ylegend  The x and y coordinates to be used to position the legend. They can be specified by keyword. If plotstyle = "graphics", see xy.coords and legend. If plotstyle = "ggplot", the xlegend keyword must be one of top, bottom, left, or right. See also guide_legend in ggplot2

linesP0  A logical to plot two horizontal lines at h=P and h=Q for cdfcompcens.

Turnbull.confint  if TRUE confidence intervals will be added to the Turnbull plot. In that case NPMLE.method is forced to "Turnbull"

NPMLE.method  Two NPMLE techniques are provided, "Wang", the default one, which is implemented in the package npsurv and "Turnbull", an older one, which is implemented in the package survival (see details). Only "Wang" enables the derivation of a Q-Q plot and a P-P plot.

add  If TRUE, adds to an already existing plot. If FALSE, starts a new plot. This parameter is not available when plotstyle = "ggplot".

lineP1  A logical to plot an horizontal line y = x for qqcompcens and ppcompcens.

lineP1col, lineP1lty  Color and line type for lineP1. See also par.

ynoise  A logical to add a small noise when plotting empirical quantiles/probabilities for qqcompcens and ppcompcens. ynoise is only used when various fits are plotted with the "graphics" plotstyle. Facets are used instead with the "ggplot" plotstyle.

plotstyle  "graphics" or "ggplot". If "graphics", the display is built with graphics functions. If "ggplot", a graphic object output is created with ggplot2 functions. In "cdfcompcens", "ggplot" graphics are only available with "Wang" NPMLE technique.

...  Further graphical arguments passed to graphical functions used in cdfcompcens, ppcompcens and qqcompcens.

Details

See details of plotdistcens for a detailed description of provided goodness-of-fit plots.
Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

plotdistcens, survfit.formula, legend and par.

Examples

```r
# (1) Plot various distributions fitted to bacterial contamination data
#
data(smokedfish)
Clog10 <- log10(smokedfish)

fitsfn <- fitdistcens(Clog10,"norm")
summary(fitsfn)

fitsfl <- fitdistcens(Clog10,"logis")
summary(fitsfl)

dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
qgumbel <- function(p,a,b) a-b*log(-log(p))
fitsfg<-fitdistcens(Clog10,"gumbel",start=list(a=-3,b=3))
summary(fitsfg)

# CDF plot
cdfcompcens(list(fitsfn,fitsfl,fitsfg))
cdfcompcens(list(fitsfn,fitsfl,fitsfg),datacol="orange",fillrect = NA,
legendtext=c("normal","logistic","Gumbel"),
main="bacterial contamination fits",
 xlab="bacterial concentration (CFU/g)",ylab="F",
xlegend = "bottom",lines=1 = TRUE)
```

# alternative Turnbull plot for the empirical cumulative distribution
# (default plot of the previous versions of the package)
cdfcompcens(list(fitsfn,fitsfl,fitsfg), NPMLE.method = "Turnbull")

# customizing graphical output with ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
```
groundbeef

Ground beef serving size data set

description

Serving sizes collected in a French survey, for ground beef patties consumed by children under 5 years old.

usage

data(groundbeef)
Format

groundbeef is a data frame with 1 column (serving: serving sizes in grams)

Source


Examples

```r
# (1) load of data
#
data(groundbeef)

# (2) description and plot of data
#
serving <- groundbeef$serving
descdist(serving)
plotdist(serving)

# (3) fit of a Weibull distribution to data
#
fitW <- fitdist(serving,"weibull")
summary(fitW)
plot(fitW)
gofstat(fitW)
```

---

**logLikplot**

(Log)likelihood plot for a fit using maximum likelihood

Description

logLikplot plots the (log)likelihood around the estimation for distributions fitted by maximum likelihood.

Usage

```r
logLikplot(mlefit, loglik = TRUE, expansion = 1, lseq = 50,
back.col = TRUE, nlev = 10, pal.col = terrain.colors(100),
fit.show = FALSE, fit.pch = 4, ...)
```

Arguments

- **mlefit**
  An object of class "fitdist" of "fitdistcens" obtained by maximum likelihood (with method = "mle")

- **loglik**
  a logical to plot log-likelihood or likelihood function.
expansion
   a expansion factor to enlarge the default range of values explored for each parameter.
1seq
   length of sequences of parameters.
back.col
   logical (for llsurface only). Contours are plotted with a background gradient of colors if TRUE.
nlev
   number of contour levels to plot.
pal.col
   Palette of colors. Colors to be used as back (for llsurface only).
fit.show
   a logical to plot the mle estimate.
fit.pch
   the type of point used to plot the mle estimate.
... Further graphical arguments passed to graphical functions.

Details

llplot plots the (log)likelihood surface(s) (or curve if there there is only one estimated parameter) around the maximum likelihood estimation. It internally calls function llsurface and llcurve. When there is more than two estimated parameters, the (log)likelihood surface is plotted for each combination of two parameters, fixing the other ones to their estimated value. For each (log)likelihood surface, when back.col image (2D-plot) is used and when nlev > 0 contour (2D-plot) is used to add nlev contours. By default the range of values explored for each estimated parameter is of 2 standard error around the mle estimate but this range can be expanded (or contracted) using the argument expansion.

Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

See llsurface and llcurve for manual (log)likelihood plots (surface ou curve) and plot, contour, image for classic plotting functions.

Examples

# (1) a distribution with one parameter
#

x <- rexp(50)
fite <- fitdist(x, "exp")
llplot(fite)
llplot(fite, col = "red", fit.show = TRUE)
llplot(fite, col = "red", fit.show = TRUE, loglik = FALSE)
# (2) a distribution with two parameters
#
data(groundbeef)
serving <- groundbeef$serving
fitg <- fitdist(serving, "gamma")
llplot(fitg)
## Not run:
llplot(fitg, expansion = 2)
llplot(fitg, pal.col = heat.colors(100), fit.show = TRUE)
llplot(fitg, back.col = FALSE, nlev = 25, fit.show = TRUE)
## End(Not run)

# (3) a distribution with two parameters with one fixed
#
fitg2 <- fitdist(serving, "gamma", fix.arg = list(rate = 0.5))
llplot(fitg2, fit.show = TRUE)

# (4) a distribution with three parameters
#
## Not run:
data(endosulfan)
ATV <- endosulfan$ATV
library("actuar")
fBurr <- fitdist(ATV, "burr", start = list(shape1 = 0.3, shape2 = 1, rate = 1))
llplot(fBurr)
llplot(fBurr, back.col = FALSE, fit.show = TRUE, fit.pch = 16)
llplot(fBurr, nlev = 0, pal.col = rainbow(100), lseq = 100)
## End(Not run)

# (5) a distribution with two parameters fitted on censored data
#
data(salinity)
fsal <- fitdistcens(salinity, "lnorm")
llplot(fsal, fit.show = TRUE)
llplot(fsal, fit.show = TRUE, loglik = FALSE)

---

logLksurface (Log)likelihood surfaces or (log)likelihood curves

Description

llsurface plots the likelihood surface for distributions with two or more parameters, llcurve plots the likelihood curve for distributions with one or more parameters.
Usage

llsurface(data, distr, plot.arg, min.arg, max.arg, lseq = 50, fix.arg = NULL, loglik = TRUE, back.col = TRUE, nlev = 10, pal.col = terrain.colors(100), weights = NULL, ...)

llcurve(data, distr, plot.arg, min.arg, max.arg, lseq = 50, fix.arg = NULL, loglik = TRUE, weights = NULL, ...)

Arguments

data A numeric vector for non-censored data or a dataframe of two columns respectively named left and right, describing each observed value as an interval for censored data. In that case the left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.

distr A character string "name" naming a distribution for which the corresponding density function dname and the corresponding distribution function pname must be classically defined.

plot.arg a two-element vector with the names of the two parameters that will vary for llsurface, only one element for llcurve.

min.arg a two-element vector with lower plotting bounds for llsurface, only one element for llcurve.

max.arg a two-element vector with upper plotting bounds for llsurface, only one element for llcurve.

lseq length of sequences of parameters.

fix.arg a named list with fixed value of other parameters.

loglik a logical to plot log-likelihood or likelihood function.

back.col logical (for llsurface only). Contours are plotted with a background gradient of colors if TRUE.

nlev number of contour levels to plot (for llsurface only).

pal.col Palette of colors. Colors to be used as back (for llsurface only).

weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector with strictly positive values (classically the number of occurrences of each observation).

... Further graphical arguments passed to graphical functions.

Details

These two function are not intended to be called directly but is internally called in llplot.
llsurface plots the likelihood surface for distributions with two varying parameters and other parameters fixed. When back.col, image (2D-plot) is used. When nlev > 0, contour (2D-plot) is used to add nlev contours.

llcurve plots the likelihood curve for distributions with one varying parameter and other parameters fixed.

Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References

See Also
See llplot for an automatic (log)likelihood plots (surface ou curve) of an object of class "fitdist" or "fitdistcens" and plot, contour, image for classic plotting functions.

Examples

```r
# (1) loglikelihood or likelihood curve
#
set.seed(1234)
x <- rexp(n)

llcurve(data = x, distr = "exp", plot.arg = "rate", min.arg = 0, max.arg = 4)

llcurve(data = x, distr = "exp", plot.arg = "rate", min.arg = 0, max.arg = 4, loglik = FALSE)

llcurve(data = x, distr = "exp", plot.arg = "rate", min.arg = 0, max.arg = 4, main = "log-likelihood for exponential distribution", col = "red")
abline(v = 1, lty = 2)

# (2) loglikelihood surface
#
x <- rnorm(n, 0, 1)

llsurface(data =x, distr="norm", plot.arg=c("mean", "sd"),
          min.arg=c(-1, 0.5), max.arg=c(1, 3/2), back.col = FALSE,
          main="log-likelihood for normal distribution")

llsurface(data =x, distr="norm", plot.arg=c("mean", "sd"),
          min.arg=c(-1, 0.5), max.arg=c(1, 3/2),
          main="log-likelihood for normal distribution",
          nlev = 20, pal.col = heat.colors(100),)
points(0, 1, pch="+", col="red")
```
mgedist

Maximum goodness-of-fit fit of univariate continuous distributions

Description

Fit of univariate continuous distribution by maximizing goodness-of-fit (or minimizing distance) for non censored data.

Usage

mgedist(data, distr, gof = "CvM", start = NULL, fix.arg = NULL, optim.method = "default", lower = -Inf, upper = Inf, custom.optim = NULL, silent = TRUE, gradient = NULL, checkstartfix=FALSE, ...)

Arguments

data A numeric vector for non censored data.
distr A character string "name" naming a distribution for which the corresponding quantile function qname and the corresponding density function dname must be classically defined.
start A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see the 'details' section of mledist).
fix.arg An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated.
optim.method "default" or optimization method to pass to optim.
lower Left bounds on the parameters for the "L-BFGS-B" method (see optim).
upper Right bounds on the parameters for the "L-BFGS-B" method (see optim).
custom.optim a function carrying the optimization.
silent A logical to remove or show warnings when bootstrapping.
mgedist

gradient A function to return the gradient of the gof distance for the "BFGS", "CG" and "L-BFGS-B" methods. If it is NULL, a finite-difference approximation will be used.

checkstartfix A logical to test starting and fixed values. Do not change it.

... further arguments passed to the optim, constrOptim or custom.optim function.

Details

The mgedist function numerically maximizes goodness-of-fit, or minimizes a goodness-of-fit distance coded by the argument gof. One may use one of the classical distances defined in Stephens (1986), the Cramer-von Mises distance ("CvM"), the Kolmogorov-Smirnov distance ("KS") or the Anderson-Darling distance ("AD") which gives more weight to the tails of the distribution, or one of the variants of this last distance proposed by Luceno (2006). The right-tail AD ("ADR") gives more weight only to the right tail, the left-tail AD ("ADL") gives more weight only to the left tail. Either of the tails, or both of them, can receive even larger weights by using second order Anderson-Darling Statistics (using "AD2R", "AD2L" or "AD2").

The optimization process is the same as mledist, see the 'details' section of that function.

This function is not intended to be called directly but is internally called in fitdist and bootdist. This function is intended to be used only with continuous distributions and weighted maximum goodness-of-fit estimation is not allowed.

NB: if your data values are particularly small or large, a scaling may be needed before the optimization process. See example (4).

Value

mgedist returns a list with following components,

estimate the parameter estimates.

convergence an integer code for the convergence of optim defined as below or defined by the user in the user-supplied optimization function. 0 indicates successful convergence. 1 indicates that the iteration limit of optim has been reached. 10 indicates degeneracy of the Neelder-Mead simplex. 100 indicates that optim encountered an internal error.

value the minimal value reached for the criterion to minimize.

hessian a symmetric matrix computed by optim as an estimate of the Hessian at the solution found or computed in the user-supplied optimization function.

optim.function the name of the optimization function used for maximum likelihood.

optim.method when optim is used, the name of the algorithm used, the field method of the custom.optim function otherwise.

fix.arg the named list giving the values of parameters of the named distribution that must kept fixed rather than estimated by maximum likelihood or NULL if there are no such parameters.

fix.arg.fun the function used to set the value of fix.arg or NULL.

weights the vector of weights used in the estimation process or NULL.
counts A two-element integer vector giving the number of calls to the log-likelihood function and its gradient respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to log-likelihood function to compute a finite-difference approximation to the gradient. counts is returned by optim or the user-supplied function or set to NULL.

optim.message A character string giving any additional information returned by the optimizer, or NULL. To understand exactly the message, see the source code.

loglik the log-likelihood value.

Gof the code of the goodness-of-fit distance maximized.

Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also
mmedist, mledist, qmedist, fitdist for other estimation methods.

Examples

# (1) Fit of a Weibull distribution to serving size data by maximum
# goodness-of-fit estimation using all the distances available
#

data(groundbeef)
serving <- groundbeef$serving
mgedist(serving, "weibull", gof="CvM")
mgedist(serving, "weibull", gof="KS")
mgedist(serving, "weibull", gof="AD")
mgedist(serving, "weibull", gof="ADR")
mgedist(serving, "weibull", gof="ADL")
mgedist(serving, "weibull", gof="AD2R")
mgedist(serving, "weibull", gof="AD2L")
mgedist(serving, "weibull", gof="AD2")

# (2) Fit of a uniform distribution using Cramer-von Mises or
# Kolmogorov-Smirnov distance
#

set.seed(1234)
u <- runif(100, min=5, max=10)
mgedist(u, "unif", gof="CvM")
mgedist(u, "unif", gof="KS")

# (3) Fit of a triangular distribution using Cramer-von Mises or
# Kolmogorov-Smirnov distance
#

## Not run:
require(mcd2d)
set.seed(1234)
t <- rtriang(100, min=5, mode=6, max=10)
mgedist(t, "triang", start = list(min=4, mode=6, max=9), gof="CvM")
mgedist(t, "triang", start = list(min=4, mode=6, max=9), gof="KS")

## End(Not run)

# (4) scaling problem
# the simulated dataset (below) has particularly small values, hence without scaling (10^0),
# the optimization raises an error. The for loop shows how scaling by 10^i
# for i=1,...,6 makes the fitting procedure work correctly.

set.seed(1234)
x2 <- rnorm(100, 1e-4, 2e-4)
for(i in 6:0)
  cat(i, try(mgedist(x*10^i,"cauchy")$estimate, silent=TRUE), "\n")

---

mledist

Maximum likelihood fit of univariate distributions

Description

Fit of univariate distributions using maximum likelihood for censored or non censored data.

Usage

mledist(data, distr, start = NULL, fix.arg = NULL, optim.method = "default",
lower = -Inf, upper = Inf, custom.optim = NULL, weights = NULL, silent = TRUE,
gradient = NULL, checkstartfix=FALSE, ...)

Arguments

data

A numeric vector for non censored data or a dataframe of two columns respectively named left and right, describing each observed value as an interval for censored data. In that case the left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for
interval censored observations, or the observed value for non-censored observations.

distr
A character string "name" naming a distribution for which the corresponding density function dname and the corresponding distribution function pname must be classically defined.

start
A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see details).

fix.arg
An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated by this maximum likelihood procedure.

optim.method
"default" (see details) or an optimization method to pass to optim.

lower
Left bounds on the parameters for the "L-BFGS-B" method (see optim).

upper
Right bounds on the parameters for the "L-BFGS-B" method (see optim).

custom.optim
A function carrying the MLE optimisation (see details).

weights
an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector with strictly positive integers (typically the number of occurrences of each observation). If non-NULL, weighted MLE is used, otherwise ordinary MLE.

silent
A logical to remove or show warnings when bootstrapping.

gradient
A function to return the gradient of the log-likelihood for the "BFGS", "CG" and "L-BFGS-B" methods. If it is NULL, a finite-difference approximation will be used, see details.

checkstartfix
A logical to test starting and fixed values. Do not change it.

... further arguments passed to the optim, constrOptim or custom.optim function.

Details
This function is not intended to be called directly but is internally called in fitdist and bootdist when used with the maximum likelihood method and fitdistcens and bootdistcens.

It is assumed that the distr argument specifies the distribution by the probability density function and the cumulative distribution function (d, p). The quantile function and the random generator function (q, r) may be needed by other function such as mmedist, qmedist, mgedist, fitdist, fitdistcens, bootdistcens and bootdist.

For the following named distributions, reasonable starting values will be computed if start is omitted (i.e. NULL): "norm", "lnorm", "exp" and "pois", "cauchy", "gamma", "logis", "nbinom" (parametrized by mu and size), "geom", "beta", "weibull" from the stats package; "invgamma", "llogis", "invweibull", "paretol", "pareto", "lgamma", "trgamma", "invtrgamma" from the actuar package. Note that these starting values may not be good enough if the fit is poor. The function uses a closed-form formula to fit the uniform distribution. If start is a list, then it should be a named list with the same names as in the d,p,q,r functions of the chosen distribution. If start
is a function of data, then the function should return a named list with the same names as in the d,p,q,r functions of the chosen distribution.

The mledist function allows user to set a fixed values for some parameters. As for start, if fix.arg is a list, then it should be a named list with the same names as in the d,p,q,r functions of the chosen distribution. If fix.arg is a function of data, then the function should return a named list with the same names as in the d,p,q,r functions of the chosen distribution.

When custom.optim=NULL (the default), maximum likelihood estimations of the distribution parameters are computed with the R base optim or constrOptim. If no finite bounds (lower=Inf and upper=Inf) are supplied, optim is used with the method specified by optim.method. Note that optim.method="default" means optim.method="Nelder-Mead" for distributions with at least two parameters and optim.method="BFGS" for distributions with only one parameter. If finite bounds are supplied (among lower and upper) and gradient != NULL, constrOptim is used. If finite bounds are supplied (among lower and upper) and gradient == NULL, constrOptim is used when optim.method="Nelder-Mead"; optim is used when optim.method="L-BFGS-B" or "Brent"; in other case, an error is raised (same behavior as constrOptim).

When errors are raised by optim, it’s a good idea to start by adding traces during the optimization process by adding control=list(trace=1, REPORT=1).

If custom.optim is not NULL, then the user-supplied function is used instead of the R base optim. The custom.optim must have (at least) the following arguments fn for the function to be optimized, par for the initialized parameters. Internally the function to be optimized will also have other arguments, such as obs with observations and ddistname with distribution name for non censored data (Beware of potential conflicts with optional arguments of custom.optim). It is assumed that custom.optim should carry out a MINIMIZATION. Finally, it should return at least the following components par for the estimate, convergence for the convergence code, value for fn(par), hessian, counts for the number of calls (function and gradient) and message (default to NULL) for the error message when custom.optim raises an error, see the returned value of optim. See examples in fitdist and fitdistcens.

Optionally, a vector of weights can be used in the fitting process. By default (when weights=NULL), ordinary MLE is carried out, otherwise the specified weights are used to balance the log-likelihood contributions. It is not yet possible to take into account weights in functions plotdist, plotdistcens, plot.fitdist, plot.fitdistcens, cdfcomp, cdfcompcri, denscomp, ppcomp, qqcomp, gofstat, descdis, bootdist, bootdistcens and mgedist. (developments planned in the future).

NB: if your data values are particularly small or large, a scaling may be needed before the optimization process. See Example (7).

Value

mledist returns a list with following components,

- **estimate** the parameter estimates,
- **convergence** an integer code for the convergence of optim/constrOptim defined as below or defined by the user in the user-supplied optimization function. 0 indicates successful convergence. 1 indicates that the iteration limit of optim has been reached. 10 indicates degeneracy of the Neelder-Mead simplex. 100 indicates that optim encountered an internal error.
- **value** the minimal value reached for the criterion to minimize.
hessian a symmetric matrix computed by optim as an estimate of the Hessian at the solution found or computed in the user-supplied optimization function. It is used in fitdist to estimate standard errors.

optim.function the name of the optimization function used for maximum likelihood.

optim.method when optim is used, the name of the algorithm used, the field method of the custom.optim function otherwise.

fix.arg the named list giving the values of parameters of the named distribution that must kept fixed rather than estimated by maximum likelihood or NULL if there are no such parameters.

fix.arg.fun the function used to set the value of fix.arg or NULL.

weights the vector of weights used in the estimation process or NULL.

counts A two-element integer vector giving the number of calls to the log-likelihood function and its gradient respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to log-likelihood function to compute a finite-difference approximation to the gradient. counts is returned by optim or the user-supplied function or set to NULL.

optim.message A character string giving any additional information returned by the optimizer, or NULL. To understand exactly the message, see the source code.

loglik the log-likelihood value.

Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

mmedist, qmedist, mgedist, fitdist, fitdistcens for other estimation methods, optim, constrOptim for optimization routines, bootdistcens and bootdist for bootstrap, and llplot for plotting the (log)likelihood.

Examples

# (1) basic fit of a normal distribution with maximum likelihood estimation
#
set.seed(1234)
x1 <- rnorm(n=100)
ml edist(x1,"norm")
# (2) defining your own distribution functions, here for the Gumbel distribution
# for other distributions, see the CRAN task view dedicated to probability distributions

dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
mledist(x1,"gumbel",start=list(a=10,b=5))

# (3) fit of a discrete distribution (Poisson)
#
set.seed(1234)
x2 <- rpois(n=30,lambda = 2)
mledist(x2,"pois")

# (4) fit a finite-support distribution (beta)
#
set.seed(1234)
x3 <- rbeta(n=100,shape1=5, shape2=10)
mledist(x3,"beta")

# (5) fit frequency distributions on USArests dataset.
#
x4 <- USArests$Assault
mledist(x4, "pois")
mledist(x4, "nbinom")

# (6) fit a continuous distribution (Gumbel) to censored data.
#
data(fluazinam)
log10EC50 <- log10(fluazinam)
# definition of the Gumbel distribution
dgumbel <- function(x,a,b) 1/b*exp((a-x)/b)*exp(-exp((a-x)/b))
pgumbel <- function(q,a,b) exp(-exp((a-q)/b))
qgumbel <- function(p,a,b) a-b*log(-log(p))
mledist(log10EC50,"gumbel",start=list(a=0,b=2),optim.method="Nelder-Mead")

# (7) scaling problem
# the simulated dataset (below) has particularly small values,
# hence without scaling (10^0),
# the optimization raises an error. The for loop shows how scaling by 10^i
# for i=1,...,6 makes the fitting procedure work correctly.
set.seed(1234)
x2 <- rnorm(100, 1e-4, 2e-4)
for(i in 0:6)
  cat(i, try(mledist(x*10^i, "cauchy")$estimate, silent=TRUE), "\n")

# (17) small example for the zero-modified geometric distribution
mmedist

Matching moment fit of univariate distributions

Description

Fit of univariate distributions by matching moments (raw or centered) for non censored data.

Usage

mmedist(data, distr, order, memp, start = NULL, fix.arg = NULL, optim.method = "default", lower = -Inf, upper = Inf, custom.optim = NULL, weights = NULL, silent = TRUE, gradient = NULL, checkstartfix = FALSE, ...)

Arguments

data
  A numeric vector for non censored data.
distr
  A character string "name" naming a distribution (see 'details').
order
  A numeric vector for the moment order(s). The length of this vector must be equal to the number of parameters to estimate.
memp
  A function implementing empirical moments, raw or centered but has to be consistent with distr argument (and weights argument). See details below.
start
  A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see the 'details' section of mledist).
fix.arg
  An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated.
optim.method
  "default" or optimization method to pass to optim.
lower
  Left bounds on the parameters for the "L-BFGS-B" method (see optim).
upper
  Right bounds on the parameters for the "L-BFGS-B" method (see optim).
custom.optim
  A function carrying the optimization.
weights
  An optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector with strictly positive integers (typically the number of occurrences of each observation). If non-NULL, weighted MME is used, otherwise ordinary MME.
silent A logical to remove or show warnings when bootstraping.

gradient A function to return the gradient of the squared difference for the "BFGS", "CG" and "L-BFGS-B" methods. If it is NULL, a finite-difference approximation will be used, see details.

checkstartfix A logical to test starting and fixed values. Do not change it.

... further arguments passed to the optim, constrOptim or custom.optim function.

Details

The argument distr can be one of the base R distributions: "norm", "lnorm", "exp" and "pois", "gamma", "logis", "nbinom", "geom", "beta" and "unif". In that case, no other arguments than data and distr are required, because the estimate is computed by a closed-form formula. For distributions characterized by one parameter ("geom", "pois" and "exp"), this parameter is simply estimated by matching theoretical and observed means, and for distributions characterized by two parameters, these parameters are estimated by matching theoretical and observed means and variances (Vose, 2000). Note that for these closed-form formula, fix.arg cannot be used and start is ignored.

The argument distr can also be the distribution name as long as a corresponding mdistr function exists, e.g. "pareto" if "mpareto" exists. In that case arguments arguments order and memp have to be supplied in order to carry out the matching numerically, by minimization of the sum of squared differences between observed and theoretical moments. Optionnally other arguments can be supplied to control optimization (see the 'details' section of mledist for details about arguments for the control of optimization). In that case, fix.arg can be used and start is taken into account.

For non closed-form estimators, memp must be provided to compute empirical moments. When weights=NULL, this function must have two arguments x, order: x the numeric vector of the data and order the order of the moment. When weights!=NULL, this function must have three arguments x, order, weights: x the numeric vector of the data, order the order of the moment, weights the numeric vector of weights. See examples below.

Optionnally, a vector of weights can be used in the fitting process. By default (when weights=NULL), ordinary MME is carried out, otherwise the specified weights are used to compute (raw or centered) weighted moments. For closed-form estimators, weighted mean and variance are computed by wtd.mean and wtd.var from the hmisc package. When a numerical minimization is used, weighted are expected to be computed by the memp function. It is not yet possible to take into account weighths in functions plotdist, plotdistcens, plot.fitdist, plot.fitdistcens, cdfcomp, cdfcompcens, denscomp, ppcomp, qqcomp, gofstat and descdist (developments planned in the future).

This function is not intended to be called directly but is internally called in fitdist and bootdist when used with the matching moments method.

Value

mmedist returns a list with following components,

estimate the parameter estimates.
convergence an integer code for the convergence of `optim` defined as below or defined by the user in the user-supplied optimization function. 0 indicates successful convergence. 1 indicates that the iteration limit of `optim` has been reached. 10 indicates degeneracy of the Nelder-Mead simplex. 100 indicates that `optim` encountered an internal error.

value the minimal value reached for the criterion to minimize.

hessian a symmetric matrix computed by `optim` as an estimate of the Hessian at the solution found or computed in the user-supplied optimization function.

optim.function (if appropriate) the name of the optimization function used for maximum likelihood.

optim.method (if appropriate) when `optim` is used, the name of the algorithm used, the field `method` of the custom `optim` function otherwise.

fix.arg the named list giving the values of parameters of the named distribution that must kept fixed rather than estimated by maximum likelihood or NULL if there are no such parameters.

fix.arg.fun the function used to set the value of `fix.arg` or NULL.

weights the vector of weights used in the estimation process or NULL.

counts A two-element integer vector giving the number of calls to the log-likelihood function and its gradient respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to log-likelihood function to compute a finite-difference approximation to the gradient. `counts` is returned by `optim` or the user-supplied function or set to NULL.

optim.message A character string giving any additional information returned by the optimizer, or NULL. To understand exactly the message, see the source code.

loglik the log-likelihood value.

method either "closed formula" or the name of the optimization method.

order the order of the moment(s) matched.

memp the empirical moment function.

Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References

See Also
mmedist, qmedist, mgedist, fitdist, fitdistcens, optim, bootdistcens and bootdist.
Examples

```r
# (1) basic fit of a normal distribution with moment matching estimation
#
set.seed(1234)
N <- 100
x1 <- rnorm(n=N)
mmedist(x1, "norm")

#weighted
w <- c(rep(1, n/2), rep(10, n/2))
mmedist(x1, "norm", weights=w)$estimate

# (2) fit a discrete distribution (Poisson)
#
set.seed(1234)
x2 <- rpois(n=30, lambda = 2)
mmedist(x2, "pois")

# (3) fit a finite-support distribution (beta)
#
set.seed(1234)
x3 <- rbeta(n=100, shape1=5, shape2=10)
mmedist(x3, "beta")

# (4) fit a Pareto distribution
#
## Not run:
require(actuar)
#simulate a sample
X <- rpareto(1000, 6, 2)

#empirical raw moment
memp <- function(x, order) mean(x^order)
memp2 <- function(x, order, weights) sum(x^order * weights)/sum(weights)

#fit by MME
mmedist(x4, "pareto", order=c(1, 2), memp=memp,
      start=list(shape=10, scale=10, lower=1, upper=Inf))
#fit by weighted MME
w <- rep(1, length(x4))
w[x4 < 1] <- 2
mmedist(x4, "pareto", order=c(1, 2), memp=memp2, weights=w,
      start=list(shape=10, scale=10, lower=1, upper=Inf))

## End(Not run)
```
Plot of empirical and theoretical distributions for non-censored data

Description

Plots an empirical distribution (non-censored data) with a theoretical one if specified.

Usage

```
plotdist(data, distr, para, histo = TRUE, breaks = "default",
demp = FALSE, discrete, ...)
```

Arguments

- `data`: A numeric vector.
- `distr`: A character string naming a distribution for which the corresponding density function `dname`, the corresponding distribution function `pname` and the corresponding quantile function `qname` must be defined, or directly the density function. This argument may be omitted only if `para` is omitted.
- `para`: A named list giving the parameters of the named distribution. This argument may be omitted only if `distr` is omitted.
- `histo`: A logical to plot the histogram using the `hist` function.
- `breaks`: If "default" the histogram is plotted with the function `hist` with its default breaks definition. Else `breaks` is passed to the function `hist`. This argument is not taken into account if `discrete` is `true`.
- `demp`: A logical to plot the empirical density on the first plot (alone or superimposed on the histogram depending of the value of the argument `histo`) using the `density` function.
- `discrete`: If `TRUE`, the distribution is considered as discrete. If both `distr` and `discrete` are missing, `discrete` is set to `FALSE`. If `discrete` is missing but not `distr`, `discrete` is set to `TRUE` when `distr` belongs to "binom", "nbinom", "geom", "hyper" or "pois".
- `...`: Further graphical arguments passed to graphical functions used in `plotdist`.

Details

Empirical and, if specified, theoretical distributions are plotted in density and in cdf. For the plot in density, the user can use the arguments `histo` and `demp` to specify if he wants the histogram using the function `hist`, the density plot using the function `density`, or both (at least one of the two arguments must be put to "TRUE"). For continuous distributions, the function `hist` is used with its default breaks definition if `breaks` is "default" or passing `breaks` as an argument if it differs from "default". For continuous distribution and when a theoretical distribution is specified by both arguments `distrname` and `para`, Q-Q plot (plot of the quantiles of the theoretical fitted distribution
(x-axis) against the empirical quantiles of the data) and P-P plot (i.e., for each value of the data set, plot of the cumulative density function of the fitted distribution (x-axis) against the empirical cumulative density function (y-axis)) are also given (Cullen and Frey, 1999). The function `ppoints` (with default parameter for argument `a`) is used for the Q-Q plot, to generate the set of probabilities at which to evaluate the inverse distribution. NOTE THAT FROM VERSION 0.4-3, `ppoints` is also used for P-P plot and cdf plot for continuous data. To personalize the four plots proposed for continuous data, for example to change the plotting position, we recommend the use of functions `cdfcomp`, `denscomp`, `qqcomp` and `ppcomp`.

**Author(s)**

Marie-Laure Delignette-Muller and Christophe Dutang.

**References**


**See Also**

`graphcomp`, `descdist`, `hist`, `plot`, `plotdistcens` and `ppoints`.

**Examples**

```r
# (1) Plot of an empirical distribution with changing
# of default line types for CDF and colors
# and optionally adding a density line
#
set.seed(1234)
x1 <- rnorm(n=30)
plotdist(x1)
plotdist(x1,demp = TRUE)
plotdist(x1,histo = FALSE, Demp = TRUE)
plotdist(x1, col="blue", type="b", pch=16)
plotdist(x1, type="s")

# (2) Plot of a discrete distribution against data
#
set.seed(1234)
x2 <- rpois(n=30, lambda = 2)
plotdist(x2, discrete=TRUE)
plotdist(x2, "pois", para=list(lambda = mean(x2)))
plotdist(x2, "pois", para=list(lambda = mean(x2)), lwd="2")

# (3) Plot of a continuous distribution against data
#
xn <- rnorm(n=100, mean=10, sd=5)
plotdist(xn, "norm", para=list(mean=mean(xn), sd=sd(xn)))
plotdist(xn, "norm", para=list(mean=mean(xn), sd=sd(xn)), pch=16)
```
Description

Plots an empirical distribution for censored data with a theoretical one if specified.

Usage

plotdistcens(censdata, distr, para, leftNA = -Inf, rightNA = Inf, NPMLE = TRUE, Turnbull = NULL, Turnbull.confint = FALSE, NPMLE.method = "Wang", ...)

Arguments

censdata | A dataframe of two columns respectively named left and right, describing each observed value as an interval. The left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.

distr | A character string "name" naming a distribution, for which the corresponding density function dname and the corresponding distribution function pname must be defined, or directly the density function.

para | A named list giving the parameters of the named distribution. This argument may be omitted only if distr is omitted.

leftNA | the real value of the left bound of left censored observations : -Inf or a finite value such as 0 for positive data for example.

rightNA | the real value of the right bound of right censored observations : Inf or a finite value such as a realistic maximum value.
If TRUE an NPMLE (nonparametric maximum likelihood estimate) technique is used to estimate the cdf curve of the censored data and previous arguments leftNA and rightNA are not used (see details).

Turnbull is a deprecated argument. You should now use npmle to choose to use an NPMLE technique (or not) and npmleNmethod to choose the method.

Turnbull.confint

If TRUE confidence intervals will be added to the Turnbull plot. In that case NPMLE.method is forced to "Turnbull".

NPMLE.method

Two NPMLE techniques are provided, "Wang", the default one, which is implemented in the package npsurv and "Turnbull", an older one, which is implemented in the package survival (see details). Only "Wang" enables the derivation of a Q-Q plot and a P-P plot.

... further graphical arguments passed to other methods. The title of the plot can be modified using the argument main only for the CDF plot.

Details

If NPMLE is TRUE, and NPMLE.method is "Wang", empirical and, if specified, theoretical distributions are plotted in cdf using either the constrained Newton method (Wang, 2008) or the hierarchical constrained Newton method (Wang, 2013) to compute the overall empirical cdf curve, by calls to functions npsurv from the npsurv package. In this plot, grey rectangles represent areas where the empirical distribution function is not unique. In cases where a theoretical distribution is specified, two goodness-of-fit plots are also provided, a Q-Q plot (plot of the quantiles of the theoretical fitted distribution (x-axis) against the empirical quantiles of the data) and a P-P plot (i.e. for each value of the data set, plot of the cumulative density function of the fitted distribution (x-axis) against the empirical cumulative density function (y-axis)). Grey rectangles in a Q-Q plot or a P-P plot also represent areas of non uniqueness of empirical quantiles or probabilities, directly derived from non uniqueness areas of the empirical cumulative distribution.

If NPMLE is TRUE, and NPMLE.method is "Turnbull", empirical and, if specified, theoretical distributions are plotted in cdf using the EM approach of Turnbull (Turnbull, 1974) to compute the overall empirical cdf curve, with confidence intervals if Turnbull.confint is TRUE, by calls to functions survfit and plot.survfit from the survival package.

If NPMLE is FALSE empirical and, if specified, theoretical distributions are plotted in cdf, with data directly reported as segments for interval, left and right censored data, and as points for non-censored data. Before plotting, observations are ordered and a rank r is associated to each of them. Left censored observations are ordered first, by their right bounds. Interval censored and non censored observations are then ordered by their mid-points and, at last, right censored observations are ordered by their left bounds. If leftNA (resp. rightNA) is finite, left censored (resp. right censored) observations are considered as interval censored observations and ordered by mid-points with non-censored and interval censored data. It is sometimes necessary to fix rightNA or leftNA to a realistic extreme value, even if not exactly known, to obtain a reasonable global ranking of observations. After ranking, each of the n observations is plotted as a point (one x-value) or a segment (an interval of possible x-values), with an y-value equal to r/n, r being the rank of each observation in the global ordering previously described. This second method may be interesting but is certainly less rigorous than the Turnbull method that should be prefered.
Author(s)

Marie-Laure Delignette-Muller and Christophe Dutang.

References


See Also

plotdist, survfit.formula.

Examples

# (1) Plot of an empirical censored distribution (censored data) as a CDF
# using the default Turnbull method
#
# data(smokedfish)
# d1 <- as.data.frame(log10(smokedfish))
# plotdistcens(d1)
#
# (2) Add the CDF of a normal distribution
# plotdistcens(d1,"norm",para=list(mean=-1.6,sd=1.5))
#
# (3) Various plots of the same empirical distribution
#
# # default Wang plot
# plotdistcens(d1, NPMLE = TRUE, NPMLE.method = "Wang")
# # Turnbull plot
# plotdistcens(d1, NPMLE = TRUE, NPMLE.method = "Turnbull")
# # Turnbull plot with confidence intervals
# plotdistcens(d1, NPMLE = TRUE, NPMLE.method = "Turnbull", Turnbull.confint = TRUE)
# # with intervals and points
# plotdistcens(d1, rightNA=3, NPMLE = FALSE)
# # with intervals and points
# # defining a minimum value for left censored values
# plotdistcens(d1, leftNA=-3, NPMLE = FALSE)
**prefit**  
*Pre-fitting procedure*

**Description**

Search good starting values

**Usage**

```
prefit(data, distr, method = c("mle", "mme", "qme", "mge"),
        feasible.par = NULL, order = NULL,
        probs = NULL, qtype = 7, gof = NULL, fix.arg = NULL, lower,
        upper, weights = NULL, silent = TRUE, ...)```

**Arguments**

- **data**
  A numeric vector.

- **distr**
  A character string "name" naming a distribution for which the corresponding density function `dname`, the corresponding distribution function ` pname` and the corresponding quantile function `qname` must be defined, or directly the density function.

- **method**
  A character string coding for the fitting method: "mle" for 'maximum likelihood estimation', "mme" for 'moment matching estimation', "qme" for 'quantile matching estimation' and "mge" for 'maximum goodness-of-fit estimation'.

- **feasible.par**
  A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see the 'details' section of `mledist`). It may not be into account for closed-form formulas.

- **order**
  A numeric vector for the moment order(s). The length of this vector must be equal to the number of parameters to estimate.

- **memp**
  A function implementing empirical moments, raw or centered but has to be consistent with `distr` argument (and `weights` argument).

- **probs**
  A numeric vector of the probabilities for which the quantile matching is done. The length of this vector must be equal to the number of parameters to estimate.

- **qtype**
  The quantile type used by the R `quantile` function to compute the empirical quantiles, (default 7 corresponds to the default quantile method in R).

- **gof**
### fix.arg
An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated by this maximum likelihood procedure. The use of this argument is not possible if method="mme" and a closed-form formula is used.

### weights
an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector. If non-NULL, weighted MLE is used, otherwise ordinary MLE.

### silent
A logical to remove or show warnings.

### lower
Lower bounds on the parameters.

### upper
Upper bounds on the parameters.

### ...
Further arguments to be passed to generic functions, or to one of the functions "mledist", "mmedist", "qmedist" or "mgedist" depending of the chosen method. See mledist, mmedist, qmedist, mgedist for details on parameter estimation.

### Details
Searching good starting values is achieved by transforming the parameters (from their constraint interval to the real line) of the probability distribution. Indeed,

- positive parameters in $(0, \infty)$ are transformed using the logarithm (typically the scale parameter sd of a normal distribution, see Normal),
- parameters in $(1, \infty)$ are transformed using the function $\log(x - 1)$,
- probability parameters in $(0, 1)$ are transformed using the logit function $\log(x/(1 - x))$ (typically the parameter prob of a geometric distribution, see Geometric),
- negative probability parameters in $(-1, 0)$ are transformed using the function $\log(-x/(1 + x))$,
- real parameters are of course not transformed at all, typically the mean of a normal distribution, see Normal.

Once parameters are transformed, an optimization is carried out by a quasi-Newton algorithm (typically BFGS) and then we transform them back to original parameter value.

### Value
A named list.

### Author(s)
Christophe Dutang and Marie-Laure Delignette-Muller.

### References
See Also

See \texttt{mledist}, \texttt{mmedist}, \texttt{qmedist}, \texttt{mgedist} for details on parameter estimation. See \texttt{fitdist} for the main procedure.

Examples

\footnotesize
\begin{verbatim}
# (1) fit of a gamma distribution by maximum likelihood estimation
#
x <- rgamma(1e3, 5/2, 7/2)

prefit(x, "gamma", "mle", list(shape=3, scale=3), lower=-Inf, upper=Inf)
\end{verbatim}

\normalsize

\textbf{qmedist} \hspace{1cm} \textit{Quantile matching fit of univariate distributions}

\section*{Description}

Fit of univariate distribution by matching quantiles for non censored data.

\section*{Usage}

\texttt{qmedist(data, distr, probs, start = NULL, fix.arg = NULL, qtype = 7,}
\texttt{ optim.method = "default", lower = -Inf, upper = Inf,}
\texttt{ custom.optim = NULL, weights = NULL, silent = TRUE, gradient = NULL,}
\texttt{ checkstartfix=FALSE, ...)}

\section*{Arguments}

\begin{itemize}
\item \textbf{data} \hspace{1cm} A numeric vector for non censored data.
\item \textbf{distr} \hspace{1cm} A character string "name" naming a distribution for which the corresponding quantile function \texttt{qname} and the corresponding density distribution \texttt{dname} must be classically defined.
\item \textbf{probs} \hspace{1cm} A numeric vector of the probabilities for which the quantile matching is done. The length of this vector must be equal to the number of parameters to estimate.
\item \textbf{start} \hspace{1cm} A named list giving the initial values of parameters of the named distribution or a function of data computing initial values and returning a named list. This argument may be omitted (default) for some distributions for which reasonable starting values are computed (see the 'details' section of \texttt{mledist}).
\item \textbf{fix.arg} \hspace{1cm} An optional named list giving the values of fixed parameters of the named distribution or a function of data computing (fixed) parameter values and returning a named list. Parameters with fixed value are thus NOT estimated.
\item \textbf{qtype} \hspace{1cm} The quantile type used by the R \texttt{quantile} function to compute the empirical quantiles, (default 7 corresponds to the default quantile method in R).
\end{itemize}
qmedist

- **optim.method**: "default" or optimization method to pass to `optim`.
- **lower**: Left bounds on the parameters for the "L-BFGS-B" method (see `optim`).
- **upper**: Right bounds on the parameters for the "L-BFGS-B" method (see `optim`).
- **custom.optim**: a function carrying the optimization.
- **weights**: an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector with strictly positive integers (typically the number of occurrences of each observation). If non-NULL, weighted QME is used, otherwise ordinary QME.
- **silent**: A logical to remove or show warnings when bootstrapping.
- **gradient**: A function to return the gradient of the squared difference for the "BFGS", "CG" and "L-BFGS-B" methods. If it is NULL, a finite-difference approximation will be used, see details.
- **checkstartfix**: A logical to test starting and fixed values. Do not change it.
- **...**: further arguments passed to the `optim`, `constrOptim` or `custom.optim` function.

**Details**

The `qmedist` function carries out the quantile matching numerically, by minimization of the sum of squared differences between observed and theoretical quantiles. Note that for discrete distribution, the sum of squared differences is a step function and consequently, the optimum is not unique, see the FAQ.

The optimization process is the same as `mledist`, see the 'details' section of that function.

Optionally, a vector of weights can be used in the fitting process. By default (when `weights=NULL`), ordinary QME is carried out, otherwise the specified weights are used to compute weighted quantiles used in the squared differences. Weighted quantiles are computed by `wtd.quantile` from the `Hmisc` package. It is not yet possible to take into account weights in functions `plotdist`, `plotdistcens`, `plot.fitdist`, `plot.fitdistcens`, `cdfcomp`, `cdffcomp`, `denscomp`, `ppcomp`, `qqcomp`, `gofstat` and `descdist` (developments planned in the future).

This function is not intended to be called directly but is internally called in `fitdist` and `bootdist`.

**Value**

`qmedist` returns a list with following components,

- **estimate**: the parameter estimates.
- **convergence**: an integer code for the convergence of `optim` defined as below or defined by the user in the user-supplied optimization function. 0 indicates successful convergence. 1 indicates that the iteration limit of `optim` has been reached. 10 indicates degeneracy of the Neelder-Mead simplex. 100 indicates that `optim` encountered an internal error.
- **value**: the minimal value reached for the criterion to minimize.
- **hessian**: a symmetric matrix computed by `optim` as an estimate of the Hessian at the solution found or computed in the user-supplied optimization function.
- **optim.function**: the name of the optimization function used for maximum likelihood.
optim.method when `optim` is used, the name of the algorithm used, the field method of the 
custom.optim function otherwise.

fix.arg the named list giving the values of parameters of the named distribution that 
must kept fixed rather than estimated by maximum likelihood or NULL if there 
are no such parameters.

fix.arg.fun the function used to set the value of fix.arg or NULL.

weights the vector of weights used in the estimation process or NULL.

counts A two-element integer vector giving the number of calls to the log-likelihood 
function and its gradient respectively. This excludes those calls needed to com-
pute the Hessian, if requested, and any calls to log-likelihood function to com-
pute a finite-difference approximation to the gradient. counts is returned by 
`optim` or the user-supplied function or set to NULL.

optim.message A character string giving any additional information returned by the optimizer,
or NULL. To understand exactly the message, see the source code.

loglik the log-likelihood value.

probs the probability vector on which quantiles are matched.

Author(s)

Christophe Dutang and Marie Laure Delignette-Muller.

References

Klugman SA, Panjer HH and Willmot GE (2012), *Loss Models: From Data to Decissions*, 4th 

Journal of Statistical Software, 64(4), 1-34.

See Also

`mmedist`, `mledist`, `mgedist`, `fitdist` for other estimation methods and `quantile` for empirical 
quantile estimation in R.

Examples

```
# (1) basic fit of a normal distribution
#
set.seed(1234)
x1 <- rnorm(n=100)
qmedist(x1, "norm", probs=c(1/3, 2/3))
```

```
# (2) defining your own distribution functions, here for the Gumbel 
# distribution for other distributions, see the CRAN task view dedicated 
# to probability distributions
```
quantiles

quantiles <- function(x, a, b) 1/b*exp((-a-x)/b)*exp(-exp((-a-x)/b))
qgumbel <- function(p, a, b) a - b*log(-log(p))
qmedist(x1, "gumbel", probs=c(1/3, 2/3), start=list(a=10, b=5))

# (3) fit a discrete distribution (Poisson)
#
set.seed(1234)
x2 <- rpois(n=30, lambda = 2)
qmedist(x2, "pois", probs=1/2)

# (4) fit a finite-support distribution (beta)
#
set.seed(1234)
x3 <- rbeta(n=100, shape1=5, shape2=10)
qmedist(x3, "beta", probs=c(1/3, 2/3))

# (5) fit frequency distributions on USAReats dataset.
#
x4 <- USAReats$Assault
qmedist(x4, "pois", probs=1/2)
qmedist(x4, "nbinom", probs=c(1/3, 2/3))

---

**quantiles**

*Quantile estimation from a fitted distribution*

**Description**

Quantile estimation from a fitted distribution, optionally with confidence intervals calculated from the bootstrap result.

**Usage**

```r
## S3 method for class 'fitdist'
quantile(x, probs = seq(0.1, 0.9, by=0.1), ...)
## S3 method for class 'fitdistcens'
quantile(x, probs = seq(0.1, 0.9, by=0.1), ...)
## S3 method for class 'bootdist'
quantile(x, probs = seq(0.1, 0.9, by=0.1), CI.type = "two.sided", CI.level = 0.95, ...)
## S3 method for class 'bootdistcens'
quantile(x, probs = seq(0.1, 0.9, by=0.1), CI.type = "two.sided", CI.level = 0.95, ...)
## S3 method for class 'quantile.fitdist'
print(x, ...)
## S3 method for class 'quantile.fitdistcens'
```

print(x, ...)  
## S3 method for class 'quantile.bootdist'
print(x, ...)  
## S3 method for class 'quantile.bootdistcens'
print(x, ...)

### Arguments

- **x**: An object of class "fitdist", "fitdistcens", "bootdist", "bootdistcens" or "quantile.fitdist", "quantile.fitdistcens", "quantile.bootdist", "quantile.bootdistcens" for the print generic function.

- **probs**: A numeric vector of probabilities with values in [0, 1] at which quantiles must be calculated.

- **CI.type**: Type of confidence intervals: either "two.sided" or one-sided intervals ("less" or "greater").

- **CI.level**: The confidence level.

- **...**: Further arguments to be passed to generic functions.

### Details

Quantiles of the parametric distribution are calculated for each probability specified in `probs`, using the estimated parameters. When used with an object of class "bootdist" or "bootdistcens", percentile confidence intervals and medians estimates are also calculated from the bootstrap result. If `CI.type` is "two.sided", the `CI.level` two-sided confidence intervals of quantiles are calculated. If `CI.type` is "less" or "greater", the `CI.level` one-sided confidence intervals of quantiles are calculated. The print functions show the estimated quantiles with percentile confidence intervals and median estimates when a bootstrap resampling has been done previously, and the number of bootstrap iterations for which the estimation converges if it is inferior to the whole number of bootstrap iterations.

### Value

`quantile` returns a list with 2 components (the first two described below) when called with an object of class "fitdist" or "fitdistcens" and 8 components (described below) when called with an object of class "bootdist" or "bootdistcens":

- **quantiles**: a dataframe containing the estimated quantiles for each probability value specified in the argument `probs` (one row, and as many columns as values in `probs`).

- **probs**: the numeric vector of probabilities at which quantiles are calculated.

- **bootquant**: A data frame containing the bootstrapped values for each quantile (many rows, as specified in the call to `bootdist` in the argument `niter`, and as many columns as values in `probs`).

- **quantCI**: If `CI.type` is "two.sided", the two bounds of the `CI.level` percent two.sided confidence interval for each quantile (two rows and as many columns as values in `probs`). If `CI.type` is "less", right bound of the `CI.level` percent one.sided confidence interval for each quantile (one row). If `CI.type` is "greater", left
quantiles

bound of the CI.level percent one.sided confidence interval for each quantile (one row).

quantmedian
Median of bootstrap estimates (per probability).

CI.type
Type of confidence interval: either "two.sided" or one-sided intervals ("less" or "greater").

CI.level
The confidence level.

nbboot
The number of samples drawn by bootstrap.

nbconverg
The number of iterations for which the optimization algorithm converges.

Author(s)
Marie-Laure Delignette-Muller and Christophe Dutang.

References

See Also
fitdist, bootdist, fitdistcens, bootdistcens and CICdfplot.

Examples

# (1) Fit of a normal distribution on acute toxicity log-transformed values of endosulfan for nonarthropod invertebrates, using maximum likelihood estimation # to estimate what is called a species sensitivity distribution # (SSD) in ecotoxicology, followed by estimation of the 5, 10 and 20 percent quantile # values of the fitted distribution, which are called the 5, 10, 20 percent hazardous # concentrations (HC5, HC10, HC20) in ecotoxicology, followed with calculations of their # confidence intervals with various definitions, from a small number of bootstrap # iterations to satisfy CRAN running times constraint. # For practical applications, we recommend to use at least niter=501 or niter=1001. #

data(endosulfan)
ATV <- subset(endosulfan, group == "NonArthroInvert")$ATV
log10ATV <- log10(subset(endosulfan, group == "NonArthroInvert")$ATV)
fln <- fitdist(log10ATV, "norm")
quantile(fln, probs = c(0.05, 0.1, 0.2))
bln <- bootdist(fln, bootmethod="param", niter=101)
quantile(bln, probs = c(0.05, 0.1, 0.2))
quantile(bln, probs = c(0.05, 0.1, 0.2), CI.type = "greater")
quantile(bln, probs = c(0.05, 0.1, 0.2), CI.level = 0.9)

# (2) Draw of 95 percent confidence intervals on quantiles of the # previously fitted distribution #
cdfcomp(fln)
q1 <- quantile(bln, probs = seq(0.1, length=101))
points(q1$quantCI[1,],q1$probs,type="l")
species-sensitivity distribution (ssd) for salinity tolerance

description

72-hour acute salinity tolerance (lc50 values) of riverine macro-invertebrates.

usage

data(salinity)

format

salinity is a data frame with 2 columns named left and right, describing each observed lc50 value (in electrical conductivity, millisiemens per centimeter) as an interval. the left column contains either na for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. the right column contains either na for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.
Source

Examples

```r
# (1) load of data
# data(salinity)

# (2) plot of data using Turnbull cdf plot
# log10LC50 <- log10(salinity)
plotdistcens(log10LC50)

# (3) fit of a normal and a logistic distribution to data in log10
# (classical distributions used for species sensitivity
# distributions, SSD, in ecotoxicology)
# and visual comparison of the fits using Turnbull cdf plot
# fln <- fitdistcens(log10LC50, "norm")
summary(fln)

fl1 <- fitdistcens(log10LC50, "logis")
summary(fl1)

cdfcomp(cens(list(fln, fl1)), legendtext=c("normal", "logistic"),
  xlab = "log10(LC50)", xlim=c(0.5,2), lines=1 = TRUE)

# (4) estimation of the 5 percent quantile value of
# the normal fitted distribution (5 percent hazardous concentration : HC5)
# with its two-sided 95 percent confidence interval calculated by
# non parametric bootstrap
# from a small number of bootstrap iterations to satisfy CRAN running times constraint.
# For practical applications, we recommend to use at least niter=501 or niter=1001.
# # in log10(LC50)
bln <- bootdistcens(fln, niter=101)
HC5ln <- quantile(bln, probs = 0.05)
# in LC50
10^(HC5ln$quantiles)
10^*(HC5ln$quantCI)

# (5) estimation of the HC5 value
# with its one-sided 95 percent confidence interval (type "greater")
# # in log10(LC50)
HC5lnb <- quantile(bln, probs = 0.05,CI.type="greater")
# in LC50
10^(HC5lnb$quantiles)
```
Description

Contamination data of *Listeria monocytogenes* in smoked fish on the Belgian market in the period 2005 to 2007.

Usage

data(smokedfish)

Format

`smokedfish` is a data frame with 2 columns named left and right, describing each observed value of *Listeria monocytogenes* concentration (in CFU/g) as an interval. The left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.

Source


Examples

```r
# (1) load of data
#
data(smokedfish)

# (2) plot of data in CFU/g
#
plotdistcens(smokedfish)

# (3) plot of transformed data in log10[CFU/g]
#
Clog10 <- log10(smokedfish)
plotdistcens(Clog10)

# (4) Fit of a normal distribution to data in log10[CFU/g]
#
fitlog10 <- fitdistcens(Clog10,"norm")
summary(fitlog10)
```
Description

Toxocara cati abundance in feral cats living on Kerguelen island.

Usage

data(toxocara)

Format

toxocara is a data frame with 1 column (number: number of parasites in digestive tract)

Source


Examples

# (1) load of data
#
data(toxocara)

# (2) description and plot of data
#
n <- toxocara$number
descdist(n, discrete=TRUE, boot=1000)
plotdist(n, discrete=TRUE)

# (3) fit of a Poisson distribution to data
#
fitp <- fitdist(n, "pois")
summary(fitp)
plot(fitp)

# (4) fit of a negative binomial distribution to data
#
fitnb <- fitdist(n, "nbinom")
summary(fitnb)
plot(fitnb)
Index

*Topic datasets
  danish, 14
dataFAQ, 16
endosulfan, 20
fluazinam, 37
groundbeef, 51
salinity, 82
smokedfish, 84
toxocara, 85

*Topic distribution
  bootdist, 4
bootdistcens, 7
CICdfplot, 11
descdist, 16
detectbound, 19
fitdist, 22
fitdistcens, 32
goFstat, 39
graphcomp, 43
graphcompCens, 48
logLikPlot, 52
logLikSurface, 54
mgedist, 57
mledist, 60
mmmedist, 65
plotdist, 69
plotdistCens, 71
prefit, 74
qmedist, 76
quantiles, 79

bootdist, 3, 4, 13, 24–26, 58, 61, 63, 66, 67, 77, 80, 81
bootdistcens, 3, 7, 34, 61, 63, 67, 81
cdfcomp, 3, 12, 13, 23, 25, 70
cdfcomp (graphcomp), 43
cdfcompCens, 3, 12, 13
cdfcompCens (graphcompCens), 48
CIcdfplot, 3, 6, 9, 11, 26, 45, 81

cof.fitdist (fitdist), 22
cof.fitdistCens (fitdistCens), 32
colorRampPalette, 4, 5
constrOptim, 25, 34, 58, 61–63, 66, 77
contour, 53, 56
danish, 14
danishmulti (danish), 14
danishuni (danish), 14
dataFAQ, 16
dataFAQlog1 (dataFAQ), 16
dataFAQscale1 (dataFAQ), 16
dataFAQscale2 (dataFAQ), 16
denscomp, 3, 25, 70
denscomp (graphcomp), 43
denscompCens (graphcompCens), 48
density, 44, 69
descdist, 3, 16, 70
detectbound, 19
endosulfan, 20
fitdist, 3–6, 19, 22, 35, 41, 52, 58, 59, 61–63, 66, 67, 76–78, 81
fitdistCens, 3, 8, 9, 26, 32, 52, 61–63, 67, 81
fitdistPlus, 6, 9, 26, 35
fitdistPlus (fitdistPlus-package), 3
fitdistPlus-package, 3
fluazinam, 37

Geometric, 75
goFstat, 3, 23, 25, 26, 39
graphcomp, 26, 42, 70
graphcompCens, 48
graphics, 45, 49
groundbeef, 51

hist, 44, 45, 69, 70

image, 5, 53, 56
INDEX

legend, 44, 45, 49, 50
llcurve, 53
llcurve (logLik.surface), 54
llplot, 3, 26, 55, 56, 63
llplot (logLikplot), 52
llsurface, 53
llsurface (logLik.surface), 54
logLik.fitdist (fitdist), 22
logLik.fitdistcens (fitdistcens), 32
logLikplot, 52
logLiksurface, 54
mge (mgedist), 57
mgedist, 3, 5, 6, 23–26, 57, 61, 63, 67, 75, 76, 78
mle (mledist), 60
mledist, 3, 5, 6, 8, 9, 23–26, 33, 35, 57–59, 60, 65, 66, 74–78
mme (mmedist), 65
mmmedist, 3, 5, 6, 23–26, 59, 61, 63, 65, 67, 75, 76, 78
Normal, 75
npsurv, 72


pairs, 5
par, 12, 43, 44, 49, 50
plot, 5, 6, 8, 9, 45, 53, 56, 70
plot.bootdist (bootdist), 4
plot.bootdistcens (bootdistcens), 7
plot.default, 44
plot.fitdist (fitdist), 22
plot.fitdistcens (fitdistcens), 32
plot.stepfun, 12, 44, 45
plot.survfit, 72
plotdist, 3, 18, 23, 25, 26, 45, 69, 73
plotdistcens, 3, 35, 49, 50, 70, 71
points, 12
ppcomp, 3, 25, 70
ppcomp (graphcomp), 43
ppcomp.cens (graphcomp.cens), 48
points, 12, 44, 45, 70
prefit, 3, 74
print.bootdist (bootdist), 4
print.bootdistcens (bootdistcens), 7
print.descdist (descdist), 16
print.fitdist (fitdist), 22
print.fitdistcens (fitdistcens), 32
print.gofstat (gofstat), 39
print.quantile.bootdist (quantiles), 79
print.quantile.bootdistcens
(quantiles), 79
print.quantile.fitdist (quantiles), 79
print.quantile.fitdistcens (quantiles), 79
qme (qmedist), 76
qmedist, 3, 5, 6, 23–26, 59, 61, 63, 67, 75, 76, 78
qqcomp, 3, 25, 70
qqcomp (graphcomp), 43
qqcomp.cens (graphcomp.cens), 48
quantile, 3, 13, 26, 74, 76, 78
quantile.bootdist, 6
quantile.bootdist (quantiles), 79
quantile.bootdistcens, 9
quantile.bootdistcens (quantiles), 79
quantile.fitdist, 26
quantile.fitdist (quantiles), 79
quantile.fitdistcens, 35
quantile.fitdistcens (quantiles), 79
quantiles, 79
salinity, 82
smokedfish, 84
stripchart, 5, 6, 8, 9
summary.bootdist (bootdist), 4
summary.bootdistcens (bootdistcens), 7
summary.fitdist (fitdist), 22
summary.fitdistcens (fitdistcens), 32
survfit, 72
survfit.formula, 50, 73
title, 4, 11, 43, 48
toxocara, 85

vcov.fitdist (fitdist), 22
vcov.fitdistcens (fitdistcens), 32

wtd.mean, 66
wtd.quantile, 77
wtd.var, 66

xy.coords, 44, 49