Package ‘extremeStat’

November 16, 2022

Type Package

Title Extreme Value Statistics and Quantile Estimation

Version 1.5.3

Date 2022-11-16

Depends R (>= 2.10)

Imports lmomco (>= 2.2.5), berryFunctions (>= 1.15.6), pbapply, RColorBrewer, grDevices, graphics, methods, stats, utils, evir, ismev, fExtremes, extRemes, evd, Renext

Author Berry Boessenkool

Maintainer Berry Boessenkool <berry-b@gmx.de>

Description Fit, plot and compare several (extreme value) distribution functions. Compute (truncated) distribution quantile estimates and plot return periods on a linear scale. On the fitting method, see Asquith (2011): Distributional Analysis with L-moment Statistics [...] ISBN 1463508417.

License GPL (>= 2)

URL https://github.com/brry/extremeStat

RoxygenNote 7.2.1

Encoding UTF-8

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

BugReports https://github.com/brry/extremeStat

NeedsCompilation no

Repository CRAN

Date/Publication 2022-11-16 14:30:04 UTC

R topics documented:

annMax ................................................................. 2
distLexBoot ......................................................... 3
annMax

annual discharge maxima (streamflow)

Description

Annual discharge maxima of a stream in Austria called Griesler or Fuschler Ache, at the measurement station (gauge) near St. Lorenz, catchment area ca 100 km^2. Extracted from the time series 1976-2010 with a resolution of 15 Minutes.

Format

num [1:35] 61.5 77 37 69.3 75.6 74.9 43.7 50.8 55.6 84.1 ...

Source

Hydrographische Dienste Oberoesterreich und Salzburg, analyzed by package author (<berry-b@gmx.de>)

Examples

data(annMax)
str(annMax)
str(annMax)
plot(1976:2010, annMax, type="l", las=1, main="annMax dataset from Austria")
# Moving Average with different window widths:
berryFunctions::movAvLines(annMax, x=1976:2010, lwd=3, alpha=0.7)
distLexBoot

**Description**
Calculates and plots bootstrap uncertainty intervals for `plotLextreme`.

**Usage**
```
distLexBoot(
  dlf,  
  nbest = 3, 
  selection = NULL, 
  n = 100, 
  prop = 0.8, 
  conf.lev = 0.95, 
  replace = FALSE, 
  RPs = NULL, 
  log = TRUE, 
  progbars = TRUE, 
  quiet = FALSE
)
```

**Arguments**
- **dlf**: dlf object, as returned by `distLextreme`
- **nbest**: Number of best fitted distribution functions in dlf for which bootstrapping is to be done. Overridden by `selection`. DEFAULT: 3
- **selection**: Character vector with distribution function names to be used. Suggested to keep this low. DEFAULT: NULL
- **n**: Number of subsamples to be processed (computing time increases extraordinarily). DEFAULT: 100
- **prop**: Proportion of sample to be used in each run. DEFAULT: 0.8
- **conf.lev**: Confidence level (Proportion of subsamples within 'confidence interval'). Quantiles extracted from this value are passed to `quantileMean`. DEFAULT: 0.95
- **replace**: Logical: replace in each sample? DEFAULT: FALSE
- **RPs**: Return Period vector, by default calculated internally based on value of `log`. DEFAULT: NULL
- **log**: RPs suitable for plot on a logarithmic axis? DEFAULT: TRUE
- **progbars**: Show progress bar for Monte Carlo simulation? DEFAULT: TRUE
- **quiet**: Logical: suppress messages? See `distLquantile`. DEFAULT: FALSE

**Details**
Has not been thoroughly tested yet. Bootstrapping defaults can probably be improved.
Value

invisible dlf object, see printL. Additional elements are: exBootCL (confidence level), exBootRPs (x values for plot) exBootSim (all simulation results) and exBootCI (aggregated into CI band). The last two are each a list with a matrix (return levels).

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2015 + Dec 2016

See Also

plotLexBoot, distLex

Examples

data(annMax)
dlf <- distLex(annMax, selection=c("gum","gev","wak","nor"))
dlfB <- distLexBoot(dlf, nbest=4, conf.lev=0.5, n=10) # n low for quick example tests
plotLexBoot(dlfB)
plotLexBoot(dlfB, selection=c("nor","gev"))
plotLexBoot(dlfB, selection=c("gum","gev","wak","nor"), order=FALSE)

---

distLex

Extreme value stats

Description

Extreme value statistics for flood risk estimation. Input: vector with annual discharge maxima (or all observations for POT approach). Output: discharge estimates for given return periods, parameters of several distributions (fit based on L-moments), quality of fits, plot with linear/logarithmic axis. (plotting positions by Weibull and Gringorton).

Usage

distLex(  
  dat = NULL,
  dlf = NULL,
  RPs = c(2, 5, 10, 20, 50),
  npy = 1,
  truncate = 0,
  quiet = FALSE,
  ...
)
Arguments

- **dat**: Vector with *either* (for Block Maxima Approach) extreme values like annual discharge maxima *or* (for Peak Over Threshold approach) all values in time-series. Ignored if dlf is given. DEFAULT: NULL
- **dlf**: List as returned by `distLfit`. See also `distLquantile`. Overrides dat! DEFAULT: NULL
- **RPs**: Return Periods (in years) for which discharge is estimated. DEFAULT: c(2,5,10,20,50)
- **npy**: Number of observations per year. Leave npy=1 if you use annual block maxima (and leave truncate at 0). If you use a POT approach (see vignette and examples below) e.g. on daily data, use npy=365.24. DEFAULT: 1
- **truncate**: Truncated proportion to determine POT threshold, see `distLquantile`. DEFAULT: 0
- **quiet**: Suppress notes and progbars? DEFAULT: FALSE
- **...**: Further arguments passed to `distLquantile` like truncate, selection, time, progbars

Details

- **plotLextreme** adds weibull and gringorton plotting positions to the distribution lines, which are estimated from the L-moments of the data itself.
- I personally believe that if you have, say, 35 values in dat, the highest return period should be around 36 years (Weibull) and not 60 (Gringorton).
- The plotting positions don’t affect the distribution parameter estimation, so this dispute is not really important. But if you care, go ahead and google "weibull vs gringorton plotting positions".
- Plotting positions are not used for fitting distributions, but for plotting only. The ranks of ascendingly sorted extreme values are used to compute the probability of non-exceedance Pn:
  
  \[
  P_n_w <- \text{Rank } / (n+1) \quad \# \text{Weibull}
  \]
  
  \[
  P_n_g <- (\text{Rank } - 0.44) / (n+0.12) \quad \# \text{Gringorton (taken from lmom:::evplot.default)}
  \]
  
  Finally: \( RP = \text{Return period} = \text{recurrence interval} = 1 / P_{\text{exceedance}} = 1 / (1 - P_{\text{nonexc}}), \) thus:
  
  \[
  RP_{\text{weibull}} = 1 / (1 - P_n_w)
  \]
  
  and analogous for gringorton.

Value

invisible dlf object, see `printL`. The added element is returnlev, a data.frame with the return level (discharge) for all given RPs and for each distribution. Note that this differs from `distLquantile` (matrix output, not data.frame)

Note

This function replaces `berryFunctions:::extremeStatLmom`

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, 2012 (first draft) - 2014 & 2015 (main updates)
distLextreme

References

https://RclickHandbuch.wordpress.com Chapter 15 (German)
Christoph Mudersbach: Untersuchungen zur Ermittlung von hydrologischen Bemessungsgroessen mit Verfahren der instationaeren Extremwertstatistik

See Also
distLfit, distLexBoot for confidence interval from Bootstrapping. fevd in the package extRemes.

Examples

# Basic examples
# BM vs POT
# Plotting options
# weighted mean based on Goodness of fit (GOF)
# Effect of data proportion used to estimate GOF
# compare extremeStat with other packages

library(lmomco)
library(berryFunctions)
data(annMax)  # annual streamflow maxima in river in Austria

# Basic examples ---------------------------------------------------------------
dlf <- distLextreme(annMax)
plotLextreme(dlf, log=TRUE)
plotLextreme(dlf, log="xy")
plotLextreme(dlf)

# Object structure:
str(dlf, max.lev=2)
printL(dlf)

# discharge levels for default return periods:
dl$f$returnlev

# Estimate discharge that could occur every 80 years (at least empirically):
Q80 <- distLextreme(dlf=dlf, RPs=80)$returnlev
round(sort(Q80[1:17,1]),1)
# 99 to 143 m^3/s can make a relevant difference in engineering!
# That's why the rows weighted by GOF are helpful. Weights are given as in plotLweights(dlf) # See also section weighted mean below
# For confidence intervals see ?distLexBoot

# Return period of a given discharge value, say 120 m^3/s:
round0(sort(1/(1-sapply(dlf$parameter, plmomco, x=120) ) ),1)
# exponential: every 29 years
# gev (general extreme value dist): 59,
# Weibull: every 73 years only
distLextreme

# BM vs POT -----------------------------------------------
# Return levels by Block Maxima approach vs Peak Over Threshold approach:
# BM distribution theoretically converges to GEV, POT to GPD

data(rain, package="ismev")
days <- seq(as.Date("1914-01-01"), as.Date("1961-12-30"), by="days")
BM <- tapply(rain, format(days,"%Y"), max) ; rm(days)
dlfBM <- plotLextreme(distLextreme(BM, emp=FALSE), ylim=lim0(100), log=TRUE, nbest=10)
plotLexBoot(distLexBoot(dlfBM, quiet=TRUE), ylim=lim0(100))
plotLextreme(dlfBM, log=TRUE, ylim=lim0(100))

dlfPOT99 <- distLextreme(rain, npy=365.24, trunc=0.99, emp=FALSE)
dlfPOT99 <- plotLextreme(dlfPOT99, ylim=lim0(100), log=TRUE, nbest=10, main="POT 99")
printL(dlfPOT99)
# using only nonzero values (normally yields better fits, but not here)
rainnz <- rain[rain>0]
dlfPOT99nz <- distLextreme(rainnz, npy=length(rainnz)/48, trunc=0.99, emp=FALSE)
dlfPOT99nz <- plotLextreme(dlfPOT99nz, ylim=lim0(100), log=TRUE, nbest=10,
                         main=paste("POT 99 x>0, npy =", round(dlfPOT99nz$npy,2)))

## Not run: ## Excluded from CRAN R CMD check because of computing time

dlfPOT99boot <- distLexBoot(dlfPOT99, prop=0.4)
printL(dlfPOT99boot)
plotLexBoot(dlfPOT99boot)

dlfPOT90 <- distLextreme(rain, npy=365.24, trunc=0.90, emp=FALSE)
dlfPOT90 <- plotLextreme(dlfPOT90, ylim=lim0(100), log=TRUE, nbest=10, main="POT 90")
dlfPOT50 <- distLextreme(rain, npy=365.24, trunc=0.50, emp=FALSE)
dlfPOT50 <- plotLextreme(dlfPOT50, ylim=lim0(100), log=TRUE, nbest=10, main="POT 50")

## End(Not run)

ig99 <- ismev::gpd.fit(rain, dlfPOT99$threshold)
ismev::gpd.diag(ig99); title(main=paste(99, ig99$threshold))

## Not run:
ig90 <- ismev::gpd.fit(rain, dlfPOT90$threshold)
ismev::gpd.diag(ig90); title(main=paste(90, ig90$threshold))
ig50 <- ismev::gpd.fit(rain, dlfPOT50$threshold)
ismev::gpd.diag(ig50); title(main=paste(50, ig50$threshold))

## End(Not run)

# Plotting options -----------------------------------------------
plotLextreme(dlf=dlf)
# Line colors / select distributions to be plotted:
plotLextreme(dlf, nbest=17, distcols=heat.colors(17), lty=1:5) # lty is recycled
plotLextreme(dlf, selection=c("gev", "gam", "gum"), distcols=4:6, PPcol=3, lty=3:2)
plotLextreme(dlf, selection=c("gpa", "glo", "wei", "exp"), pch=c(NA,NA,6,8),
# weighted mean based on Goodness of fit (GOF) ---------------------------------
# Add discharge weighted average estimate continuously:
plotLextreme(dlf, nbest=17, legend=FALSE)
abline(h=115.6, v=50)
RP <- seq(1, 70, len=100)
DischargeEstimate <- distLextreme(dlf=dlf, RPs=RP, plot=FALSE)$returnlev
lines(RP, DischargeEstimate["weighted2",], lwd=3, col="orange")

# Or, on log scale:
plotLextreme(dlf, nbest=17, legend=FALSE, log=TRUE)
abline(h=115.9, v=50)
RP <- unique(round(logSpaced(min=1, max=70, n=200, plot=FALSE),2))
DischargeEstimate <- distLextreme(dlf=dlf, RPs=RP)$returnlev
lines(RP, DischargeEstimate["weighted2",], lwd=5)

# Minima -----------------------------------------------------------------------

browseURL("https://nrfa.ceh.ac.uk/data/station/meanflow/39072")
qfile <- system.file("extdata/discharge39072.csv", package="berryFunctions")
Q <- read.table(qfile, skip=19, header=TRUE, sep=","), fill=TRUE)
rm(qfile)
colnames(Q) <- c("date","discharge")
Q$date <- as.Date(Q$date)
plot(Q, type="l")
Qmax <- tapply(Q$discharge, format(Q$date,"%Y"), max)
plotLextreme(distLextreme(Qmax, quiet=TRUE))
Qmin <- tapply(Q$discharge, format(Q$date,"%Y"), min)
dlf <- distLextreme(-Qmin, quiet=TRUE, RPs=c(2,5,10,20,50,100,200,500))
plotLextreme(dlf, ylim=c(0,-31), yaxs="i", yaxt="n", ylab="Q annual minimum", nbest=14)
axis(2, -(0:3*10), 0:3*10, las=1)
-dlf$returnlev[c(1:14,21),]

# Some distribution functions are an obvious bad choice for this, so I use
# weighted 3: Values weighted by GOF of dist only for the best half.
# For the Thames in Windsor, we will likely always have > 9 m3/s streamflow
distLfit

# compare extremeStat with other packages: ---------------------------------------
library(extRemes)
plot(fevd(annMax))
par(mfrow=c(1,1))
return.level(fevd(annMax, type="GEV")) # "GP", "PP", "Gumbel", "Exponential"
distLextreme(dlf=dlf, RPs=c(2,20,100))$returnlev["gev",]
# differences are small, but noticeable...
# if you have time for a more thorough control, please pass me the results!

# yet another dataset for testing purposes:
Dresden_AnnualMax <- c(403, 468, 497, 539, 542, 634, 662, 765, 834, 847, 851, 873,
885, 983, 996, 1020, 1028, 1090, 1096, 1110, 1173, 1180, 1180,
1220, 1270, 1285, 1329, 1360, 1360, 1387, 1401, 1410, 1410, 1456,
1556, 1580, 1610, 1630, 1680, 1734, 1740, 1748, 1780, 1800, 1820,
plotLextreme(distLextreme(Dresden_AnnualMax))

## End(Not run) # end dontrun

---

**distLfit**

Fit distributions via L-moments

**Description**

Fit several distributions via L-moments with `lmomco::lmom2par` and compute goodness of fit measures.

**Usage**

```r
distLfit(
dat,
datname = deparse(substitute(dat)),
selection = NULL,
speed = TRUE,
ks = FALSE,
truncate = 0,
threshold = berryFunctions::quantileMean(dat, truncate),
progbars = length(dat) > 200,
time = TRUE,
quiet = FALSE,
ssquiet = quiet,
...)
```


Arguments

dat Vector with values
datname Character string for main, xlab etc. DEFAULT: deparse(substitute(dat))
selection Selection of distributions. Character vector with types as in lmom2par. Overrides speed. DEFAULT: NULL
speed If TRUE, several distributions are omitted, for the reasons shown in lmomco::dist.list(). DEFAULT: TRUE
ks Include ks.test results and CDF R^2 in dlf$gof? Computing is much faster when FALSE. DEFAULT: FALSE
truncate Number between 0 and 1. POT Censored distLquantile: fit to highest values only (truncate lower proportion of x). Probabilities are adjusted accordingly. DEFAULT: 0
threshold POT cutoff value. If you want correct percentiles, set this only via truncate, see Details of q_gpd. DEFAULT: quantileMean(x, truncate)
progbars Show progress bars for each loop? DEFAULT: TRUE if n > 200
time message execution time? DEFAULT: TRUE
quiet Suppress notes? DEFAULT: FALSE
ssquiet Suppress sample size notes? DEFAULT: quiet
... Further arguments passed to distLweights like weightc, order=FALSE

Value

invisible dlf object, see printL.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014, July 2015, Dec 2016

See Also

plotLfit, distLweights, plotLweights, extRemes::fevd, MASS::fitdistr.


Examples

data(annMax)
# basic usage on real data (annual discharge maxima in Austria)
dlf <- distLfit(annMax)
str(dlf, max.lev=2)
printL(dlf)
plotLfit(dlf)

# arguments that can be passed to plotting function:
plotLfit(dlf, lty=2, col=3, nbest=17, legargs=list(lwd=3), main="booh!")
```
set.seed(42)
dlf_b <- distLfit(rbeta(100, 5, 2))
plotLfit(dlf_b, nbest=10, legargs=c(x="left"))
plotLfit(dlf_b, selection=c("gpa", "glo", "gev", "wak"))
plotLfit(dlf_b, selection=c("gpa", "glo", "gev", "wak"), order=TRUE)
plotLfit(dlf_b, distcols=c("orange",3:6), lty=1:3) # lty is recycled
plotLfit(dlf_b, cdf=TRUE)
plotLfit(dlf_b, cdf=TRUE, histargs=list(do.points=FALSE), sel="nor")

# logarithmic axes:
set.seed(1)
y <- 10^rnorm(300, mean=2, sd=0.3) # if you use 1e4, distLfit will be much slower
hist(y, breaks=20)
berryFunctions::logHist(y, col=8)
dlf <- distLfit(log10(y))
plotLfit(dlf, breaks=50)
plotLfit(dlf, breaks=50, log=TRUE)

# Goodness of fit: how well do the distributions fit the original data?
# measured by RMSE of cumulated distribution function and ?ecdf
# RMSE: root of average of ( errors squared ) , errors = line distances
dlf <- distLfit(annMax, ks=TRUE)
plotLfit(dlf, cdf=TRUE, sel=c("wak", "revgum"))
x <- sort(annMax)
segments(x0=x, y0=lmomco::plmomco(x, dlf$parameter$revgum), y1=ecdf(annMax)(x), col=2)
segments(x0=x, y0=lmomco::plmomco(x, dlf$parameter$wak), y1=ecdf(annMax)(x), col=4, lwd=2)
# weights by three different weighting schemes, see distLweights:
plotLweights(dlf)
plotLfit(distLfit(annMax ), cdf=TRUE, nbest=17)$gof
plotLfit(distLfit(annMax, truncate=0.7), cdf=TRUE, nbest=17)$gof
pairs(dlf$gof[-(2:5)]) # measures of goodness of fit are correlated quite well here.
dlf$gof

# Kolmogorov-Smirnov Tests for normal distribution return slightly different values:
library(lmomco)
ks.test(annMax, "pnorm", mean(annMax), sd(annMax))$p.value
ks.test(annMax, "cdfnor", parnor(lmoms(annMax)))$p.value

# Fit all available distributions (30):
## Not run: # this takes a while...
d_all <- distLfit(annMax, speed=FALSE, progbars=TRUE) # 20 sec
printL(d_all)
plotLfit(d_all, nbest=30, distcols=grey(1:22/29), xlim=c(20,140))
plotLfit(d_all, nbest=30, ylim=c(0,0.04), xlim=c(20,140))
plotLweights(d_all)
d_all$gof

## End(Not run)
```
distLquantile  
distribution quantiles

Description

Parametric quantiles of distributions fitted to a sample.

Usage

distLquantile(
  x = NULL,
  probs = c(0.8, 0.9, 0.99),
  truncate = 0,
  threshold = quantileMean(dlf$dat_full[is.finite(dlf$dat_full)], truncate),
  sanerange = NA,
  sanevals = NA,
  selection = NULL,
  order = TRUE,
  dlf = NULL,
  datname = deparse(substitute(x)),
  list = FALSE,
  empirical = TRUE,
  qemp.type = 8,
  weighted = empirical,
  gpd = empirical,
  speed = TRUE,
  quiet = FALSE,
  ssquiet = quiet,
  ttquiet = quiet,
  gpquiet = missing(quiet) | quiet,
  ...
)

Arguments

x  Sample for which parametric quantiles are to be calculated. If it is NULL (the
default), dat from dlf is used. DEFAULT: NULL
probs  Numeric vector of probabilities with values in [0,1]. DEFAULT: c(0.8,0.9,0.99)
truncate  Number between 0 and 1 (proportion of sample discarded). Censored quantile: fit to highest values only (truncate lower proportion of x). Probabilities are adjusted accordingly. DEFAULT: 0
threshold  POT cutoff value. If you want correct percentiles, set this only via truncate, see Details of q_gpd. DEFAULT: quantileMean(x, truncate)
sanerange  Range outside of which results should be changed to sanevals. This can capture numerical errors in small samples (notably GPD_MLE_extRemes). If NA, this is ignored. Attention: the RMSE column is also checked and changed. DEFAULT: NA
sanevals  Values to be used below [1] and above [2] sanerange. DEFAULT: NA
selection  Distribution type, eg. "gev" or "wak", see lmomco:::dist.list. Can be a vector. If NULL (the default), all types present in dlf$distnames are used. DEFAULT: NULL
order  Logical: sort by RMSE, even if selection is given? See distLweights. DEFAULT: TRUE
dlf  dlf object described in extremeStat. Use this to save computing time for large datasets where you already have dlf. DEFAULT: NULL
datname  Character string: data name, important if list=TRUE. DEFAULT: deparse(substitute(x))
list  Return full dlf list with output attached as element quant? If FALSE (the default), just the matrix with quantile estimates is returned. DEFAULT: FALSE
empirical  Add rows "empirical" and "quantileMean" in the output matrix? Uses quantile with qemp.type (ignoring truncation) and quantileMean. DEFAULT: TRUE
qemp.type  Method passed to quantile for row "empirical". Only used if empirical=TRUE. DEFAULT: 8 (NOT the stats::quantile default)
weighted  Include weighted averages across distribution functions to the output? DEFAULT: empirical, so additional options can all be excluded with emp=F.
gpd  Include GPD quantile estimation via q_gpd? Note that the 'GPD_LMO_lmomco' result differs slightly from 'gpa', especially if truncate=0. This comes from using x>threshold (all 'GPD_*' distributions) or x>=threshold ('gpa' and all other distributions in extremeStat). DEFAULT: empirical
speed  Compute q_gpd only for fast methods? Currently, only the Bayesian method is excluded. DEFAULT: TRUE
quiet  Suppress notes? If it is actually set to FALSE (not missing), gpquiet is set to FALSE to print all the warnings including stacks. DEFAULT: FALSE
ssquiet  Suppress sample size notes? DEFAULT: quiet
ttquiet  Suppress truncation!=threshold note? DEFAULT: quiet
gpquiet  Suppress warnings in q_gpd? DEFAULT: TRUE if quiet is not specified, else quiet
...  Arguments passed to distLfit and distLweights like weightc, ks=TRUE

Details

Very high quantiles (99% and higher) need large sample sizes for quantile to yield a robust estimate. Theoretically, at least 1/(1-probs) values must be present, e.g. 10’000 for Q99.99%. With smaller sample sizes (eg n=35), they underestimate the actual (but unknown) quantile. Parametric quantiles need only small sample sizes. They don’t have a systematical underestimation bias, but have higher variability.

Value

if list=FALSE (default): invisible matrix with distribution quantile values . if list=TRUE: invisible dlf object, see printL
Note

NAs are always removed from x in distLfit

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, March + July 2015, Feb 2016

References

On GPD: https://stats.stackexchange.com/questions/69438

See Also


Examples

data(annMax) # Annual Discharge Maxima (streamflow)
distLquantile(annMax, emp=FALSE)[,] # several distribution functions in lmomco

## Not run:
## Taken out from CRAN package check because it's slow
distLquantile(annMax, truncate=0.8, probs=0.95)[,] # POT (annMax already block maxima)
dlf <- distLquantile(annMax, probs=0.95, list=TRUE)
plotLquantile(dlf, linargs=list(lwd=3), nbest=5, breaks=10)
dlf$quant
# Parametric 95% quantile estimates range from 92 to 111!
# But the best fitting distributions all lie around 103.

# compare General Pareto Fitting methods
# Theoretically, the tails of distributions converge to GPD (General Pareto)
# q_gpd compares several R packages for fitting and quantile estimation:
dlq <- distLquantile(annMax, weighted=FALSE, quiet=TRUE, probs=0.97, list=TRUE)
dlq$quant
plotLquantile(dlq) # per default best fitting distribution functions
plotLquantile(dlq, row=c("wak","GPDx"), nbest=14)

# Sanity checks: important for very small samples:
x1 <- c(2.6, 2.5, 2.9, 3, 5, 2.7, 2.7, 5.7, 2.8, 3.1, 3.6, 2.6, 5.8, 5.6, 5.7, 5.3)
q1 <- distLquantile(x1, sanerange=c(0,500), sanevals=c(NA,500))
x2 <- c(6.1, 2.4, 4.1, 2.4, 6, 6.3, 2.9, 6.8, 3.5)
\begin{verbatim}
 # weighted distribution quantiles are calculated by different weighting schemes:
 # plotWeights(dlf)
 # If speed is important and parameters are already available, pass them via dlf:
 distLquantile(dlf=dlf, probs=0:5/5, selection=c("wak","gev","kap"))

distLquantile(dlf=dlf, truncate=0.3, list=TRUE)$truncate

# censored (truncated, trimmed) quantile, Peak Over Treshold (POT) method:
qwak <- distLquantile(annMax, sel="wak", prob=0.95, emp=FALSE, list=TRUE)
plotLquantile(qwak, ylim=c(0,0.06) ); qwak$quant
qwak2 <-distLquantile(annMax, sel="wak", prob=0.95, emp=FALSE, list=TRUE, truncate=0.6)
plotLquantile(qwak2, add=TRUE, distcols="blue")

# Simulation of truncation effect
library(lmomco)
#set.seed(42)
rnum <- rlmomco(n=1e3, para=dlf$parameter$gev)
myprobs <- c(0.9, 0.95, 0.99, 0.999)
mytrunc <- seq(0, 0.9, length.out=20)
trunceffect <- sapply(mytrunc, function(mt) distLquantile(rnum, selection="gev",
    probs=myprobs, truncate=mt, quiet=TRUE,
    pemprirical=FALSE)['gev',])

# If more values are truncated, the function runs faster
op <- par(mfrow=c(2,1), mar=c(2,4.5,2,0.5), cex.main=1)
dlf1 <- distLquantile(rnum, sel="gev", probs=myprobs, emp=FALSE, list=TRUE)
dlf2 <- distLquantile(rnum, sel="gev", probs=myprobs, emp=FALSE, list=TRUE, truncate=0.3)
plotLquantile(dlf1, ylab="", xlab="")
plotLquantile(dlf2, add=TRUE, distcols=4)
legend("right", c("fitted GEV", "fitted with truncate=0.3"), lty=1, col=c(2,4), bg="white")
par(mar=c(3,4.5,3,0.5))
plot(mytrunc, trunceffect[1,], ylim=range(trunceffect), las=1, type="l",
    xlab="Proportion censored", ylab="parametric quantile")
title(xlab="Proportion censored", mgp=c(1.8,1,0))
for(i in 2:4) lines(mytrunc, trunceffect[1,], xlab="", ylab="")
library("berryFunctions")
textField(rep(0.5,4), trunceffect[,11], paste0("Q",myprobs*100,"%")
par(op)

trunc <- seq(0.0,0.1,len=200)
 dd <- pbapply(trunc, function(t) distLquantile(annMax,
    selection="gpa", weight=FALSE, truncate=t, prob=0.99, quiet=T)[c(1,3),])
 plot(trunc, dd[,1], type="o", las=1)
lines(trunc, dd[2,], type="o", col=2)
\end{verbatim}
distLweights

Compute distribution weights from GOF

Description

Determine distribution function weights from RMSE for weighted averages. The weights are inverse to RMSE: weight1 for all dists, weight2 places zero weight on the worst fitting function, weight3 on the worst half of functions.

Usage

distLweights(
  RMSE,
  order = TRUE,
  onlydn = TRUE,
  weightc = NA,
  quiet = FALSE,
  ...
)

Arguments

RMSE Numeric: Named vector with goodness of fit values (RMSE). Can also be a data.frame, in which case the column rmse or RMSE is used.

order Logical: should result be ordered by RMSE? If order=FALSE, the order of appearance in RMSE is kept (alphabetic or selection in distLfit). DEFAULT: TRUE

onlydn Logical: weight only distributions from lmomco::dist.list? DEFAULT: TRUE (all other RMSEs are set to 0)
distLweights

weightc  Optional: a named vector with custom weights for each distribution. Are internally normalized to sum=1 after removing nonfitted dists. Names match the parameter names from RMSE. DEFAULT: NA

quiet  Logical: Suppress messages. DEFAULT: FALSE

...  Ignored arguments (so a set of arguments can be passed to distLfit and distLquantile and arguments used only in the latter will not throw errors)

Value
data.frame

Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Dec 2016

See Also
distLfit, distLquantile

Examples

# weights from RMSE vector:
RMSE <- c(gum=0.20, wak=0.17, gam=0.21, gev=0.15)
distLweights(RMSE)
distLweights(RMSE, order=FALSE)

# weights from RMSE in data.frame:
df <- data.frame("99.9%"=2:5, RMSE=sample(3:6))
rownames(df) <- letters[1:4]
df ; distLweights(df, onlydn=FALSE)

# custom weights:
set.seed(42); x <- data.frame(A=1:5, RMSE=runif(5)); x
distLweights(x) # two warnings
distLweights(x, weightc=c("1"=3, "3"=5), onlydn=FALSE)
distLweights(x, weightc=c("1"=3, "3"=5), order=FALSE, onlydn=FALSE)

# real life example:
data(annMax)
cw <- c("gpa"=7, "gev"=3, "wak"=6, "wei"=4, "kap"=3.5, "gum"=3, "ray"=2.1, "ln3"=2, "pe3"=2.5, "gno"=4, "gam"=5)
dlf <- distLfit(annMax, weightc=cw, quiet=TRUE, order=FALSE)
plotLweights(dlf)

# GOF judgement by RMSE, not R2 -------
# Both RMSE and R2 are computed with ECDF and TCDF
# R2 may be very good (see below), but fit needs to be close to 1:1 line,
# which is better measured by RMSE
dlfit <- distLfit(annMax, ks=TRUE)
extremeStat

Extreme value statistics on a linear scale

Description

Fit (via L moments), plot (on a linear scale) and compare (by goodness of fit) several (extreme value) distributions. Compute high quantiles even in small samples and estimate extrema at given return periods.

Open the Vignette for an introduction to the package: vignette("extremeStat")

This package heavily relies on and thankfully acknowledges the package lmomco by WH Asquith.

Package overview

The main functions in the extremeStat package are:

```r
op <- par(mfrow=c(1,2), mar=c(3,4,0.5,0.5), mgp=c(1.9,0.7,0))
yy <- nrow(dlf$gof):1 # depends on length of lmomco::dist.list()
plot(dlf$gof$RMSE, yy, yaxt="n", ylab="", type="o"); axis(2, yy, rownames(dlf$gof), las=1)
plot(dlf$gof$R2, yy, yaxt="n", ylab="", type="o"); axis(2, yy, rownames(dlf$gof), las=1)
par(op)

sel <- c("wak", "lap", "nor", "revgum")
plotLfit(dlf, selection=sel, cdf=TRUE)
dlf$gof[sel,-(2:7)]

x <- sort(annMax, decreasing=TRUE)
ECDF <- ecdf(x)(x)
TCDF <- sapply(sel, function(d) lmomco::plmomco(x, dlf$parameter[[d]]))

plot(TCDF[,"lap"], ECDF, col="cyan", asp=1, las=1)
points(TCDF[,"nor"], ECDF, col="green")
points(TCDF[,"revgum"], ECDF, col="red")
abline(a=0, b=1, lwd=3, lty=3)
legend("bottomright", c("lap good RMSE bad R2", "nor bad RMSE good R2"),
col=c("cyan", "green"), lwd=2)

# more distinct example (but with fake data)
set.seed(42); x <- runif(30)
y1 <- x+rnorm(30, sd=0.09)
y2 <- 1.5*x+rnorm(30, sd=0.01)-0.3
plot(x,x, asp=1, las=1, main="High cor (R2) does not necessarily mean good fit!")
berryFunctions::linReg(x, y2, add=TRUE, digits=4, pos=1="topleft")
points(x,y2, col=4, pch=3)
points(x,y1, col=2)
berryFunctions::linReg(x, y1, add=TRUE, digits=4, col=2)
abline(a=0, b=1, lwd=3, lty=3)
```
They create and modify a list object printed by (and documented in) `printL`.

**Author(s)**

Berry Boessenkool, <berry-b@gmx.de>, 2014-2016

**See Also**

If you are looking for more detailed (uncertainty) analysis, eg confidence intervals, check out the package `extRemes`, especially the function `fevd`. https://cran.r-project.org/package=extRemes
Intro slides: https://sites.lsa.umich.edu/eva2015/wp-content/uploads/sites/44/2015/06/IntroEVT.pdf
Parameter fitting and distribution functions: https://cran.r-project.org/package=lmomco
R in Hydrology: https://abouthydrology.blogspot.de/2012/08/r-resources-for-hydrologists.html

**Examples**

```r
data(annMax) # annual discharge maxima from a stream in Austria
plot(annMax, type="l")
dle <- distLextreme(annMax)
dle$returnlev
```

---

**plotLexBoot**  
*Bootstrapping uncertainty intervals for return periods*

**Description**

plot bootstrap uncertainty intervals for `plotLextreme`.

**Usage**

```r
plotLexBoot(dlf, selection = NULL, add = FALSE, log = TRUE, ...)
```
Arguments

dlf      dlf object, as returned by distLexBoot
selection Character vector with distribution function names to be used. Suggested to keep this low. DEFAULT: NULL
add      Add to existing plot? DEFAULT: FALSE
log      Plot on a logarithmic axis. DEFAULT: TRUE
...      Further arguments passed to plotLextreme. If add=TRUE, they are instead passed to berryFunctions::ciBand

Value

invisible dlf object, see printL

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2016

See Also

distLexBoot

Examples

# see distLexBoot

plotLextreme

Plot extreme value statistics

Description

Plots distributions fitted by L-moments and adds plotting positions by Weibull and Gringorton. This is an auxiliary graphing function to distLexExtreme

Usage

plotLextreme(
  dlf,
  selection = NULL,
  order = FALSE,
  add = FALSE,
  nbest = 5,
  log = "",
  xlim = NULL,
  ylim = NULL,
  las = 1,
Arguments

dlf
 selection
 order
 add
 nbest
 log
 xlim
 ylim
 las
 main
 xlab
 ylab
 PPcol
 PPPch
 PPcex

List as returned by distLextreme or distLexBoot
Selection of distributions. Character vector with type as in lmom2par. DEFAULT: NULL
If selection is given, should legend and colors be ordered by gof anyways? DEFAULT: FALSE
If TRUE, plot is not called before adding lines. This lets you add lines to an existing plot. DEFAULT: FALSE
Number of distributions plotted, in order of goodness of fit. Overwritten internally if selection is given. DEFAULT: 5
Charstring ("x", "y", "xy") for logarithmic axes. See logargs. DEFAULT: ""
X-axis limits. DEFAULT: ylim of plotting positions
Y-lim. DEFAULT: from min to extended max
LabelAxisStyle to orient labels, see par. DEFAULT: 1
Title of plot. DEFAULT: dlf$datname
X axis label. DEFAULT: "Return Period RP [a]"
Y axis label. Please note that the ubuntu pdf viewer might be unable to display unicode superscript. DEFAULT: "Discharge HQ [m\(^3\)/s]"
Plotting Position point colors, vector of length two for Weibull and Gringorton, recycled. PP are not used for fitting distributions, but for plotting only. DEFAULT: "black"
point characters for plotting positions after Weibull and Gringorton, respectively. NA to suppress in plot and legend. DEFAULT: c(16,3)
Character EXpansion of plotting points. DEFAULT: 1
distcols  Color for each distribution added with lines. Recycled, if necessary. DEFAULT: rainbow
lty     Line TYPE for plotted distributions. Is recycled to from a vector of length nbest, i.e. a value for each dist. DEFAULT: 1
lwd     Line Width of distribution lines. Recycled vector of length nbest. DEFAULT: 1
pch     Point CHARACTER of points added at regular intervals. This makes lines more distinguishable from each other. NA to suppress. Recycled vector of length nbest. DEFAULT: NA
cex     if pch != NA, size of points. Recycled vector of length nbest. DEFAULT: 1
n_pch   Number of points spread evenly along the line. Recycled vector of length nbest. DEFAULT: 15
legend  Logical. Add a legend? DEFAULT: TRUE
rmse    Integer. If rmse > 0, RMSE values are added to legend. They are rounded to rmse digits. DEFAULT: 4
legargs list of arguments passed to legend except for legend, col, pch, lwd, lty. DEFAULT: NULL
quiet   Suppress notes? DEFAULT: FALSE
logargs list of arguments passed to berryFunctions::logAxis.
...    Further arguments passed to plot like yaxt="n", ...

Value

invisible dlf object, see printL

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, March 2015, updated heavily Aug 2015

See Also

distLextreme, plotLfit

Examples

#see
?distLextreme
plotLfit

Plot distributions fitted with L-moments

Description

Plot histogram and distribution densities or ecdf with cumulated probability

Usage

plotLfit(
  dlf,
  nbest = 5,
  selection = NULL,
  order = TRUE,
  rmse = 4,
  cdf = FALSE,
  log = FALSE,
  supportends = TRUE,
  breaks = 20,
  xlim = extendrange(dlf$dat, f = 0.15),
  ylim = NULL,
  col = "grey",
  main = paste(if (cdf) "Cumulated", "density distributions of", dlf$datname),
  xlab = dlf$datname,
  ylab = if (cdf) "(Empirical) Cumulated Density (CDF)" else "Probability Density Function (PDF)",
  las = 1,
  distcols = berryFunctions::rainbow2(nbest),
  lty = 1,
  add = FALSE,
  logargs = NULL,
  legend = TRUE,
  legargs = NULL,
  histargs = NULL,
  ...
)

Arguments

dlf List as returned by distLfit, containing the elements dat, parameter, gof, datname

nbest Number of distributions plotted, in order of goodness of fit. DEFAULT: 5

selection Names of distributions in dlf$parameter that will be drawn. Overrides nbest. DEFAULT: NULL

order Logical: order legend and colors by RMSE, even if dlf$gof is unordered or selection is given? DEFAULT: TRUE
\texttt{rmse} Integers. If \texttt{rmse} $\neq 0$, RMSE values are added to legend. They are rounded to \texttt{rmse} digits. DEFAULT: 4

\texttt{cdf} If TRUE, plot cumulated DF instead of probability density. DEFAULT: FALSE

\texttt{log} If TRUE, logAxis is called. Only makes sense if \texttt{dlf$dat} is already logarithmic and ranges eg. from -2 to 3. DEFAULT: FALSE

\texttt{supportends} If TRUE, dots are placed at the support bounds. DEFAULT: TRUE

\texttt{breaks} \texttt{hist} breaks. DEFAULT: 20

\texttt{xlim, ylim} \texttt{hist} or \texttt{ecdf} axis limits.

\texttt{col} \texttt{hist} bar color or \texttt{ecdf} point color. DEFAULT: "grey"

\texttt{main, xlab, ylab} \texttt{hist} or \texttt{ecdf} main, xlab, ylab. DEFAULT: abstractions from \texttt{dlf$dat$name}

\texttt{las} Label Axis Style for orientation of numbers along axes. DEFAULT: 1

\texttt{distcols} Color for each distribution added with \texttt{lines}. DEFAULT: \texttt{rainbow2}

\texttt{lty} Line TYPE for plotted distributions. Recycled vector of length \texttt{nbest}. DEFAULT: 1

\texttt{add} If TRUE, \texttt{hist/ecdf} is not called before adding lines. This lets you add lines highly customized one by one. DEFAULT: FALSE

\texttt{logargs} List of arguments passed to \texttt{logAxis} if \texttt{log=TRUE}. DEFAULT: NULL

\texttt{legend} Should \texttt{legend} be called? DEFAULT: TRUE

\texttt{legargs} List of arguments passed to \texttt{legend} except for legend and \texttt{col}. DEFAULT: NULL

\texttt{histargs} List of arguments passed to \texttt{hist} or \texttt{ecdf} except for \texttt{x, freq}. DEFAULT: NULL

\texttt{...} Further arguments passed to \texttt{lines}, like \texttt{type}, \texttt{pch}, ...

**Details**

By default, this plots density instead of CDF, because the distributions are easier to discern and tail behavior is easier to judge visually. See also [https://www.vosesoftware.com/ModelRiskHelp/index.htm#Presenting_results/Cumulative_plots/Relationship_between_cdf_and_density_(histogram)_plots.htm](https://www.vosesoftware.com/ModelRiskHelp/index.htm#Presenting_results/Cumulative_plots/Relationship_between_cdf_and_density_(histogram)_plots.htm)

**Value**

invisible \texttt{dlf} object, see \texttt{printL}

**Author(s)**

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

**See Also**

\texttt{distLfit, plotLquantile}

**Examples**

# See \texttt{distLfit}
plotLquantile

Plot quantiles of distributions fitted with L-moments

Description

Plot quantiles of distributions fitted with L-moments

Usage

plotLquantile(
  dlf,
  nbest = 5,
  selection = NULL,
  order = FALSE,
  rows = NULL,
  heights = stats::quantile(par("usr")[3:4], 0.2),
  distcols = dlfplot$distcols,
  linargs = NULL,
  ...
)

Arguments

dlf List as returned by distLquantile, containing the elements dat, parameter, gof, datname, quant
nbest, selection, order Distributions to be plotted, see plotLfit
rows Rowname(s) of dlf$quant that should be drawn instead of the selection / nbest highest ranking distribution functions. 'GPD*' will select all the gpd fits. heights and distcols must then accordingly have at least 13 elements (or will be recycled). DEFAULT: NULL
heights Coordinates of quantile line ends, recycled if necessary. DEFAULT: 20% of plot height.
distcols Color for each distribution added with lines. DEFAULT: dlfplot$distcols
linargs Arguments passed to lines. DEFAULT: NULL
...
Further arguments passed to plotLfit

Value

invisible dlf object, see printL

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Dec 2016
plotLweights

Description

Plot rank comparison of fitted distributions calculated by distLfit.

Usage

plotLweights(
  dlf,
  type = "o",
  col = RColorBrewer::brewer.pal(5, "Set2"),
  pch = c(1:4, NA),
  lty = 1,
  lwd = 1,
  legargs = NULL,
  main = "Distribution function GOF and weights",
  xlab = "Weight / RMSE",
  ylab = "",
  xlim = range(gof[, grep("weight", colnames(gof))], na.rm = TRUE),
  ...
)

Arguments

dlf List as returned by distLfit, containing the element gof
type, col, pch, lty, lwd
  Vectors with 5 values for line customization. Recycled if necessary.
legargs List of arguments passed to legend, like cex, bg, etc.
main, xlab, ylab
  plot title and axis labels
xlim Range of x axis. DEFAULT: range(gof$weight*)
...
  Further arguments passed to plot.

Value

None.

See Also
distLquantile, plotLfit

Examples

# See distLquantile
Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Sept 2014

See Also
distLweights, distLfit

Examples

# see distLweights and distLfit

printL
print dlf objects

Description

print list objects created in this package

Usage

printL(dlf, digits = 1)

Arguments

dlf List as explained in section Details
digits number of digits rounded to. DEFAULT: 1

Details

The common object to share between functions (see overview in extremeStat) is a list with the following elements:

dat numeric vector with (extreme) values, with all NAs and values below threshold removed
dat_full original input data complete with NAs
datname character string for main, xlab etc
parameter list (usually of length 17 if speed=TRUE in distLfit) with parameters of each distribution
gof dataframe with 'Goodness of Fit' measures, sorted by RMSE of theoretical and empirical cumulated density
distnames character vector with selected distribution names
distfailed Names of nonfitted distributions or ""
distcols colors for distnames (for plotting). If not given manually, determined by berryFunctions::rainbow2
distselector character string with function name creating the selection
truncate, threshold Truncation percentage and threshold value, relevant for distLquantile

optionally, it can also contain:
quantGPD

Fast GPD quantile estimate

Quantile estimates from distLquantile objects from distLexBoot

Value

none, prints via message.

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Sept 2014, March + July 2015, Dec 2016

See Also

extremeStat

Examples

# see
?distLexreme

Usage

quantGPD(
  x,
  probs = c(0.8, 0.9, 0.99),
  truncate = 0,
  threshold = berryFunctions::quantileMean(x, truncate),
  addn = TRUE,
  quiet = FALSE,
  ...
)

Description

Fast GPD quantile estimate through L-moments
Arguments

- **x**: Vector with numeric values. NAs are silently ignored.
- **probs**: Probabilities. DEFAULT: c(0.8, 0.9, 0.99)
- **truncate, threshold**: Truncation proportion or threshold. DEFAULT: 0, computed See `q_gpd`.
- **addn**: Logical: add element with sample size (after truncation). DEFAULT: TRUE
- **quiet**: Should messages from this function be suppressed? DEFAULT: FALSE
- **...**: Further arguments passed to `lmomco::pargpa`

Value

Vector with quantiles

Author(s)

Berry Boessenkool, <berry-b@gmx.de>, Jun 2017

See Also

- `q_gpd` for a comparison across R packages and methods, `distLquantile` to compare distributions

Examples

```r
data(annMax)
quantile(annMax, 0.99)
quantGPD(annMax, 0.99)
## Not run: # Excluded from CRAN checks to reduce checking time
data(rain, package="ismev"); rain <- rain[rain>0]
tr <- seq(0, 0.999, len=50)
qu <- pbapply::pbsapply(tr, quantGPD, x=rain, probs=c(0.9, 0.99, 0.999)) # 30 s
plot(tr, qu[3,], ylim=range(rain), las=1, type="l")
lines(tr, qu[2,], col=2); lines(tr, qu[1,], col=4)

tr <- seq(0.88, 0.999, len=50)
qu <- pbapply::pbsapply(tr, quantGPD, x=rain, probs=c(0.9, 0.99, 0.999)) # 5 s
plot(tr, qu[3,], ylim=range(rain), las=1, type="l")
lines(tr, qu[2,], col=2); lines(tr, qu[1,], col=4);
tail(qu["n",])

library(microbenchmark)
data(rain, package="ismev"); rain <- rain[rain>0]
mb <- microbenchmark(quantGPD(rain[1:200], truncate=0.8, probs=0.99, addn=F),
distLquantile(rain[1:200], sel="gpa", emp=F, truncate=0.8, quiet=T, probs=0.99)[1,1])
boxplot(mb)
# since computing the lmoments takes most of the computational time,
# there's not much to optimize in large samples like n=2000
```
## End(Not run)

### Description
Compute quantile of General Pareto Distribution fitted to sample by peak over threshold (POT) method using threshold from truncation proportion, comparing several R packages doing this.

### Usage

```r
q_gpd(
  x,
  probs = c(0.8, 0.9, 0.99),
  truncate = 0,
  threshold = berryFunctions::quantileMean(x, truncate),
  package = "extRemes",
  method = NULL,
  list = FALSE,
  undertruncNA = TRUE,
  quiet = FALSE,
  ttquiet = quiet,
  efquiet = quiet,
  ...
)
```

### Arguments

- `x` Vector with numeric values. NAs are silently ignored.
- `probs` Probabilities of truncated (Peak over threshold) quantile. DEFAULT: c(0.8, 0.9, 0.99)
- `truncate` Truncation percentage (proportion of sample discarded). DEFAULT: 0
- `threshold` POT cutoff value. If you want correct percentiles, set this only via truncate, see Details. DEFAULT: `quantileMean(x, truncate)`
- `package` Character string naming package to be used. One of c("lmomco","evir","evd","extRemes","fExtremes","ismev"). DEFAULT: "extRemes"
- `method` method passed to the fitting function, if applicable. Defaults are internally specified (See Details), depending on package, if left to the DEFAULT: NULL.
- `list` Return result from the fitting function with the quantiles added to the list as element `quant` and some information in elements starting with `q_gpd_`. DEFAULT: FALSE
- `undertruncNA` Return NAs for probs below truncate? Highly recommended to leave this at the DEFAULT: TRUE

---

**q_gpd**

**GPD quantile of sample**

### Description
Compute quantile of General Pareto Distribution fitted to sample by peak over threshold (POT) method using threshold from truncation proportion, comparing several R packages doing this.

### Usage

```r
q_gpd(
  x,
  probs = c(0.8, 0.9, 0.99),
  truncate = 0,
  threshold = berryFunctions::quantileMean(x, truncate),
  package = "extRemes",
  method = NULL,
  list = FALSE,
  undertruncNA = TRUE,
  quiet = FALSE,
  ttquiet = quiet,
  efquiet = quiet,
  ...
)
```

### Arguments

- `x` Vector with numeric values. NAs are silently ignored.
- `probs` Probabilities of truncated (Peak over threshold) quantile. DEFAULT: c(0.8, 0.9, 0.99)
- `truncate` Truncation percentage (proportion of sample discarded). DEFAULT: 0
- `threshold` POT cutoff value. If you want correct percentiles, set this only via truncate, see Details. DEFAULT: `quantileMean(x, truncate)`
- `package` Character string naming package to be used. One of c("lmomco","evir","evd","extRemes","fExtremes","ismev"). DEFAULT: "extRemes"
- `method` method passed to the fitting function, if applicable. Defaults are internally specified (See Details), depending on package, if left to the DEFAULT: NULL.
- `list` Return result from the fitting function with the quantiles added to the list as element `quant` and some information in elements starting with `q_gpd_`. DEFAULT: FALSE
- `undertruncNA` Return NAs for probs below truncate? Highly recommended to leave this at the DEFAULT: TRUE
quiet Should messages from this function be suppressed? DEFAULT: FALSE

ttquiet Should truncation!=threshold messages from this function be suppressed? DEFAULT: quiet

efquiet Should warnings in function calls to the external packages be suppressed via options(warn=-1)? The usual type of warning is: NAs produced in log(...). DEFAULT: quiet

Further arguments passed to the fitting function listed in section Details.

Details

Depending on the value of "package", this fits the GPD using

- lmomco::pargpa
- evir::gpd
- evd::fpot
- extRemes::fevd
- fExtremes::gpdFit
- ismev::gpd.fit
- Renext::Renouv or Renext::fGPD

The method defaults (and other possibilities) are

- lmomco: none, only L-moments
- evir: "pwm" (probability-weighted moments), or "ml" (maximum likelihood)
- evd: none, only Maximum-likelihood fitting implemented
- extRemes: "MLE", or "GMLE", "Bayesian", "Lmoments"
- fExtremes: "pwm", or "mle"
- ismev: none, only Maximum-likelihood fitting implemented
- Renext: "r" for Renouv (since distname.y = "gpd", evd::fpot is used), or 'f' for fGPD (with minimum POTs added)

The Quantiles are always given with probs in regard to the full (uncensored) sample. If e.g. truncate is 0.90, the distribution function is fitted to the top 10% of the sample. The 95th percentile of the full sample is equivalent to the 50% quantile of the subsample actually used for fitting. For computation, the probabilities are internally updated with \( p2=(p-t)/(1-t) \) but labeled with the original \( p \). If you truncate 90% of the sample, you cannot compute the 70th percentile anymore, thus undertruncNA should be left to TRUE.

If not exported by the packages, the quantile functions are extracted from their source code (Nov 2016).

Value

Named vector of quantile estimates for each value of probs,

or if(list): list with element q_gpd_quant and info-elements added. q_gpd_n_geq is number of values greater than or equal to q_gpd_threshold. gt is only greater than.
Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Feb 2016

References

See Also

distLquantile which compares results for all packages
Other related packages (not implemented):
https://cran.r-project.org/package=gPdtest
https://cran.r-project.org/package=actuar
https://cran.r-project.org/package=fitdistrplus
https://cran.r-project.org/package=lmom

Examples

data(annMax)
q_gpd(annMax)
q_gpd(annMax, truncate=0.6)
q_gpd(annMax, truncate=0.85)
q_gpd(annMax, truncate=0.91)
q_gpd(annMax, package="evir")
q_gpd(annMax, package="evir", method="ml")
q_gpd(annMax, package="evd")
q_gpd(annMax, package="extRemes")
q_gpd(annMax, package="extRemes", method="GMLE")  # computes a while
q_gpd(annMax, package="extRemes", method="Lmoments")
q_gpd(annMax, package="extRemes", method="nonsense")  # NAs
q_gpd(annMax, package="fExtremes")  # log warnings
q_gpd(annMax, package="fExtremes", efquiet=TRUE)  # silenced warnings
q_gpd(annMax, package="fExtremes", method="mle")
q_gpd(annMax, package="ismev")
q_gpd(annMax, package="Renext")
q_gpd(annMax, package="Renext", method="f")
berryFunctions::is.error(q_gpd(annMax, package="nonsense"), force=TRUE)

# compare all at once with
d <- distLquantile(annMax); d
# d <- distLquantile(annMax, speed=FALSE); d # for Bayesian also
q_gpd(annMax, truncate=0.85, package="evd")  # Note about quantiles
q_gpd(annMax, truncate=0.85, package="evir")
q_gpd(annMax, truncate=0.85, package="evir", quiet=TRUE)  # No note
q_gpd(annMax, truncate=0.85, package="evir", undertruncNA=FALSE)
q_gpd(annMax, truncate=0.85, package="evir", list=TRUE)
```r
str( q_gpd(annMax, truncate=0.85, probs=0.6, package="evir", list=TRUE) ) # NAs
str( q_gpd(annMax, package="evir", list=TRUE) )
str( q_gpd(annMax, package="evd", list=TRUE) )
str( q_gpd(annMax, package="extRemes", list=TRUE) )
str( q_gpd(annMax, package="fExtremes", list=TRUE) )
str( q_gpd(annMax, package="ismev", list=TRUE) )
str( q_gpd(annMax, package="Renext", list=TRUE) )

q_gpd(annMax, package="evir", truncate=0.9, method="ml") # NAs (MLE fails often)

trunc <- seq(0,0.9,len=500)
library("pbapply")
quant <- pbsapply(trunc, function(tr) q_gpd(annMax, pack="evir", method = "pwm",
  truncate=tr, quiet=TRUE))
quant <- pbsapply(trunc, function(tr) q_gpd(annMax, pack="lmomco", truncate=tr, quiet=TRUE))
plot(trunc, quant[,"99%"], type="l", ylim=c(80,130), las=1)
lines(trunc, quant[,"90%"],
lines(trunc, quant[,"80%"],
plot(trunc, quant[,"RMSE"], type="l", las=1)

## Not run:
## Not run in checks because simulation takes too long

trunc <- seq(0,0.9,len=200)
dlfs <- pblapply(trunc, function(tr) distLfit(annMax, truncate=tr, quiet=TRUE, order=FALSE))
rmses <- sapply(dlfs, function(x) x$gof$RMSE)
plot(trunc, rmses[1,], ylim=c(20,135), nbest=1)
cols <- rainbow2(17)[rank(rmses[,1])]
for(i in 1:17) lines(trunc, rmses[i,], col=cols[i])
dlfs2 <- lapply(0:8/10, function(tr) distLfit(annMax, truncate=tr, quiet=TRUE))
pdf("dummy.pdf")
dummy <- sapply(dlfs2, function(x)
  plotLfit(x, cdf=TRUE, main=x$truncate, ylim=0:1, xlim=c(20,135), nbest=1)
  title(sub=round(x$gof$RMSE[1],4))
)
dev.off()

# truncation effect
mytruncs <- seq(0, 0.9, len=150)
oo <- options(show.error.messages=FALSE, warn=-1)
myquants <- sapply(mytruncs, function(t) q_gpd(annMax, truncate=t, quiet=TRUE))
options(o)
plot(1, type="n", ylim=range(myquants, na.rm=TRUE), xlim=c(0,0.9), las=1,
  xlab="truncated proportion", ylab="estimated quantiles")
abline(h=quantileMean(annMax, probs=c(0.8,0.9,0.99)))
for(i in 1:3) lines(mytruncs, myquants[i,], col=i)
text(0.3, c(87,97,116), rownames(myquants), col=1:3)

# Underestimation in small samples
# create known population:
dat <- extRemes::revd(1e5, scale=50, shape=-0.02, threshold=30, type="GP")
```
op <- par(mfrow=c(1,2), mar=c(2,2,1,1))
hist(dat, breaks=50, col="tan")
berryFunctions::logHist(dat, breaks=50, col="tan")
par(op)

# function to estimate empirical and GPD quantiles from subsamples
samsizeeffect <- function(n, nrep=30, probs=0.999, trunc=0.5, Q=c(0.4,0.5,0.6))
{
  res <- replicate(nrep, {
    subsample <- sample(dat, n)
    qGPD <- q_gpd(subsample, probs=probs, truncate=trunc)
    qEMP <- berryFunctions::quantileMean(subsample, probs=probs, truncate=trunc)
    c(qGPD=qGPD, qEMP=qEMP))
  })
apply(res, MARGIN=1, berryFunctions::quantileMean, probs=Q)

# Run and plot simulations
samplesize <- c(seq(20, 150, 10), seq(200,800, 100))
results <- pbapply::pblapply(samplesize, samsizeeffect)
res <- function(row, col) sapply(results, function(x) x[row,col])
berryFunctions::ciBand(yu=res(3,1),yl=res(1,1),ym=res(2,1),x=samplesize,
  main="99.9% Quantile underestimation", xlab="subsample size", ylim=c(200,400), colm=4)
berryFunctions::ciBand(yu=res(3,2),yl=res(1,2),ym=res(2,2),x=samplesize, add=TRUE)
abline(h=berryFunctions::quantileMean(dat, probs=0.999))
text(300, 360, "empirical quantile of full sample")
text(300, 340, "GPD parametric estimate", col=4)
text(300, 300, "empirical quantile estimate", col="green3")

## End(Not run) # end of dontrun

---

q_weighted

**q_weighted**

Compute weighted averages of quantile estimates

**Description**

Compute weighted averages of quantile estimates

**Usage**

```r
q_weighted(quant, weights = distLweights(quant, ...), onlyc = FALSE, ...)
```

**Arguments**

- `quant` : Data.frame as in `distLquantile` output.
- `weights` : Data.frame as in `distLweights` output.
- `onlyc` : Logical: only return custom weighted quantile estimates as a vector? Useful to add those to existing results. See examples. DEFAULT: FALSE
Arguments passed to `distLweights` like weightc, onlydn=FALSE. order will be ignored, as `q_weighted` only adds/changes the rows weighted*.

Value
data.frame with rows "weighted*" added.

Author(s)
Berry Boessenkool, <berry-b@gmx.de>, Dec 2016

See Also
distLquantile

Examples

```r
x <- data.frame(A=1:5, RMSE=runif(5))
distLweights(x, onlydn=FALSE)

q_weighted(x, onlydn=FALSE)
q_weighted(x, distLweights(x, weightc=c("1"=3, "3"=5), order=FALSE, onlydn=FALSE) )

## Not run: # time consuming
x <- rexp(190)
d <- distLquantile(x)
d2 <- q_weighted(d)
stopifnot(all(d==d2, na.rm=TRUE))

# fast option for adding custom weighted estimates:
cw <- runif(17)
dw <- distLweights(d, weightc=cw)
qw1 <- q_weighted(d, weightc=cw); qw1
qw2 <- q_weighted(d, weights=dw); qw2
stopifnot(all(qw1==qw2, na.rm=TRUE))
q_weighted(d, weights=dw, onlyc=TRUE)
q_weighted(d, weights=data.frame(weightc=cw), onlyc=TRUE)

system.time(pbreplicate(5000, q_weighted(d, weightc=cw))) # 8.5 secs
system.time(pbreplicate(5000, q_weighted(d, weights=dw, onlyc=TRUE))) # 0.8 secs

## End(Not run)
```
**weightp**

*distribution weights*

**Description**


**Format**

named num [1:17]

**Source**

See paper revisions (not yet online at moment of extremeStat update) (<berry-b@gmx.de>)

**Examples**

```r
data(weightp)
data.frame(weightp)
barplot(weightp, horiz=TRUE, las=1)
stopifnot(all.equal(sum(weightp), 1))

data(annMax); data(weightp)
dlf <- distLfit(annMax, weightc=weightp)
dlf$gof
quant <- distLquantile(annMax, weightc=weightp)
quant
```
Index

∗ bootstrap
  distLexBoot, 3
  plotLexBoot, 19
∗ datasets
  annMax, 2
  weightp, 36
∗ distribution
  distLexBoot, 3
  distLextreme, 4
  distLfit, 9
  distLquantile, 12
  distLweights, 16
  plotLexBoot, 19
  plotLextreme, 20
  plotLfit, 23
  plotLquantile, 25
  plotLweights, 26
  q_gpd, 30
  q_weighted, 34
  quantGPD, 28
∗ documentation
  extremeStat, 18
∗ dplot
  distLexBoot, 3
  distLextreme, 4
  distLfit, 9
  plotLexBoot, 19
  plotLextreme, 20
∗ hplot
  distLexBoot, 3
  distLextreme, 4
  distLfit, 9
  plotLexBoot, 19
  plotLextreme, 20
  plotLfit, 23
  plotLquantile, 25
  plotLweights, 26
∗ list
  printL, 27

∗ methods
  printL, 27
∗ montecarlo
  distLexBoot, 3
  plotLexBoot, 19
∗ package
  extremeStat, 18
∗ print
  printL, 27
∗ robust
  distLquantile, 12
  q_gpd, 30
  quantGPD, 28
∗ ts
  distLexBoot, 3
  distLextreme, 4
  plotLexBoot, 19
∗ univar
  distLfit, 9
  distLquantile, 12
  q_gpd, 30
  quantGPD, 28

annMax, 2

ciBand, 20

dist.list, 10, 13, 16
distLexBoot, 3, 6, 19–21, 28
distLextreme, 3, 4, 4, 19–22, 28
distLfit, 5, 6, 9, 13, 14, 16, 17, 19, 23, 24, 26, 27
distLquantile, 3, 5, 10, 12, 17, 19, 25–29, 32, 34, 35
distLweights, 10, 13, 16, 19, 27, 34, 35
dcdf, 24

extremeStat, 13, 18, 27, 28
extremeStat-package (extremeStat), 18

fevd, 6, 10, 19, 31
fGPD, 31
fitdistr, 10
fpot, 31

gpd, 31
gpd.fit, 31
gpdFit, 31

hist, 24
legend, 22, 24, 26
lines, 22, 24, 25
lmom2par, 9, 10, 21
logAxis, 22, 24

message, 10, 28

options, 31

par, 21
pargpa, 29, 31
plot, 22, 26
plotLexBoot, 4, 19
plotLextreme, 3, 5, 19, 20, 20, 28
plotLfit, 10, 19, 22, 23, 25, 26
plotLquantile, 19, 24, 25
plotLweights, 10, 19, 26

printL, 4, 5, 10, 13, 19, 20, 22, 24, 25, 27

q_gpd, 10, 12–14, 19, 29, 30
q_weighted, 19, 34
quantGPD, 28
quantile, 13
quantileMean, 3, 10, 12, 13, 30

rainbow2, 22, 24, 27
Renouv, 31
round, 27

sample, 3

weightp, 36