R documentation

of all in 'man'

July 6, 2013

R topics documented:

`evmix-package` *Extreme Value Mixture Modelling and Threshold Estimation*

Description

Functions for univariate extreme value mixture modelling, threshold estimation and uncertainty quantification

Details

Package:	evmix
Type:	Package
Version:	0.1-0
Date:	2013-05-06
License:	GPL-3
LazyLoad:	yes

The usual distribution functions, maximum likelihood inference and model diagnostics for univariate stationary extreme value mixture models are provided.

Kernel density estimation including various boundary corrected kernel density estimation methods, with cross-validation likelihood based bandwidth estimators are included.

Reasonable consistency with the base functions in the `evd` package is provided, so that users can safely interchange most code.

Author(s)

Yang Hu and Carl Scarrott, University of Canterbury, New Zealand <carl.scarrott@canterbury.ac.nz>

References

http://www.math.canterbury.ac.nz/~c.scarrott/evmix

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Hu, Y. (2013). Extreme value mixture modelling: An R package and simulation study. MSc (Hons) thesis, University of Canterbury, New Zealand. http://ir.canterbury.ac.nz/simple-search?query=extreme&submit=Go

MacDonald, A. (2012). Extreme value mixture modelling with medical and industrial applications. PhD thesis, University of Canterbury, New Zealand. http://ir.canterbury.ac.nz/bitstream/10092/6679/1/thesis_fulltext.pdf

See Also

`evd`, `ismev` and `condmixt`

---

bckden *Boundary Corrected Kernel Density Estimation*

Description

Density, cumulative distribution function, quantile function and random number generation for the boundary corrected kernel density estimators with a constant bandwidth `lambda`. The kernel centres (typically the data) are given by `kerncentres`.

Usage

```
dbckden(x, kerncentres, lambda = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, log = FALSE)

pbckden(q, kerncentres, lambda = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, lower.tail = TRUE)

qbckden(p, kerncentres, lambda = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, lower.tail = TRUE)

rbckden(n, kerncentres, lambda = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf)
```

Arguments

lambda	scalar value of fixed bandwidth, or NULL (default)
bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
xmax	upper bound on support, for copula and beta kernels only
offset	offset added to kernel centres, for logtrans
x	quantile
kerncentres	kernel centres (typically sample data)
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Boundary corrected kernel density estimation (KDE) to improve the bias properties near the boundary. A fairly wide range of methods are implemented for the user to choose from to cope with a lower boundary at zero and potentially also both upper and lower boundaries. Some boundary correction methods require a secondary correction for negative density estimates, so an option is available. Further, some methods also need to be normalised to ensure the density estimate is proper (i.e. integrates to one), so an option is provided to renormalise.

It assumes there is a lower boundary at zero, so prior transformation of the data would be required for alternative boundaries (including negation to use it for only an upper boundary).

Renormalisation of the kernel to integrate to unity is assumed by default (proper=TRUE), but the user can specify if the raw density estimate is provided instead (proper=FALSE). For the methods implemented thus far, this is needed for bcmethod="simple" which can be evaluated in closed form, and bcmethod="beta1" or bcmethod="beta2" which require numerical integration.

Correction of the density estimate to ensure non-negativity can be applied, which is only relevant for the `bcmethod="simple"` approach. The Jones and Foster (1996) method will be applied (`nn="jf96"`) by default. The non-negative value can simply be zeroed (`nn="zero"`). By default, correction is applied (`nn="none"`). This method can occassionally give an extra boundary bias for certain populations (e.g. Gamma(2, 1)), see their paper for details. Renormlisation should be used after these non-negativity corrections.

The non-negative correction is applied before renormalisation (when either is requested).

The boundary correction methods implemented are:

`bcmethod="simple"` is the default and applies the simple boundary correction method in equation (3.4) of Jones (1993) and is equivalent to the kernel weighted local linear fitting at the boundary. Normal kernels are used.

`bcmethod="renorm"` applies the renormalisation method discussed in Diggle (1985), where the kernels are simply truncated at the boundary and renormalised to unity. But this still exhibits a o(h) boundary bias. Normal kernels are used.

`bcmethod="reflect"` applies the reflection method of Boneva, Kendall and Stefanov (1971) which is equivalent to the dataset being supplemented by the same dataset negated. This method implicitly assumes f'(0)=0, so can causes extra artefacts at the boundary. Normal kernels are used.

`bcmethod="logtrans"` applies KDE on the log-scale and then transforms back (with explicit normalisation), following Marron and Ruppert (1992). This is the approach implmented in the `ks` package. As the KDE is applied on the log scale, the effective bandwidth on the original scale is not constant. Normal kernels are used on the log-scale. The `offset` option is only used for this method, to offset zero values, to prevent `log(0)`.

`bcmethod="beta1"` and `"beta2"` due to Chen (1999) which uses beta kernels and modified beta kernels respectively in the KDE. The `xmax` argument has been provided so that the user can have the beta kernels rescaled to be appropriate for a support of [0, xmax] rather than [0, 1].

`bcmethod="gamma1"` and `"gamma2"` due to Chen (2000) which uses gamma kernels and modified gamma kernels respectively in the KDE.

`bcmethod="copula"` due to Jones and Henderson (2007) uses bivariate normal copula based kernels in the KDE, essentially by taking condition slices thorugh the bivariate copula at each kernel centre. As with the `bcmethod="beta"` option the `xmax` argument has been provided to rescale the kernels over [0, xmax] rather than [0, 1]. In this case the bandwidth is defined as $lambda = 1 - \rho^2$, so is limited to (0, 1).

The examples below show a trick you can use to see the actual kernels used in the chosen boundary correction method.

The quantile function is rather complicated as there is typically no closed form solution, so in these cases it is obtained by approximation or numerical solution to $P(X \leq x_p) = p$ to find x_p. The quantile function `qbckden` evaluates the KDE cumulative distribution function over the range from `c(0, max(kerncentre) - 5*lambda)`. Outside of this range the quantiles are set to `0` for lower tail and `Inf` for upper tail. A sequence of values of length fifty times the number of kernels is first calculated. Spline based interpolation using `splinefun`, with default `monoH.FC` method, is then used to approximate the quantile function. This is a similar approach to that taken by Matt Wand in the `qkde` in the `ks` package.

Unlike the standard KDE, there is no general rule-of-thumb bandwidth for all these estimators, with only certain methods having a guideline in the literature, so none have been implmented. Hence, the user has to specify a bandwidth, but should consider using `fbckden` function for cross-validation likelihood fitting.

Random number generation is slow as inversion sampling using the (numerically evaluated) quantile function is implemented. Users may want to consider alternative approaches instead, like rejection sampling.

Value

dbckden gives the density, pbckden gives the cumulative distribution function, qbckden gives the quantile function and rbckden gives a random sample.

Note

Unlike all the other extreme value mixture model functions the bckden functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length.

The kernel centres kerncentres can either be a single datapoint or a vector of data. The kernel centres (kerncentres) and locations to evaluate density (x) and cumulative distribution function (q) would usually be different.

Default values are provided for all inputs, except for the fundamentals lambda, kerncentres, x, q and p. The default sample size for rbckden is 1.

The xmax option is only relevant for the beta and copula methods, so a warning is produced if this is changed from the default in other methods.

The renormalisation is only relevant for the bcmethod="simple", "beta1" and "beta2" approaches, so will not be applied in any other case (even if the user specifies proper=TRUE).

Non-negative correction will be applied by default, but this is only relevant for the bcmethod="simple" approach. It will not be applied in any other cases (even if the user specifies a method for nn).

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>. Based on code by Anna MacDonald produced for MATLAB.

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

Chen, S.X. (1999). Beta kernel estimators for density functions. Computational Statistics and Data Analysis 31, 1310-45.

Chen, S.X. (2000). Probability density function estimation using gamma kernels. Annals of the Institute of Statisical Mathematics 52(3), 471-480.

Boneva, L.I., Kendall, D.G. and Stefanov, I. (1971). Spline transformations: Three new diagnostic aids for the statistical data analyst (with discussion). Journal of the Royal Statistical Society B, 33, 1-70.

Diggle, P.J. (1985). A kernel method for smoothing point process data. Applied Statistics 34, 138-147.

Marron, J.S. and Ruppert, D. (1994) Transformations to reduce boundary bias in kernel density estimation, Journal of the Royal Statistical Society. Series B 56(4), 653-671.

Jones, M.C. and Henderson, D.A. (2007). Kernel-type density estimation on the unit interval. Biometrika 94(4), 977-984.

See Also

kden, density, logspline and dkde.

Other bckdengpd bckden kden: bckdengpd, dbckdengpd, pbckdengpd, qbckdengpd, rbckdengpd

Examples

```
## Not run:
n=100
x = rgamma(n, shape = 1, scale = 2)
xx = seq(-0.5, 12, 0.01)
plot(xx, dgamma(xx, shape = 1, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.3), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")

#  # Trick to show actual kernels on the plot - just add one kernel centre at a time:
for (i in 1:min(n,20)){
  lines(xx, dbckden(xx, x[i], lambda = 0.3, proper = FALSE)*0.05, col = "blue")
}
# Notice the negative weights in the kernels for this approach

legend("topright", c("True Density", "Simple boundary correction", "KDE using density function",
"Boundary Corrected Kernels"),
lty = c(1, 1, 2, 1), lwd = c(1, 2, 2, 1), col = c("black", "red", "green", "blue"))

n=100
x = rbeta(n, shape1 = 3, shape2 = 2)*5
xx = seq(-0.5, 5.5, 0.01)
plot(xx, dbeta(xx/5, shape1 = 3, shape2 = 2)/5, type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.01, bcmethod = "beta2", xmax = 5),
  lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "Modified Beta KDE Using evmix",
  "KDE using density function"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "green"))

# Demonstrate renormalisation (usually small difference)
n=100
x = rgamma(n, shape = 2, scale = 2)
xx = seq(-0.5, 15, 0.01)
plot(xx, dgamma(xx, shape = 2, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = TRUE),
```

```
  lwd = 2, col = "red")
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = FALSE),
  lwd = 2, col = "purple")
legend("topright", c("True Density", "Simple BC with renomalisation",
"Simple BC without renomalisation"),
lty = 1, lwd = c(1, 2, 2), col = c("black", "red", "purple"))

## End(Not run)
```

---

bckdengpd *Boundary Corrected Kernel Density Estimators for Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the boundary corrected kernel density estimators for the bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the bandwidth `lambda`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dbckdengpd(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = 0, xmax = Inf, log = FALSE)

pbckdengpd(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = 0, xmax = Inf, lower.tail = TRUE)

qbckdengpd(p, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = 0, xmax = Inf, lower.tail = TRUE)

rbckdengpd(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = 0, xmax = Inf)
```

Arguments

`x`	quantile
`kerncentres`	kernel centres (typically sample data)
`lambda`	scalar value of fixed bandwidth, or `NULL` (default)

bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
offset	offset added to kernel centres, for logtrans
xmax	upper bound on support, for copula and beta kernels only
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1]

Details

Extreme value mixture model combining boundary corrected kernel density estimators for the bulk below the threshold and GPD for upper tail.

See gpd for details of boundary corrected kernel density estimators.

The user can pre-specify phiu permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the kernel density estimation Using normal kernel (phiu=TRUE), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = (H(u)) + [1 - (H(u))]G(x)$$

where $H(x)$ and $G(X)$ are the boundary correction kernel density estimator and conditional GPD cumulative distribution functions (i.e. pbckden(x, kerncentres, lambda, bcmethod, proper, nn, offset, xma and pgpd(x, u, sigmau, xi)).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - (H(u))$.

Value

dbckdengpd gives the density, pbckdengpd gives the cumulative distribution function, qbckdengpd gives the quantile function and rbckdengpd gives a random sample.

Note

Unlike all the other extreme value mixture model functions the bckdengpd functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length. The kerncentres can also be a scalar or vector.

The kernel centres kerncentres can either be a single datapoint or a vector of data. The kernel centres (kerncentres) and locations to evaluate density (x) and cumulative distribution function (q) would usually be different.

Default values are provided for all inputs, except for the fundamentals kerncentres, x, q and p. The default sample size for rbckdengpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The normal mean nmean and GPD threshold u will also require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

gpd and dnorm

Other bckdengpd bckden kden: bckden, dbckden, pbckden, qbckden, rbckden

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rgamma(500, 2, 1)
xx = seq(0.1, 10, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1))

plot(xx, pbckdengpd(xx, kerncentres, lambda = 0.1), type = "l")
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.1, xi = 0.3), col = "red")
```

```
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.1, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1, cex = 0.5)

kerncentres=rweibull(1000, 2, 1)
x = rbckdengpd(1000, kerncentres, lambda = 0.1, phiu = TRUE)
xx = seq(0.01, 3.5, 0.01)
hist(x, breaks = 100, freq = FALSE)
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, phiu = TRUE))

plot(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, xi=0, phiu = 0.1), type = "l")
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, xi=-0.2, phiu = 0.1), col = "red")
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, xi=0.2, phiu = 0.1), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`betagpd` *Beta Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the beta shape `wshape` and scale `wscale`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dbetagpd(x, bshape1 = 1, bshape2 = 1,
  u = qbeta(0.9, bshape1, bshape2),
  sigmau = sqrt(bshape1 * bshape2/(bshape1 + bshape2)^2/(bshape1 + bshape2 + 1)),
  xi = 0, phiu = TRUE, log = FALSE)

pbetagpd(q, bshape1 = 1, bshape2 = 1,
  u = qbeta(0.9, bshape1, bshape2),
  sigmau = sqrt(bshape1 * bshape2/(bshape1 + bshape2)^2/(bshape1 + bshape2 + 1)),
  xi = 0, phiu = TRUE, lower.tail = TRUE)

qbetagpd(p, bshape1 = 1, bshape2 = 1,
  u = qbeta(0.9, bshape1, bshape2),
  sigmau = sqrt(bshape1 * bshape2/(bshape1 + bshape2)^2/(bshape1 + bshape2 + 1)),
  xi = 0, phiu = TRUE, lower.tail = TRUE)

rbetagpd(n = 1, bshape1 = 1, bshape2 = 1,
  u = qbeta(0.9, bshape1, bshape2),
  sigmau = sqrt(bshape1 * bshape2/(bshape1 + bshape2)^2/(bshape1 + bshape2 + 1)),
  xi = 0, phiu = TRUE)
```

Arguments

bshape1	beta shape 1 (non-negative)
bshape2	beta shape 2 (non-negative)
u	threshold over $(0, 1)$
phiu	probability of being above threshold [0,1] or TRUE
x	quantile
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining beta distribution for the bulk below the threshold and GPD for upper tail. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the beta bulk model.

The usual beta distribution is defined on $[0, 1]$, but this mixture is generally not limited in the upper tail $[0, \infty]$, except for the usual upper tail limits for the GPD when `xi<0` discussed in gpd. Therefore, the threshold is limited to $(0, 1)$.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the beta bulk model (`phiu=TRUE`), upto the threshold $0 \leq x \leq u < 1$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the beta and conditional GPD cumulative distribution functions (i.e. `pbeta(x, wshape, wscale)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 \leq x \leq u < 1$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

See gpd for details of GPD upper tail component and dbeta for details of beta bulk component.

Value

dbetagpd gives the density, pbetagpd gives the cumulative distribution function, qbetagpd gives the quantile function and rbetagpd gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main inputs (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rbetagpd any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default sample size for rbetagpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Beta_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

MacDonald, A. (2012). Extreme value mixture modelling with medical and industrial applications. PhD thesis, University of Canterbury, New Zealand. http://ir.canterbury.ac.nz/bitstream/10092/6679/1/thesis_fulltext.pdf

See Also

gpd and dbeta

Other betagpd: fbetagpd, lbetagpd, nlbetagpd

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rbetagpd(1000, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2)
xx = seq(-0.1, 2, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, dbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2))

# three tail behaviours
plot(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2), type = "l")
lines(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2, xi = 0.3), col = "red")
lines(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rbetagpd(1000, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, dbetagpd(xx, bshape1 = 2, bshape2 = 0.6, u = 0.7, phiu = 0.5))
```

```
plot(xx, dbetagpd(xx, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5, xi=0), type = "l")
lines(xx, dbetagpd(xx, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5, xi=-0.2), col = "red")
lines(xx, dbetagpd(xx, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5, xi=0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

dwm *Dynamically Weighted Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the dynamically weighted mixture model. The parameters are the Weibull shape `wshape` and scale `wscale`, Cauchy location `cmu`, Cauchy scale `ctau`, GPD scale `sigmau`, shape `xi` and initial value for the quantile `qinit`.

Usage

```
ddwm(x, wshape = 1, wscale = 1, cmu = 1, ctau = 1,
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, log = FALSE)

pdwm(q, wshape = 1, wscale = 1, cmu = 1, ctau = 1,
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, lower.tail = TRUE)

qdwm(p, wshape = 1, wscale = 1, cmu = 1, ctau = 1,
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, lower.tail = TRUE, qinit = NULL)

rdwm(n = 1, wshape = 1, wscale = 1, cmu = 1, ctau = 1,
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0)
```

Arguments

`cmu`	Cauchy location
`ctau`	Cauchy scale
`qinit`	vector of initial values for the quantile estimate
`x`	quantile
`wshape`	Weibull shape (non-negative)
`wscale`	Weibull scale (non-negative)
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

The dynamic weighted mixture model combines a Weibull for the bulk model with GPD for the tail model. However, unlike all the other mixture models the GPD is defined over the entire range of support rather than as a conditional model above some threshold. A transition function is used to apply weights to transition between the bulk and GPD for the upper tail, thus providing the dynamically weighted mixture. They use a Cauchy cumulative distribution function for the transition function.

The density function is then a dynamically weighted mixture given by:

$$f(x) = [1 - p(x)]h(x) + p(x)g(x)/r$$

where $h(x)$ and $g(x)$ are the Weibull and unscaled GPD density functions respectively (i.e. `pweibull(x, wshape, wsca` and `pgpd(x, u, sigmau, xi)`). The Cauchy cumulative distribution function used to provide the transition is defined by $p(x)$ (i.e. `pcauchy(x, cmu, ctau`. The normalisation constant r ensures a proper density.

The quantile function is not available in closed form, so has to be solved numerically. The argument `qinit` is the initial quantile estimate which is used for numerical optimisation and should be set to a reasonable guess. When the `qinit` is `NULL`, the initial quantile value is given by the midpoint between the Weibull and GPD quantiles. As with the other inputs `qinit` is also vectorised, but `R` does not permit vectors combining `NULL` and numeric entries.

Value

`ddwm` gives the density, `pdwm` gives the cumulative distribution function, `qdwm` gives the quantile function and `rdwm` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rdwm` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rdwm` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Frigessi, A., Haug, O. and Rue, H. (2002). A dynamic mixture model for unsupervised tail estimation without threshold selection. Extremes 5 (3), 219-235

See Also

weibullgpd, gpd and dweibull

Examples

```
## Not run:
par(mfrow = c(2, 2))
xx = seq(0.001, 5, 0.01)
f = ddwm(xx, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2,
  ylab = "density", main = "Plot example in Frigessi et al. (2002)")
lines(xx, dgpd(xx, xi = 1, sigmau = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, dweibull(xx, shape = 2, scale = 1/gamma(1.5)), col = "blue", lty = 2, lwd = 2)
legend('topright', c('DWM', 'Weibull', 'GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# three tail behaviours
plot(xx, pdwm(xx, xi = 0), type = "l")
lines(xx, pdwm(xx, xi = 0.3), col = "red")
lines(xx, pdwm(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)), col=c("black", "red", "blue"), lty = 1)

x = rdwm(10000, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.1)
xx = seq(0, 15, 0.01)
hist(x, freq = FALSE, breaks = 100)
lines(xx,ddwm(xx, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.1),
  lwd = 2, col = 'black')

plot(xx, pdwm(xx, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.1),
 xlim = c(0, 15), type = 'l', lwd = 2,
  xlab = "x", ylab = "F(x)")
lines(xx, pgpd(xx, sigmau = 1, xi = 0.1), col = "red", lty = 2, lwd = 2)
lines(xx, pweibull(xx, shape = 2, scale = 1/gamma(1.5)), col = "blue", lty = 2, lwd = 2)
legend('bottomright', c('DWM', 'Weibull', 'GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

## End(Not run)
```

---

evmix.diag *Diagnostic Plots for Extreme Value Mixture Models*

Description

The classic four diagnostic plots for evaluating extreme value mixture models: 1) return level plot, 2) Q-Q plot, 3) P-P plot and 4) density plot. Each plot is available individually or as the usual 2x2 collection.

Usage

```
evmix.diag(modelfit, upperfocus = TRUE, ci = TRUE,
  alpha = 0.05, N = 1000, legend = FALSE, ...)

rlplot(modelfit, upperfocus = TRUE, ci = TRUE,
```

```
  alpha = 0.05, N = 1000, legend = TRUE, ...)

qplot(modelfit, upperfocus = TRUE, ci = TRUE,
  alpha = 0.05, N = 1000, legend = TRUE, ...)

pplot(modelfit, upperfocus = TRUE, ci = TRUE,
  alpha = 0.05, N = 1000, legend = TRUE, ...)

densplot(modelfit, upperfocus = TRUE, legend = TRUE, ...)
```

Arguments

`modelfit`	fitted extreme value mixture model object
`upperfocus`	logical, should plot focus on upper tail?
`ci`	logical, should Monte Carlo based CI's be plotted
`alpha`	logical, significance level (0, 1)
`N`	number of Monte Carlo simulation for CI (N>=10)
`legend`	logical, should legend be included
`...`	further arguments to be passed to the plotting functions

Details

Model diagnostics are available for all the fitted extreme mixture models in the `evmix` package. These `modelfit` is output by all the fitting functions, e.g. `fgpd` and `fnormgpd`.

Consistent with `plot` function in the `evd` library the `ppoints` to estimate the empirical cumulative probabilities. The current default behaviour of this function is to use

$$(i - 0.5)/n$$

as the estimate for the ith order statistic of the given sample of size n.

The return level plot quantile (x_p where $P(X \leq x_p) = p$ on the y-axis and the (approximate) return period $1/p$ is shown on the x-axis. It is approximate as the tranformation $-log(-log(p)) \approx 1/p$ is used for the x-axis, which is common in extreme value application as Type I ($\xi = 0$) upper tail behaviour will be linear on this scale. The approximation is better for smaller upper tail probability.

The crosses are the empirical quantiles/return levels (i.e. the ordered sample data) against their corresponding transfomred empirical return period (from `ppoints`). The solid line is the theoretical return level (quantile) function using the estimated model parameters. The estimated threshold `u` and tail fraction `phiu` are shown. For the two tailed models both thresholds `ul` and `ur` and corresponding tail fractions `phiul` and `phiur` are shown. The approximate pointwise confidence intervals for the quantiles are obtained by Monte Carlo simulation using the estimated parameters. Notice that these intervals ignore the parameter estimation uncertainty.

The Q-Q and P-P plots have the empirical values on the y-axis and theoretical values from the fitted model on the x-axis.

The density plot provides a histogram of the sample data overlaid with the fitted density and a standard kernel density estimate using the `density` function. The default settings for the `density` function are used. Note that for distributions with bounded support (e.g. GPD) with high density near the boundary standard kernel density estimators exhibit a negative bias due to leakage past the boundary. So in this case they should not be taken too seriously.

For the kernel density estimates (i.e. `kden` and `bckden`) there is no threshold, so no upper tail focus is carried out.

See `plot.uevd` for more detailed explanations of these types of plots.

Value

rlplot gives the return level plot, qplot gives the Q-Q plot, pplot gives the P-P plot, densplot gives density plot and evmix.diag gives the collection of all 4.

Note

For all mixture models the missing values are removed by the fitting functions (e.g. fnormgpd and fgng). However, these are retained in the GPD fitting fgpd, as they are interpreted as values below the threshold.

By default all the plots focus in on the upper tail, but they can be used to display the fit over the entire range of support.

You cannot pass xlim or ylim to the plotting functions via ...

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Based on model diagnostic functions in the evd package.

http://en.wikipedia.org/wiki/Q-Q_plot

http://en.wikipedia.org/wiki/P-P_plot

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Coles S.G. (2004). An Introduction to the Statistical Modelling of Extreme Values. Springer-Verlag: London.

See Also

ppoints and plot.uvevd

Examples

```
## Not run:
x = sort(rnorm(1000))
fit = fnormgpd(x)
evmix.diag(fit)

# repeat without focussing on upper tail
par(mfrow=c(2,2))
rlplot(fit, upperfocus = FALSE)
qplot(fit, upperfocus = FALSE)
pplot(fit, upperfocus = FALSE)
densplot(fit, upperfocus = FALSE)

## End(Not run)
```

fbckden *Cross-validation MLE Fitting of Boundary Corrected Kernel Density Estimation*

Description

Maximum likelihood estimation for fitting boundary corrected kernel density estimator, by treating it as a mixture model.

Usage

```
fbckden(x, linit = NULL, extracentres = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, add.jitter = FALSE,
  factor = 0.1, amount = NULL, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

x	quantile
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
offset	offset added to kernel centres, for logtrans
xmax	upper bound on support, for copula and beta kernels only
finitelik	logical, should log-likelihood return finite value for invalid parameters
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
...	optional inputs passed to optim
linit	initial value for bandwidth parameter or NULL
add.jitter	logical, whether jitter is needed for rounded data
factor	see jitter
amount	see jitter

Details

The boundary corrected kernel density estimator is fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated bandwidth, variance and standard error are automatically output.

The beta1 and beta2 densities requires renormalisation which is achieved by numerical integration, so is very time consuming. Practically we have found leaving out the renormalisation proper=FALSE still yields reliable bandwidth estimates.

The cross-validation likelihood estimates of the bandwidth for the `simple`, `gamma1` and `gamma2` methods of boundary correction are biased high (particularly when there is a pole at the boundary) leading to oversmoothing. We have empirically found that leaving off the data from the upper tail in the likelihood appears to help, see examples for an implementation.

Missing values (`NA` and `NaN`) are assumed to be invalid data so are ignored.

Normally for likelihood estimation of the bandwidth the kernel centres and the data where the likelihood is evaluated are the same. However, when using KDE for extreme value mixture modelling the likelihood only those data in the bulk of the distribution should contribute to the likelihood, but all the data (including those beyond the threshold) should contribute to the density estimate. The `extracentres` option allows the use to specify extra kernel centres used in estimating the density, but not evaluated in the likelihood. The default is to just use the existing data, so `extracentres=NULL`.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation `finitelik=TRUE`. For invalid parameters, a zero likelihood is replaced with `exp(-1e6)`. The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default `std.err=TRUE` and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set `std.err=FALSE`.

Value

Returns a simple list with the following elements

`call`:	`optim` call
`x`:	(jittered) data vector `x`
`kerncentres`: actual kernel centres used `x`	
`init`:	`linit`
`optim`:	complete `optim` output
`mle`:	vector of MLE of bandwidth
`cov`:	variance of MLE of bandwidth
`se`:	standard error of MLE of bandwidth
`nllh`:	minimum negative cross-validation log-likelihood
`n`:	total sample size
`lambda`:	MLE of bandwidth
`bcmethod`:	boundary correction method
`proper`:	logical, whether renormalisation is requested
`nn`:	non-negative correction method
`offset`:	offset for log transformation method
`xmax`:	maximum value of scale beta or copula

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from `fpot` and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that

the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (`x` only) has been included in the fitting inputs, using the `jitter` function, to remove the ties. The default options red in the `jitter` are specified above, but the user can override these. Notice the default scaling `factor=0.1`, which is a tenth of the default value in the `jitter` function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

2) For heavy tailed populations the bandwidth is positively biased, giving oversmoothing (see example). The bias is due to the distance between the upper (or lower) order statistics not necessarily decaying to zero as the sample size tends to infinity. Essentially, as the distance between the two largest (or smallest) sample datapoints does not decay to zero, some smoothing between them is required (i.e. bandwidth cannot be zero). One solution to this problem is to splice the GPD at a suitable threshold to remove the problematic tail from the inference for the bandwidth, using either the `kdengpd` function for a single heavy tail or the `kdengng` function if both tails are heavy. See MacDonald et al (2013).

Note

When `linit=NULL` then the initial value for the bandwidth is calculated using `bw.nrd0` function.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott `<carl.scarrott@canterbury.ac.nz>`

References

`http://en.wikipedia.org/wiki/Kernel_density_estimation`

`http://en.wikipedia.org/wiki/Cross-validation_(statistics)`

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from `http://www.ine.pt/revstat/pdf/rs120102.pdf`

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: `http://www.math.canterbury.ac.nz/~c.scarrott`.

See Also

jitter, density and bw.nrd0

Examples

```
## Not run:
nk=50
x = rgamma(nk, shape = 3, scale = 1)
xx = seq(-1, 10, 0.01)
fit = fbckden(x, linit = 0.2, bcmethod = "renorm")
hist(x, nk/5, freq = FALSE)
rug(x)
for (i in 1:nk) lines(xx, dbckden(xx, x[i], lambda = fit$lambda, bcmethod = "renorm")*0.05)
lines(xx, dgamma(xx, shape = 3, scale = 1), col = "black")
lines(xx, dbckden(xx, x, lambda = fit$lambda, bcmethod = "renorm"), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "BC KDE fitted evmix",
"KDE Using density, default bandwidth"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "green"))

nk=100
x = rgamma(nk, shape = 0.5, scale = 1)
q75 = qgamma(0.75, shape = 0.5, scale = 1)
xx = seq(-1, 10, 0.01)
fit = fbckden(x, linit = 0.2, bcmethod = "simple")
fitnotail = fbckden(x[x <= q75], linit = 0.1, bcmethod = "simple", extracentres = x[x > q75])
hist(x, nk/5, freq = FALSE, ylim = c(0, 8))
rug(x)
lines(xx, dgamma(xx, shape = 0.5, scale = 1), col = "black")
lines(xx, dbckden(xx, x, lambda = fit$lambda, bcmethod = "simple"), lwd = 2, col = "red")
lines(xx, dbckden(xx, x, lambda = fitnotail$lambda, bcmethod = "simple"), lwd = 2, col = "blue")
legend("topright", c("True Density", "BC KDE (complete dataset)",
"BC KDE (upper tail ignored)"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "blue"))

## End(Not run)
```

---

fbckdengpd *Cross-validation MLE Fitting of Boundary Corrected Kernel Density Estimation and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting boundary corrected kernel density estimators for the bulk and GPD tail extreme value mixture model

Usage

```
fbckdengpd(x, phiu = TRUE, pvector = NULL,
  add.jitter = FALSE, factor = 0.1, amount = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

x	quantile
bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
offset	offset added to kernel centres, for logtrans
xmax	upper bound on support, for copula and beta kernels only
phiu	logical
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
add.jitter	logical, whether jitter is needed for rounded data
factor	see jitter
amount	see jitter
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

Extreme value mixture model combining boundary corrected kernel density estimators for the bulk below the threshold and GPD for upper tail is fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated parameters, their variance and standard error are automatically output.

Cross-validation likelihood is used for boundary corrected kernel density component, but standard likelihood is used for GPD component. The default value for phiu=TRUE so that the tail fraction is specified by boundary corrected kernel density estimators cumulative distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	(jittered) data vector x
kerncentres: actual kernel centres used x	
init:	pvector
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance of MLE parameters
se:	standard error of MLE parameters
nllh:	minimum negative cross-validation log-likelihood
allparams:	vector of MLE of model parameters, including phiu
allse:	vector of standard error of all parameters, including phiu
n:	total sample size
lambda:	MLE of bandwidth
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction
bcmethod:	boundary correction method
proper:	logical, whether renormalisation is requested
nn:	non-negative correction method
offset:	offset for log transformation method
xmax:	maximum value of scale beta or copula

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (x only) has been included in the fitting inputs, using the jitter function, to remove the ties. The default options red in the jitter are specified above, but the user can override these. Notice the default scaling factor=0.1, which is a tenth of the default value in the jitter function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

Note

When pvector=NULL then the initial value for the parameters are calculated type fkdengpdcon to see how.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: http://www.math.canterbury.ac.nz/~c.scarrott.

See Also

fkden, jitter, density and bw.nrd0

Examples

```
## Not run:
xx = seq(0.1, 10, 0.01)
x = rgamma(500, 2, 1)
pinit = c(0.1, quantile(x, 0.9), 1, 0.1)
fit = fbckdengpd(x, phiu = FALSE, pvector = pinit, std.err = FALSE, bcmethod = "reflect")
hist(x, 100, freq = FALSE,ylim=c(0,0.6))
lines(xx, dbckdengpd(xx, x, fit$lambda, fit$u, fit$sigmau, fit$xi,
  fit$phiu, bcmethod = "reflect"), col="blue")
abline(v = fit$u)

## End(Not run)
```

---

fbetagpd *MLE Fitting of Beta Bulk and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold

Usage

```
fbetagpd(x, phiu = TRUE, pvector = NULL, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values mixture model parameters (bshape1, bshape2, u, sigmau, xi) or NULL
phiu	logical
control	optimisation control list (see optim)
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with beta bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Non-positive data are ignored. Values above 1 must come from GPD component, as threshold u<1.

The default value for phiu=TRUE so that the tail fraction is specified by beta distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu

allse:	vector of standard error of all parameters and phiu
n:	total sample size
bshape1:	MLE of beta shape 1
bshape2:	MLE of beta shape 2
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fbetagpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/beta_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

MacDonald, A. (2012). Extreme value mixture modelling with medical and industrial applications. PhD thesis, University of Canterbury, New Zealand. http://ir.canterbury.ac.nz/bitstream/10092/6679/1/thesis_fulltext.pdf

See Also

lgpd and gpd

Other betagpd: betagpd, dbetagpd, lbetagpd, nlbetagpd, pbetagpd, qbetagpd, rbetagpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rbeta(1000, shape1 = 0.5, shape2 = 2)
xx = seq(-0.1, 2, 0.01)
```

```
y = dbeta(xx, shape1 = 0.5, shape2 = 2)

# Bulk model base tail fraction
fit = fbetagpd(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, y)
lines(xx, dbetagpd(xx, bshape1 = fit$bshape1, bshape2 = fit$bshape2, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fbetagpd(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dbetagpd(xx, bshape1 = fit$bshape1, bshape2 = fit$bshape2,, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dbetagpd(xx, bshape1 = fit2$bshape1, bshape2 = fit2$bshape2,, u = fit2$u,
  sigmau = fit2$sigmau, xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fdwm` *MLE Fitting of Dynamically Weighted Mixture Model*

Description

Maximum likelihood estimation for fitting the dynamically weighted mixture model

Usage

```
fdwm(x, pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`pvector`	vector of initial values of mixture model parameters (`wshape`, `wscale`, `cmu`, `ctau`, `sigmau`, `xi`) or `NULL`
`x`	vector of sample data
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`control`	optimisation control list (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`

Details

The dynamically weighted mixture model is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Non-positive data are ignored.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu
allse:	vector of standard error of all parameters and phiu
n:	total sample size
wshape:	MLE of Weibull shape
wscale:	MLE of Weibull scale
mu:	MLE of Cauchy location
tau:	MLE of Cauchy scale
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fdwm to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Frigessi, A., O. Haug, and H. Rue (2002). A dynamic mixture model for unsupervised tail estimation without threshold selection. Extremes 5 (3), 219-235

See Also

lgpd and gpd

Other dwm: ldwm, nldwm

Examples

```
## Not run:
x = rweibull(1000, shape = 2)
xx = seq(-1, 4, 0.01)
y = dweibull(xx, shape = 2)

fit = fdwm(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 4))
lines(xx, y)
lines(xx, ddwm(xx, wshape = fit$wshape, wscale = fit$wscale, cmu = fit$cmu, ctau = fit$ctau,
  sigmau = fit$sigmau, xi = fit$xi), col="red")

## End(Not run)
```

---

fgammagpd *MLE Fitting of Gamma Bulk and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold

Usage

```
fgammagpd(x, phiu = TRUE, pvector = NULL, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values mixture model parameters (gshape, gscale, u, sigmau, xi) or NULL
phiu	logical
control	optimisation control list (see optim)
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with gamma bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Negative data are ignored.

The default value for phiu=TRUE so that the tail fraction is specified by gamma distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters

cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu
allse:	vector of standard error of all parameters and phiu
n:	total sample size
gshape:	MLE of gamma shape
gscale:	MLE of gamma scale
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fgammagpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Camma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other gammagpd: dgammagpd, gammagpd, lgammagpd, nlgammagpd, pgammagpd, qgammagpd, rgammagpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rgamma(1000, shape = 2)
xx = seq(-1, 10, 0.01)
y = dgamma(xx, shape = 2)

# Bulk model base tail fraction
fit = fgammagpd(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, y)
lines(xx, dgammagpd(xx, gshape = fit$gshape, gscale = fit$gscale, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fgammagpd(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dgammagpd(xx, gshape = fit$gshape, gscale = fit$gscale, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dgammagpd(xx, gshape = fit2$gshape, gscale = fit2$gscale, u = fit2$u,
  sigmau = fit2$sigmau, xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fgammagpdcon` *MLE Fitting of Gamma Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
fgammagpdcon(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`pvector`	vector of initial values mixture model parameters (`gshape`, `gscale`, `u`, `xi`) or `NULL`
`x`	vector of sample data
`phiu`	logical
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)

control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with gamma bulk and GPD tail with a continuity constraint is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Negative data are ignored.

The default value for phiu=TRUE so that the tail fraction is specified by gamma distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including sigmau and phiu
allse:	vector of standard error of all parameters, including sigmau and phiu
n:	total sample size
gshape:	MLE of gamma shape
gscale:	MLE of gamma scale
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fgammagpdcon to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Camma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other gammagpdcon: dgammagpdcon, gammagpdcon, lgammagpdcon, nlgammagpdcon, pgammagpdcon, qgammagpdcon, rgammagpdcon

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rgamma(1000, shape = 2)
xx = seq(-1, 10, 0.01)
y = dgamma(xx, shape = 2)

# Bulk model base tail fraction
fit = fgammagpdcon(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, y)
lines(xx, dgammagpdcon(xx, gshape = fit$gshape, gscale = fit$gscale, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fgammagpdcon(x, phiu = FALSE, std.err = FALSE)
```

```
plot(xx, y, type = "l")
lines(xx, dgammagpdcon(xx, gshape = fit$gshape, gscale = fit$gscale, u = fit$u,
 xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dgammagpdcon(xx, gshape = fit2$gshape, gscale = fit2$gscale, u = fit2$u,
 xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fgkg` *Cross-validation MLE Fitting of Kernel Density Estimation for Bulk and GPD for Both Upper and Lower Tails in Extreme Value Mixture Model*

Description

Maximum likelihood estimation for the extreme value mixture model with kernel density estimation using normal kernel for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails

Usage

```
fgkg(x, phiul = TRUE, phiur = TRUE, pvector = NULL,
  add.jitter = FALSE, factor = 0.1, amount = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`x`	vector of sample data
`pvector`	vector of initial values of mixture model parameters (`nmean`, `nsd`, `u`, `sigmau`, `xi`) or `NULL`
`add.jitter`	logical, whether jitter is needed for rounded data
`factor`	see `jitter`
`amount`	see `jitter`
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`control`	optimisation control list (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`
`phiul`	logical
`phiur`	logical

Details

Extreme value mixture model combining for the extreme value mixture model with kernel density estimation using normal kernel for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails. Fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated parameters, their variance and standard error are automatically output.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD components. The default value for `phiul=TRUE` so that the tail fraction is specified by normal distribution $\phi_u = 1 - mean(H(ul))$. When `phiul=FALSE` then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion below the threshold `ul`. In this case the standard error for `phiu` is estimated and output as `sephiu`.

Missing values (`NA` and `NaN`) are assumed to be invalid data so are ignored, which is inconsistent with the `evd` library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation `finitelik=TRUE`. For invalid parameters, a zero likelihood is replaced with `exp(-1e6)`. The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default `std.err=TRUE` and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set `std.err=FALSE`.

Value

Returns a simple list with the following elements

`call`:	`optim` call
`x`:	(jittered) data vector `x`
`kerncentres`: actual kernel centres used `x`	
`init`:	`pvector`
`optim`:	complete `optim` output
`mle`:	vector of MLE of parameters
`cov`:	variance of MLE parameters
`se`:	standard error of MLE parameters
`nllh`:	minimum negative cross-validation log-likelihood
`allparams`:	vector of MLE of model parameters, including `phiul` and `phiur`
`allse`:	vector of standard error of all parameters, including `phiul` and `phiur`
`n`:	total sample size
`lambda`:	MLE of bandwidth
`ul`:	lower threshold
`sigmaul`:	MLE of lower tail GPD scale
`xil`:	MLE of lower tail GPD shape
`phiul`:	MLE of lower tail fraction
`ur`:	upper threshold
`sigmaur`:	MLE of upper tail GPD scale
`xir`:	MLE of upper tail GPD shape
`phiur`:	MLE of upper tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (x only) has been included in the fitting inputs, using the jitter function, to remove the ties. The default options red in the jitter are specified above, but the user can override these. Notice the default scaling factor=0.1, which is a tenth of the default value in the jitter function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

Note

When pvector=NULL then the initial value for the parameters are calculated type fkdengpdcon to see how.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: http://www.math.canterbury.ac.nz/~c.scarrott.

See Also

fkdengpd, fkden, jitter, density and bw.nrd0

Other gkg: dgkg, gkg, lgkg, nlgkg, pgkg, qgkg, rgkg

Examples

```
## Not run:
x = rnorm(1000, 0, 1)
fit = fgkg(x, phiul = FALSE, phiur = FALSE, std.err = FALSE)
hist(x, 100, freq = FALSE, xlim = c(-5, 5))
xx = seq(-5, 5, 0.01)
lines(xx, dgkg(xx, x, fit$lambda, fit$ul, fit$sigmaul, fit$xil, fit$phiul,
 fit$ur, fit$sigmaur, fit$xir, fit$phiur), col="blue")
abline(v = fit$ul)
abline(v = fit$ur)

## End(Not run)
```

---

fgng *MLE Fitting of Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model*

Description

Maximum likelihood estimation for the extreme value mixture model with normal for bulk distribution between the lower and upper thresholds with conditional GPD for the two tails.

Usage

```
fgng(x, phiul = TRUE, phiur = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values of mixture model parameters or NULL
phiul	logical
phiur	logical
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with normal bulk and GPD for both tails is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

`pvector` is (`nmean`, `nsd`, `ul`, `sigmaul`, `xil`, `ur`, `sigmaur`, `xir`)

The default values for `phiul=TRUE` and `phiur=TRUE` so that the corresponding tail fractions are specified by normal distribution $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$. When `phiul=FALSE` and `phiur=FALSE` then the corresponding tail fractions are treated as an extra parameter estimated using the MLE which is the sample proportion beyond the corresponding threshold. In this case the standard error for `phiul` and `phiur` are estimated and output as `sephiul` and `sephiur`.

Missing values (`NA` and `NaN`) are assumed to be invalid data so are ignored, which is inconsistent with the `evd` library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation `finitelik=TRUE`. For invalid parameters, a zero likelihood is replaced with `exp(-1e6)`. The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default `std.err=TRUE` and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set `std.err=FALSE`.

Value

Returns a simple list with the following elements

`x`:	data vector `x`
`init`:	`pvector`
`optim`:	complete `optim` output
`mle`:	vector of MLE of model parameters
`cov`:	variance-covariance matrix of MLE of model parameters
`se`:	vector of standard errors of MLE of model parameters
`rate`:	`phiu` to be consistent with `evd`
`nllh`:	minimum negative log-likelihood
`allparams`:	vector of MLE of model parameters and tail fractions `phiul` and `phiur`
`allse`:	vector of standard error of model parameters and tail fractions `phiul` and `phiur`
`n`:	total sample size
`nmean`:	MLE of normal mean
`nsd`:	MLE of normal standard deviation
`ul`:	lower threshold
`sigmaul`:	MLE of lower tail GPD scale
`xil`:	MLE of lower tail GPD shape
`phiul`:	MLE of lower tail fraction
`ur`:	upper threshold
`sigmaur`:	MLE of upper tail GPD scale
`xir`:	MLE of upper tail GPD shape
`phiur`:	MLE of upper tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and tail fractions phiul and phiur. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fgng to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameters to xil=0 or xir=0 as depending on the optimisation method it may be get stuck.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

fnormgpd, lgpd and gpd

Other gng: dgng, gng, lgng, nlgng, pgng, qgng, rgng

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Bulk model base tail fraction
fit = fgng(x, phiul = TRUE, phiur = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6), main = "N(0, 1)")
lines(xx, y)
lines(xx, dgng(xx, nmean = fit$nmean, nsd = fit$nsd,
```

```
  ul = fit$ul, sigmaul = fit$sigmaul, xil = fit$xil, phiul = TRUE,
  ur = fit$ur, sigmaur = fit$sigmaur, xir = fit$xir, phiur = TRUE), col="red")
abline(v = c(fit$ul, fit$ur))

# Parameterised tail fraction
fit2 = fgng(x, phiul = TRUE, phiur = TRUE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dgng(xx, nmean = fit$nmean, nsd = fit$nsd,
  ul = fit$ul, sigmaul = fit$sigmaul, xil = fit$xil, phiul = TRUE,
  ur = fit$ur, sigmaur = fit$sigmaur, xir = fit$xir, phiur = TRUE), col="red")
lines(xx, dgng(xx, nmean = fit2$nmean, nsd = fit2$nsd,
  ul = fit2$ul, sigmaul = fit2$sigmaul, xil = fit2$xil, phiul = fit2$phiul,
  ur = fit2$ur, sigmaur = fit2$sigmaur, xir = fit2$xir, phiur = fit2$phiur), col="blue")
abline(v = c(fit$ul, fit$ur), col = "red")
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)
x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Two tail is safest if bulk has lower tail which is not normal tail
x = rt(1000, df = 3)
xx = seq(-10, 10, 0.01)
y = dt(xx, df = 3)

# Bulk model base tail fraction
fit = fnormgpd(x, phiu = FALSE, std.err = FALSE)
fit2 = fgng(x, phiul = FALSE, phiur = FALSE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-10, 10), main = "t (df=3)")
lines(xx, y)
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd,
  u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu), col="red")
abline(v = fit$u)

# Bulk model base tail fraction
plot(xx, y, type = "l")
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd,
  u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu), col="red")
lines(xx, dgng(xx, nmean = fit2$nmean, nsd = fit2$nsd,
  ul = fit2$ul, sigmaul = fit2$sigmaul, xil = fit2$xil, phiul = fit2$phiul,
  ur = fit2$ur, sigmaur = fit2$sigmaur, xir = fit2$xir, phiur = fit2$phiur), col="blue")
abline(v = c(fit$ul, fit$ur), col = "red")
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density","GPD Upper Tail Only","GPD Both Tails"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

fgngcon *MLE Fitting of Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model with Continuity Constraints*

Description

Maximum likelihood estimation for the extreme value mixture model with normal for bulk distribution between the lower and upper thresholds with conditional GPD for the two tails with continuity constraints

Usage

```
fgngcon(x, phiul = TRUE, phiur = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

x	vector of sample data
phiul	logical
phiur	logical
pvector	vector of initial values of mixture model parameters or NULL
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with normal bulk and GPD for both tails with continuity constraints is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

pvector is (nmean, nsd, ul, sigmaul, xil, ur, sigmaur, xir)

The default values for phiul=TRUE and phiur=TRUE so that the corresponding tail fractions are specified by normal distribution $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$. When phiul=FALSE and phiur=FALSE then the corresponding tail fractions are treated as an extra parameter estimated using the MLE which is the sample proportion beyond the corresponding threshold. In this case the standard error for phiul and phiur are estimated and output as sephiul and sephiur.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including sigmaul, sigmaur and tail fractions phiul and phiur
allse:	vector of standard error of model parameters, including sigmaul, sigmaurand tail fractions phiul and phi
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
ul:	lower threshold
sigmaul:	MLE of lower tail GPD scale
xil:	MLE of lower tail GPD shape
phiul:	MLE of lower tail fraction
ur:	upper threshold
sigmaur:	MLE of upper tail GPD scale
xir:	MLE of upper tail GPD shape
phiur:	MLE of upper tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and tail fractions phiul and phiur. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fgngcon to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameters to xil=0 or xir=0 as depending on the optimisation method it may be get stuck.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

fgng, fnormgpd, lgpd and gpd

Other gngcon: dgngcon, gngcon, lgngcon, nlgngcon, pgngcon, qgngcon, rgngcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Bulk model base tail fraction
fit = fgngcon(x, phiul = TRUE, phiur = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6), main = "N(0, 1)")
lines(xx, y)
lines(xx, dgngcon(xx, nmean = fit$nmean, nsd = fit$nsd,
  ul = fit$ul, xil = fit$xil, phiul = TRUE,
  ur = fit$ur, xir = fit$xir, phiur = TRUE), col="red")
abline(v = c(fit$ul, fit$ur))

# Parameterised tail fraction
fit2 = fgngcon(x, phiul = TRUE, phiur = TRUE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dgngcon(xx, nmean = fit$nmean, nsd = fit$nsd,
  ul = fit$ul, xil = fit$xil, phiul = TRUE,
  ur = fit$ur, xir = fit$xir, phiur = TRUE), col="red")
lines(xx, dgngcon(xx, nmean = fit2$nmean, nsd = fit2$nsd,
  ul = fit2$ul, xil = fit2$xil, phiul = fit2$phiul,
  ur = fit2$ur, xir = fit2$xir, phiur = fit2$phiur), col="blue")
abline(v = c(fit$ul, fit$ur), col = "red")
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)
x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Two tail is safest if bulk has lower tail which is not normal tail
x = rt(1000, df = 3)
xx = seq(-10, 10, 0.01)
y = dt(xx, df = 3)

# Bulk model base tail fraction
fit = fnormgpd(x, phiu = FALSE, std.err = FALSE)
fit2 = fgngcon(x, phiul = FALSE, phiur = FALSE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-10, 10), main = "t (df=3)")
lines(xx, y)
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd,
```

```
  u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu), col="red")
abline(v = fit$u)

# Bulk model base tail fraction
plot(xx, y, type = "l")
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd,
  u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu), col="red")
lines(xx, dgngcon(xx, nmean = fit2$nmean, nsd = fit2$nsd,
  ul = fit2$ul, xil = fit2$xil, phiul = fit2$phiul,
  ur = fit2$ur, xir = fit2$xir, phiur = fit2$phiur), col="blue")
abline(v = c(fit$ul, fit$ur), col = "red")
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density","GPD Upper Tail Only","GPD Both Tails"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fgpd` *MLE Fitting of Generalised Pareto Distribution (GPD)*

Description

Maximum likelihood estimation for fitting the GPD with parameters scale `sigmau` and shape `xi` to the threshold exceedances, conditional on being above a threshold `u`. Unconditional likelihood fitting also provided when the probability `phiu` of being above the threshold `u` is given.

Usage

```
fgpd(x, u = 0, phiu = NULL, pvector = NULL,
  std.err = TRUE, method = "BFGS", finitelik = TRUE, ...)
```

Arguments

`x`	vector of sample data
`pvector`	vector of initial values of GPD parameters (`sigmau`, `xi`) or `NULL`
`phiu`	probability of being above threshold [0,1] or `NULL`
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`
`u`	threshold

Details

The GPD is fitted to the exceedances of the threshold `u` using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

The default value for `phiu` is NULL, which means it will be estimated as the MLE of the tail fraction which is the sample proportion above the threshold. In this case the standard error for `phiu`

is estimated and output as sephiu. Consistent with the evd library the missing values (NA and NaN) are assumed to be below the threshold.

Otherwise, phiu can be specified as any value over [0, 1] leading to the unconditional log-likelihood being used for estimation. In this case the standard error will be output as NA. The value of phiu does not effect the GPD parameter estimates, only the value of the likelihood, as:

$$L(\sigma_u, \xi; u, \phi_u) = (\phi_u^{n_u}) L(\sigma_u, \xi; u, \phi_u = 1)$$

where the GPD has scale σ_u and shape ξ, the threshold is u and nu is the number of exceedances. A non-unit value for phiu simply scales the likelihood, and shifts the log-likelihood, thus the GPD parameter estimates are invariant to phiu.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of GPD parameters and (possibly given) phiu
allse:	vector of standard error of GPD parameters and (possibly given) phiu
n:	total sample size
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the GPD, the MLE fitting only permits single scalar values for each parameter, phiu and threshold u. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fgpd to see the default formulae used. The GPD fitting is not very sensitive to the initial values, so you will rarely have to give alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

Default values for the threshold u=0 and tail fraction phiu=NULL are given. If the threshold is left as the default u=0 and the tail fraction set to phiu=1 then MLE assumes tha excesses over the threshold are given, rather than exceedances.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Based on GPD fitting function in the evd package.

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

See Also

dgpd, fpot and fitdistr

Other gpd: dgpd, gpd, lgpd, nlgpd, pgpd, qgpd, rgpd

Examples

```
par(mfrow=c(2,1))
x = rgpd(1000, u = 10, sigmau = 5, xi = 0.2)
xx = seq(0, 100, 0.1)
hist(x, breaks = 100, freq = FALSE, xlim = c(0, 100))
lines(xx, dgpd(xx, u = 10, sigmau = 5, xi = 0.2))
fit = fgpd(x, u = 10, phiu = NULL, std.err = FALSE)
lines(xx, dgpd(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi), col="red")

# This time with phiu
x = rnorm(10000)
xx = seq(-4, 4, 0.01)
hist(x, breaks = 200, freq = FALSE, xlim = c(0, 4))
lines(xx, dnorm(xx), lwd = 2)
fit = fgpd(x, u = 1, phiu = NULL, std.err = FALSE)
lines(xx, dgpd(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu),
  col = "red", lwd = 2)
legend("topright", c("True Density","Fitted Density"), col=c("black", "red"), lty = 1)
```

fhpd *MLE Fitting of Hybrid Pareto Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the hybrid Pareto extreme value mixture model

Usage

```
fhpd(x, pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values of mixture model parameters (nmean, nsd, xi) or NULL
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The hybrid Pareto model is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters

cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including u, sigmau and phiu
allse:	vector of standard error of all parameters, including u, sigmau and phiu
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fhpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

lgpd and gpd The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the likelihood of the hybrid Pareto hpareto.negloglike and fitting hpareto.fit.

Other hpd: dhpd, hpd, lhpd, nlhpd, phpd, qhpd, rhpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Hybrid Pareto provides reasonable fit for asymmetric heavy tailed distribution
# but not for cases such as the normal distribution
fit = fhpd(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dhpd(xx, nmean = fit$nmean, nsd = fit$nsd,
  xi = fit$xi), col="red")
abline(v = fit$u)

# Notice that if tail fraction is included a better fit is obtained
fit2 = fnormgpdcon(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dnormgpdcon(xx, nmean = fit2$nmean, nsd = fit2$nsd, u = fit2$u,
  xi = fit2$xi), col="blue")
abline(v = fit2$u)

## End(Not run)
```

---

fhpdcon — *MLE Fitting of Hybrid Pareto Extreme Value Mixture Model with Single Continuity Constraint*

Description

Maximum likelihood estimation for fitting the hybrid Pareto extreme value mixture modelwith a single continuiuty constraint

Usage

```
fhpdcon(x, pvector = NULL, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values of mixture model parameters (nmean, nsd, u, xi) or NULL
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The hybrid Pareto model is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including sigmau and phiu
allse:	vector of standard error of all parameters, includingsigmau and phiu
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fhpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

lgpd and gpd The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the likelihood of the hybrid Pareto hpareto.negloglike and fitting hpareto.fit.

Other hpdcon: dhpdcon, hpdcon, lhpdcon, nlhpdcon, phpdcon, qhpdcon, rhpdcon

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Hybrid Pareto provides reasonable fit for asymmetric heavy tailed distribution
# but not for cases such as the normal distribution
fitcon = fhpdcon(x, std.err = FALSE)
fit = fhpd(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dhpdcon(xx, nmean = fitcon$nmean, nsd = fitcon$nsd, u = fitcon$u,
  xi = fitcon$xi), col="red")
abline(v = fit$u)
lines(xx, dhpd(xx, nmean = fit$nmean, nsd = fit$nsd, xi = fit$xi), col="green")

# Notice that if tail fraction is included a better fit is obtained
fit2 = fnormgpdcon(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dnormgpdcon(xx, nmean = fit2$nmean, nsd = fit2$nsd, u = fit2$u,
  xi = fit2$xi), col="blue")
lines(xx, dhpdcon(xx, nmean = fitcon$nmean, nsd = fitcon$nsd, u = fitcon$u,
  xi = fitcon$xi), col="red")
abline(v = fit2$u)
```

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

`fkden` *Cross-validation MLE Fitting of Kernel Density Estimator Using normal Kernel*

Description

Maximum likelihood estimation for fitting kernel density estimator using a normal kernels, by treating it as a mixture model.

Usage

```
fkden(x, linit = NULL, extracentres = NULL,
  add.jitter = FALSE, factor = 0.1, amount = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`linit`	initial value for bandwidth parameter or `NULL`
`extracentres`	extra kernel centres used in KDE, but likelihood contribution not evaluated, or `NULL`
`add.jitter`	logical, whether jitter is needed for rounded data
`factor`	see `jitter`
`amount`	see `jitter`
`x`	vector of sample data
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`control`	optimisation control list (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`

Details

The kernel density estimator with a normal kernel is fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated bandwidth, variance and standard error are automatically output.

Missing values (`NA` and `NaN`) are assumed to be invalid data so are ignored.

Normally for likelihood estimation of the bandwidth the kernel centres and the data where the likelihood is evaluated are the same. However, when using KDE for extreme value mixture modelling the likelihood only those data in the bulk of the distribution should contribute to the likelihood, but all the data (including those beyond the threshold) should contribute to the density estimate. The `extracentres` option allows the use to specify extra kernel centres used in estimating the density, but not evaluated in the likelihood. The default is to just use the existing data, so `extracentres=NULL`.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation `finitelik=TRUE`. For invalid parameters, a zero likelihood is replaced with `exp(-1e6)`.

The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default `std.err=TRUE` and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set `std.err=FALSE`.

Value

Returns a simple list with the following elements

`call`:	`optim` call
`x`:	(jittered) data vector `x`
`kerncentres`: actual kernel centres used `x`	
`init`:	`linit`
`optim`:	complete `optim` output
`mle`:	vector of MLE of bandwidth
`cov`:	variance of MLE of bandwidth
`se`:	standard error of MLE of bandwidth
`nllh`:	minimum negative cross-validation log-likelihood
`n`:	total sample size
`lambda`:	MLE of bandwidth

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from `fpot` and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (`x` only) has been included in the fitting inputs, using the `jitter` function, to remove the ties. The default options red in the `jitter` are specified above, but the user can override these. Notice the default scaling `factor=0.1`, which is a tenth of the default value in the `jitter` function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

2) For heavy tailed populations the bandwidth is positively biased, giving oversmoothing (see example). The bias is due to the distance between the upper (or lower) order statistics not necessarily decaying to zero as the sample size tends to infinity. Essentially, as the distance between the two largest (or smallest) sample datapoints does not decay to zero, some smoothing between them is required (i.e. bandwidth cannot be zero). One solution to this problem is to splice the GPD at a

suitable threshold to remove the problematic tail from the inference for the bandwidth, using either the kdengpd function for a single heavy tail or the kdengng function if both tails are heavy. See MacDonald et al (2013).

Note

When `linit=NULL` then the initial value for the bandwidth is calculated using `bw.nrd0` function.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: http://www.math.canterbury.ac.nz/~c.scarrott.

See Also

`jitter`, `density` and `bw.nrd0`

Other kden: `dkden`, `kden`, `lkden`, `nlkden`, `pkden`, `qkden`, `rkden`

Examples

```
## Not run:
nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 7))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$lambda)*0.05)
lines(xx,dnorm(xx), col = "black")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
lines(density(x, bw = fit$lambda), lwd = 2, lty = 2,  col = "blue")
legend("topright", c("True Density", "KDE fitted evmix",
```

```
"KDE Using density, default bandwidth", "KDE Using density, c-v likelihood bandwidth"),
lty = c(1, 1, 2, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

# bandwidth is biased towards oversmoothing for heavy tails
nk=100
x = rt(nk, df = 2)
xx = seq(-8, 8, 0.01)
fit = fkden(x)
hist(x, seq(floor(min(x)), ceiling(max(x)), 0.5), freq = FALSE, xlim = c(-8, 10))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$lambda)*0.05)
lines(xx,dt(xx , df = 2), col = "black")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
legend("topright", c("True Density", "KDE fitted evmix, c-v likelihood bandwidth"),
lty = c(1, 1), lwd = c(1, 2), col = c("black", "red"))

# remove heavy tails from likelihood evaluation, but still include used in KDE within likelihood
# often gives better bandwidth
nk=100
x = rt(nk, df = 2)
xx = seq(-8, 8, 0.01)
fit2 = fkden(x[(x > -4) & (x < 4)], extracentres = x[(x <= -4) | (x >= 4)])
hist(x, seq(floor(min(x)), ceiling(max(x)), 0.5), freq = FALSE, xlim = c(-8, 10))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit2$lambda)*0.05)
lines(xx,dt(xx , df = 2), col = "black")
lines(xx, dkden(xx, x, lambda = fit2$lambda), lwd = 2, col = "red")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "blue")
legend("topright", c("True Density", "KDE fitted evmix, tails removed",
"KDE fitted evmix, tails included"),
lty = c(1, 1, 1), lwd = c(1, 2, 2), col = c("black", "red", "blue"))

## End(Not run)
```

---

fkdengpd *Cross-validation MLE Fitting of Kernel Density Estimator Using Normal Kernel and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting kernel density estimator using a normal kernels and GPD tail extreme value mixture model

Usage

```
fkdengpd(x, phiu = TRUE, pvector = NULL,
  add.jitter = FALSE, factor = 0.1, amount = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

x	vector of sample data
phiu	logical

pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
add.jitter	logical, whether jitter is needed for rounded data
factor	see jitter
amount	see jitter

Details

Extreme value mixture model combining kernel density estimation using normal kernel for the bulk below the threshold and GPD for upper tail is fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated parameters, their variance and standard error are automatically output.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD component. The default value for phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	(jittered) data vector x
kerncentres:	actual kernel centres used x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance of MLE parameters
se:	standard error of MLE parameters
nllh:	minimum negative cross-validation log-likelihood
allparams:	vector of MLE of model parameters, including phiu

allse:	vector of standard error of all parameters, including phiu
n:	total sample size
lambda:	MLE of bandwidth
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (x only) has been included in the fitting inputs, using the jitter function, to remove the ties. The default options red in the jitter are specified above, but the user can override these. Notice the default scaling factor=0.1, which is a tenth of the default value in the jitter function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

2) For heavy tailed populations the bandwidth is positively biased, giving oversmoothing (see example). The bias is due to the distance between the upper (or lower) order statistics not necessarily decaying to zero as the sample size tends to infinity. Essentially, as the distance between the two largest (or smallest) sample datapoints does not decay to zero, some smoothing between them is required (i.e. bandwidth cannot be zero). One solution to this problem is to splice the GPD at a suitable threshold to remove the problematic tail from the inference for the bandwidth, using either the kdengpdgpd function for a single heavy tail or the kdengpdgng function if both tails are heavy. See MacDonald et al (2013).

Note

When pvector=NULL then the initial value for the parameters are calculated type fkdengpdcon to see how.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: http://www.math.canterbury.ac.nz/~c.scarrott.

See Also

fkden, jitter, density and bw.nrd0

Other kdengpd: dkdengpd, kdengpd, pkdengpd, qkdengpd, rkdengpd

Examples

```
## Not run:
x = rnorm(1000, 0, 1)
fit = fkdengpd(x, phiu = FALSE, std.err = FALSE)
hist(x, 100, freq = FALSE, xlim = c(-5, 5))
xx = seq(-5, 5, 0.01)
lines(xx, dkdengpd(xx, x, fit$lambda, fit$u, fit$sigmau, fit$xi, fit$phiu), col="blue")
abline(v = fit$u)

## End(Not run)
```

---

fkdengpdcon *Cross-validation MLE Fitting of Kernel Density Estimator Using Normal Kernel and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

Description

Maximum likelihood estimation for fitting kernel density estimator using a normal kernels and GPD tail extreme value mixture model and continuous at threshold.

Usage

```
fkdengpdcon(x, phiu = TRUE, pvector = NULL,
  add.jitter = FALSE, factor = 0.1, amount = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

x	vector of sample data
phiu	logical
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
add.jitter	logical, whether jitter is needed for rounded data
factor	see jitter
amount	see jitter
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

Extreme value mixture model combining kernel density estimation using normal kernel for the bulk below the threshold and GPD for upper tail, with a constraint to be continuous at the threshold is fitted to the entire dataset using maximum cross-validation likelihood estimation. The estimated parameters, their variance and standard error are automatically output.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD component.

The default value for phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance (from inverse hessian) and standard error of bandwidth parameter cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the bandwidth estimate even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	(jittered) data vector x
kerncentres: actual kernel centres used x	
init:	pvector

optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance of MLE parameters
se:	standard error of MLE parameters
nllh:	minimum negative cross-validation log-likelihood
allparams:	vector of MLE of model parameters, including phiu and sigmau
allse:	vector of standard error of all parameters, including phiu and sigmau
n:	total sample size
lambda:	MLE of bandwidth
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Warning

Two important practical issues arise with MLE for the kernel bandwidth: 1) Cross-validation likelihood is needed for the KDE bandwidth parameter as the usual likelihood degenerates, so that the MLE $\hat{\lambda} \to 0$ as $n \to \infty$, thus giving a negative bias towards a small bandwidth. Leave one out cross-validation essentially ensures that some smoothing between the kernel centres is required (i.e. a non-zero bandwidth), otherwise the resultant density estimates would always be zero if the bandwidth was zero.

This problem occassionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (x only) has been included in the fitting inputs, using the jitter function, to remove the ties. The default options red in the jitter are specified above, but the user can override these. Notice the default scaling factor=0.1, which is a tenth of the default value in the jitter function itself.

A warning message is given if the data appear to be rounded (i.e. more than 5 estimated bandwidth is too small, then data rounding is the likely culprit. Only use the jittering when the MLE of the bandwidth is far too small.

2) For heavy tailed populations the bandwidth is positively biased, giving oversmoothing (see example). The bias is due to the distance between the upper (or lower) order statistics not necessarily decaying to zero as the sample size tends to infinity. Essentially, as the distance between the two largest (or smallest) sample datapoints does not decay to zero, some smoothing between them is required (i.e. bandwidth cannot be zero). One solution to this problem is to splice the GPD at a suitable threshold to remove the problematic tail from the inference for the bandwidth, using either the kdengpdgpd function for a single heavy tail or the kdengpdgng function if both tails are heavy. See MacDonald et al (2013).

Note

When pvector=NULL then the initial value for the parameters are calculated type fkdengpdcon to see how.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

MacDonald, A., C. J. Scarrott, and D. S. Lee (2011). Boundary correction, consistency and robustness of kernel densities using extreme value theory. Submitted. Available from: http://www.math.canterbury.ac.nz/~c.scarrott.

See Also

fkdengpd, fkden, jitter, density and bw.nrd0

Other kdengpdcon: dkdengpdcon, kdengpdcon, lkdengpdcon, nlkdengpdcon, pkdengpdcon, qkdengpdcon, rkdengpdcon

Examples

```
## Not run:
x = rnorm(1000, 0, 1)
fit = fkdengpdcon(x, phiu = FALSE, std.err = FALSE)
hist(x, 100, freq = FALSE, xlim = c(-5, 5))
xx = seq(-5, 5, 0.01)
lines(xx, dkdengpdcon(xx, x, fit$lambda, fit$u, fit$xi, fit$phiu), col="blue")
abline(v = fit$u)

## End(Not run)
```

---

flognormgpd *MLE Fitting of Log-Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold

Usage

```
flognormgpd(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values of mixture model parameters (lnmean, lnsd, u, sigmau, xi) or NULL
x	vector of sample data
phiu	logical
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with log-normal bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Negative data are ignored.

The default value for phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters

cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu
allse:	vector of standard error of all parameters and phiu
n:	total sample size
nmean:	MLE of log-normal mean
nsd:	MLE of log-normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type flognormgpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

lgpd and gpd

Other lognormgpd: dlognormgpd, llognormgpd, lognormgpd, nllognormgpd, plognormgpd, qlognormgpd, rlognormgpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rlnorm(1000)
xx = seq(-1, 6, 0.01)
y = dlnorm(xx)

# Bulk model base tail fraction
fit = flognormgpd(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, y)
lines(xx, dlognormgpd(xx, lnmean = fit$lnmean, lnsd = fit$lnsd, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = flognormgpd(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dlognormgpd(xx, lnmean = fit$lnmean, lnsd = fit$lnsd, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dlognormgpd(xx, lnmean = fit2$lnmean, lnsd = fit2$lnsd, u = fit2$u,
  sigmau = fit2$sigmau, xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`flognormgpdcon` *MLE Fitting of Log-Normal Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
flognormgpdcon(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`pvector`	vector of initial values of mixture model parameters (`lnmean`, `lnsd`, `u`, `xi`) or `NULL`
`x`	vector of sample data
`phiu`	logical
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)

control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with log-normal bulk and GPD tail with continuity constraint is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Negative data are ignored.

The default value for phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including sigmau and phiu
allse:	vector of standard error of all parameters, including sigmau and phiu
n:	total sample size
nmean:	MLE of log-normal mean
nsd:	MLE of log-normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type flognormgpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

flognormgpd, llognormgpd, lgpd and gpd

Other lognormgpdcon: dlognormgpdcon, llognormgpdcon, lognormgpdcon, nllognormgpdcon, plognormgpdcon, qlognormgpdcon, rlognormgpdcon

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rlnorm(1000)
xx = seq(-1, 6, 0.01)
y = dlnorm(xx)

# Bulk model base tail fraction
fit = flognormgpdcon(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, y)
lines(xx, dlognormgpdcon(xx, lnmean = fit$lnmean, lnsd = fit$lnsd, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
```

```
fit2 = flognormgpdcon(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dlognormgpdcon(xx, lnmean = fit$lnmean, lnsd = fit$lnsd, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dlognormgpdcon(xx, lnmean = fit2$lnmean, lnsd = fit2$lnsd, u = fit2$u,
  xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fnormgpd` *MLE Fitting of Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold

Usage

```
fnormgpd(x, phiu = TRUE, pvector = NULL, std.err = TRUE,
  method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)
```

Arguments

`pvector`	vector of initial values of mixture model parameters (`nmean`, `nsd`, `u`, `sigmau`, `xi`) or `NULL`
`phiu`	logical
`control`	optimisation control list (see `optim`)
`x`	vector of sample data
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`

Details

The extreme value mixture model with normal bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

The default value for `phiu=TRUE` so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When `phiu=FALSE` then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for `phiu` is estimated and output as `sephiu`.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu
allse:	vector of standard error of all parameters and phiu
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fnormgpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other normgpd: dnormgpd, lnormgpd, nlnormgpd, normgpd, pnormgpd, qnormgpd, rnormgpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model base tail fraction
fit = fnormgpd(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fnormgpd(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dnormgpd(xx, nmean = fit$nmean, nsd = fit$nsd, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dnormgpd(xx, nmean = fit2$nmean, nsd = fit2$nsd, u = fit2$u,
  sigmau = fit2$sigmau, xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

fnormgpdcon *MLE Fitting of Normal Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
fnormgpdcon(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values mixture model parameters (nmean, nsd, u, xi) or NULL
x	vector of sample data
phiu	logical
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with normal bulk and GPD tail with a continuity constraint is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

The default value for phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters, including phiu and sigmau
allse:	vector of standard error of all parameters, including phiu and sigmau
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fnormgpdcon to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other normgpdcon: dnormgpdcon, lnormgpdcon, nlnormgpdcon, normgpdcon, pnormgpdcon, qnormgpdcon, rnormgpdcon

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model base tail fraction
fit = fnormgpdcon(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
lines(xx, dnormgpdcon(xx, nmean = fit$nmean, nsd = fit$nsd, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fnormgpdcon(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dnormgpdcon(xx, nmean = fit$nmean, nsd = fit$nsd, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dnormgpdcon(xx, nmean = fit2$nmean, nsd = fit2$nsd, u = fit2$u,
  xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

fweibullgpd *MLE Fitting of Weibull Bulk and GPD Tail Extreme Value Mixture Model*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold

Usage

```
fweibullgpd(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

pvector	vector of initial values mixture model parameters (wshape, wscale, u, sigmau, xi) or NULL
x	vector of sample data
phiu	logical
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with Weibull bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Non-positive data are ignored.

The default value for phiu=TRUE so that the tail fraction is specified by Weibull distribution $\phi_u = 1 - H(u)$. When phiu=FALSE then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for phiu is estimated and output as sephiu.

Missing values (NA and NaN) are assumed to be invalid data so are ignored, which is inconsistent with the evd library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with exp(-1e6). The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for finitelik will be overridden and set to finitelik=TRUE if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from optim function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default std.err=TRUE and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set std.err=FALSE.

Value

Returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
optim:	complete optim output
mle:	vector of MLE of model parameters
cov:	variance-covariance matrix of MLE of model parameters
se:	vector of standard errors of MLE of model parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
allparams:	vector of MLE of model parameters and phiu

allse:	vector of standard error of all parameters and phiu
n:	total sample size
wshape:	MLE of Weibull shape
wscale:	MLE of Weibull scale
u:	threshold
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fweibullgpd to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other weibullgpd: dweibullgpd, lweibullgpd, nlweibullgpd, pweibullgpd, qweibullgpd, rweibullgpd, weibullgpd

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rweibull(1000, shape = 2)
xx = seq(-1, 4, 0.01)
```

```
y = dweibull(xx, shape = 2)

# Bulk model base tail fraction
fit = fweibullgpd(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 4))
lines(xx, y)
lines(xx, dweibullgpd(xx, wshape = fit$wshape, wscale = fit$wscale, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fweibullgpd(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dweibullgpd(xx, wshape = fit$wshape, wscale = fit$wscale, u = fit$u,
  sigmau = fit$sigmau, xi = fit$xi, phiu = TRUE), col="red")
lines(xx, dweibullgpd(xx, wshape = fit2$wshape, wscale = fit2$wscale, u = fit2$u,
  sigmau = fit2$sigmau, xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fweibullgpdcon` *MLE Fitting of Weibull Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Maximum likelihood estimation for fitting the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
fweibullgpdcon(x, phiu = TRUE, pvector = NULL,
  std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)
```

Arguments

`pvector`	vector of initial values mixture model parameters (`wshape`, `wscale`, `u`, `sigmau`, `xi`) or `NULL`
`x`	vector of sample data
`phiu`	logical
`std.err`	logical, should standard errors be calculated
`method`	optimisation method (see `optim`)
`control`	optimisation control list (see `optim`)
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`...`	optional inputs passed to `optim`

Details

The extreme value mixture model with Weibull bulk and GPD tail is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Non-positive data are ignored.

The default value for `phiu=TRUE` so that the tail fraction is specified by Weibull distribution $\phi_u = 1 - H(u)$. When `phiu=FALSE` then the tail fraction is treated as an extra parameter estimated using the MLE which is the sample proportion above the threshold. In this case the standard error for `phiu` is estimated and output as `sephiu`.

Missing values (`NA` and `NaN`) are assumed to be invalid data so are ignored, which is inconsistent with the `evd` library which assumes the missing values are below the threshold.

The default optimisation algorithm is "BFGS", which requires a finite negative log-likelihood function evaluation `finitelik=TRUE`. For invalid parameters, a zero likelihood is replaced with `exp(-1e6)`. The "BFGS" optimisation algorithms require finite values for likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call.

If the hessian is of reduced rank then the variance covariance (from inverse hessian) and standard error of parameters cannot be calculated, then by default `std.err=TRUE` and the function will stop. If you want the parameter estimates even if the hessian is of reduced rank (e.g. in a simulation study) then set `std.err=FALSE`.

Value

Returns a simple list with the following elements

`call`:	`optim` call
`x`:	data vector `x`
`init`:	`pvector`
`optim`:	complete `optim` output
`mle`:	vector of MLE of model parameters
`cov`:	variance-covariance matrix of MLE of model parameters
`se`:	vector of standard errors of MLE of model parameters
`rate`:	`phiu` to be consistent with `evd`
`nllh`:	minimum negative log-likelihood
`allparams`:	vector of MLE of model parameters, including `sigmau` and `phiu`
`allse`:	vector of standard error of all parameters, including `sigmau` and `phiu`
`n`:	total sample size
`wshape`:	MLE of Weibull shape
`wscale`:	MLE of Weibull scale
`u`:	threshold
`sigmau`:	MLE of GPD scale
`xi`:	MLE of GPD shape
`phiu`:	MLE of tail fraction

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from `fpot` and to make it as useable as possible.

Note

Unlike all the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu. Only the data is a vector.

When pvector=NULL then the initial values are calculated, type fweibullgpdcon to see the default formulae used. The mixture model fitting can be ***extremely*** sensitive to the initial values, so you if you get a poor fit then try some alternatives. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The fitting function will stop if infinite sample values are given.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other weibullgpdcon: dweibullgpdcon, lweibullgpdcon, nlweibullgpdcon, pweibullgpdcon, qweibullgpdcon, rweibullgpdcon, weibullgpdcon

Examples

```
## Not run:
par(mfrow=c(2,1))
x = rweibull(1000, shape = 2)
xx = seq(-1, 4, 0.01)
y = dweibull(xx, shape = 2)

# Bulk model base tail fraction
fit = fweibullgpdcon(x, phiu = TRUE, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 4))
lines(xx, y)
lines(xx, dweibullgpdcon(xx, wshape = fit$wshape, wscale = fit$wscale, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
abline(v = fit$u)

# Parameterised tail fraction
fit2 = fweibullgpdcon(x, phiu = FALSE, std.err = FALSE)
plot(xx, y, type = "l")
lines(xx, dweibullgpdcon(xx, wshape = fit$wshape, wscale = fit$wscale, u = fit$u,
  xi = fit$xi, phiu = TRUE), col="red")
```

```
lines(xx, dweibullgpdcon(xx, wshape = fit2$wshape, wscale = fit2$wscale, u = fit2$u,
  xi = fit2$xi, phiu = fit2$phiu), col="blue")
abline(v = fit$u, col = "red")
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gammagpd *Gamma Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the gamma shape `gshape` and scale `gscale`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dgammagpd(x, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale),
  sigmau = sqrt(gshape) * gscale, xi = 0, phiu = TRUE,
  log = FALSE)

pgammagpd(q, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale),
  sigmau = sqrt(gshape) * gscale, xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qgammagpd(p, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale),
  sigmau = sqrt(gshape) * gscale, xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rgammagpd(n = 1, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale),
  sigmau = sqrt(gshape) * gscale, xi = 0, phiu = TRUE)
```

Arguments

`gshape`	gamma shape (non-negative)
`gscale`	gamma scale (non-negative)
`u`	threshold (non-negative)
`phiu`	probability of being above threshold [0,1] or TRUE
`x`	quantile
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density

q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining gamma distribution for the bulk below the threshold and GPD for upper tail. The user can pre-specify phiu permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the gamma bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the gamma bulk model (phiu=TRUE), upto the threshold $0 < x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the gamma and conditional GPD cumulative distribution functions (i.e. pgamma(x, gshape, scale = gscale) and pgpd(x, u, sigmau, xi)).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 < x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The gamma is defined on the non-negative reals, so the threshold must be non-negative.

See gpd for details of GPD upper tail component and dgamma for details of gamma bulk component.

Value

dgammagpd gives the density, pgammagpd gives the cumulative distribution function, qgammagpd gives the quantile function and rgammagpd gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main inputs (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rgammagpd any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default sample size for rgammagpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Gamma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd and dgamma

Other gammagpd: fgammagpd, lgammagpd, nlgammagpd

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgammagpd(1000, gshape = 2)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammagpd(xx, gshape = 2))

# three tail behaviours
plot(xx, pgammagpd(xx, gshape = 2), type = "l")
lines(xx, pgammagpd(xx, gshape = 2, xi = 0.3), col = "red")
lines(xx, pgammagpd(xx, gshape = 2, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rgammagpd(1000, gshape = 2, u = 3, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammagpd(xx, gshape = 2, u = 3, phiu = 0.2))

plot(xx, dgammagpd(xx, gshape = 2, u = 3, xi=0, phiu = 0.2), type = "l")
lines(xx, dgammagpd(xx, gshape = 2, u = 3, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dgammagpd(xx, gshape = 2, u = 3, xi=0.2, phiu = 0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gammagpdcon *Gamma Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint. The parameters are the gamma shape `gshape` and scale `gscale`, threshold `u` and GPD shape `xi` and tail fraction `phiu`.

Usage

```
dgammagpdcon(x, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale), xi = 0, phiu = TRUE,
  log = FALSE)

pgammagpdcon(q, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qgammagpdcon(p, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rgammagpdcon(n = 1, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale), xi = 0, phiu = TRUE)
```

Arguments

`x`	quantile
`gshape`	gamma shape (non-negative)
`gscale`	gamma scale (non-negative)
`u`	threshold (non-negative)
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or TRUE
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining gamma distribution for the bulk below the threshold and GPD for upper tail with a continuity constraint. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the gamma bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the gamma bulk model (`phiu=TRUE`), upto the threshold $0 < x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(x)$ are the gamma and conditional GPD cumulative distribution functions (i.e. `pgamma(x, gshape, scale = gscale)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 < x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the gamma and conditional GPD density functions (i.e. `dgamma(x, gshape, scale = gscale)` and `dgpd(x, u, sigmau, xi)`). The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u)/[1 - \phi_u]h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)]/h(u)$$

.

The gamma is defined on the non-negative reals, so the threshold must be non-negative.

See gpd for details of GPD upper tail component and dgamma for details of gamma bulk component.

Value

dgammagpdcon gives the density, pgammagpdcon gives the cumulative distribution function, qgammagpdcon gives the quantile function and rgammagpdcon gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rgammagpdcon` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for rgammagpdcon is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Gamma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd, dgamma and dgammagpd

Other gammagpdcon: fgammagpdcon, lgammagpdcon, nlgammagpdcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgammagpdcon(1000, gshape = 2)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammagpdcon(xx, gshape = 2))

# three tail behaviours
plot(xx, pgammagpdcon(xx, gshape = 2), type = "l")
lines(xx, pgammagpdcon(xx, gshape = 2, xi = 0.3), col = "red")
lines(xx, pgammagpdcon(xx, gshape = 2, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rgammagpdcon(1000, gshape = 2, u = 3, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammagpdcon(xx, gshape = 2, u = 3, phiu = 0.2))

plot(xx, dgammagpdcon(xx, gshape = 2, u = 3, xi=0, phiu = 0.2), type = "l")
lines(xx, dgammagpdcon(xx, gshape = 2, u = 3, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dgammagpdcon(xx, gshape = 2, u = 3, xi=0.2, phiu = 0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gkg *Kernel Density Estimation for Bulk and GPD for Both Upper and Lower Tails in Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with kernel density estimation using normal kernel for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails. The parameters are the kernel bandwidth `lambda`, lower tail (threshold `ul`, GPD scale `sigmaul` and shape `xil`

and tail fraction phiul) and upper tail (threshold ur, GPD scale sigmaur and shape xiR and tail fraction phiur).

Usage

```
dgkg(x, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)),
  sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0,
  phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)),
  sigmaur = sqrt(6 * var(kerncentres))/pi, xir = 0,
  phiur = TRUE, log = FALSE)

pgkg(q, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)),
  sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0,
  phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)),
  sigmaur = sqrt(6 * var(kerncentres))/pi, xir = 0,
  phiur = TRUE, lower.tail = TRUE)

qgkg(p, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)),
  sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0,
  phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)),
  sigmaur = sqrt(6 * var(kerncentres))/pi, xir = 0,
  phiur = TRUE, lower.tail = TRUE)

rgkg(n = 1, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)),
  sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0,
  phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)),
  sigmaur = sqrt(6 * var(kerncentres))/pi, xir = 0,
  phiur = TRUE)
```

Arguments

x	quantile
kerncentres	kernel centres (typically sample data)
lambda	bandwidth for normal kernel (standard deviation of normal)
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (non-negative)
xil	lower tail GPD shape parameter

phiul	probability of being below lower threshold (0,1)
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (non-negative)
xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold (0,1)

Details

Extreme value mixture model combining kernel density estimator (KDE) with normal kernels to represent the bulk between the lower and upper thresholds and GPD for the upper and lower tails. The user can pre-specify phiul and phiur permitting a parameterised value for the lower and upper tail fraction respectively. Alternatively, when phiul=TRUE or phiur=TRUE the corresponding tail fraction is estimated as from the normal bulk model.

Notice that the tail fraction cannot be 0 or 1, and the sum of upper and lower tail fractions phiul + phiur < 1, so the lower threshold must be less than the upper, ul < ur.

The cumulative distribution function has three components. The lower tail with tail fraction ϕ_{ul} defined by the KDE bulk model (phiul=TRUE) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)[1 - G_l(x)].$$

where $H(x)$ is the kernel density estimator cumulative distribution function (i.e. mean(pnorm(x, kerncentres, lambda and $G_l(X)$ is the conditional GPD cumulative distribution function with negated x value and threshold, i.e. pgpd(-x, -ul, sigmaul, xil, phiul). The KDE bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = H(x).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = H(u_r) + [1 - H(u_r)]G_r(x)$$

where $G_r(X)$ is the GPD cumulative distribution function, i.e. pgpd(x, ur, sigmaur, xir, phiur).

The cumulative distribution function for the pre-specified tail fractions ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}[1 - G_l(x)].$$

The KDE bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

See gpd for details of GPD upper tail component, dkden for details of KDE bulk component and dkdengpd for KDE with sinlge upper tail GPD extreme value mixture model.

Value

dgkg gives the density, pgkg gives the cumulative distribution function, qgkg gives the quantile function and rgkg gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main input (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rgkg` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rgkg` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

See Also

Other gkg: `fgkg`, `lgkg`, `nlgkg`

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(1000,0,1)
x = rgkg(1000, kerncentres, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkg(xx, kerncentres, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgkg(xx, kerncentres), type = "l")
lines(xx, pgkg(xx, kerncentres,xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgkg(xx, kerncentres,xil = -0.3, xir = -0.3), col = "blue")
legend("topleft", paste("Symmetric xil=xir=",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rgkg(1000, kerncentres, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkg(xx, kerncentres, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2))

plot(xx, dgkg(xx, kerncentres, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2),
  type = "l", ylim = c(0, 0.4))
lines(xx, dgkg(xx, kerncentres, xil = -0.3, phiul = 0.3, xir = 0.3, phiur = 0.3),
```

```
  col = "red")
lines(xx, dgkg(xx, kerncentres, xil = -0.3, phiul = TRUE, xir = 0.3, phiur = TRUE),
  col = "blue")
legend("topleft", c("phiul = phiur = 0.2", "phiul = phiur = 0.3", "Bulk Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gng *Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails. The parameters are the normal mean `nmean` and standard deviation `nsd`, lower tail (threshold `ul`, GPD scale `sigmaul` and shape `xil` and tail fraction `phiul`) and upper tail (threshold `ur`, GPD scale `sigmaur` and shape `xiR` and tail fraction `phiuR`).

Usage

```
dgng(x, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), sigmaur = nsd, xir = 0,
  phiur = TRUE, log = FALSE)

pgng(q, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), sigmaur = nsd, xir = 0,
  phiur = TRUE, lower.tail = TRUE)

qgng(p, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), sigmaur = nsd, xir = 0,
  phiur = TRUE, lower.tail = TRUE)

rgng(n = 1, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), sigmaul = nsd, xil = 0,
  phiul = TRUE, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, phiur = TRUE)
```

Arguments

`ul`	lower tail threshold
`sigmaul`	lower tail GPD scale parameter (non-negative)
`xil`	lower tail GPD shape parameter
`phiul`	probability of being below lower threshold (0,1)
`ur`	upper tail threshold
`sigmaur`	upper tail GPD scale parameter (non-negative)

xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold (0,1)
x	quantile
nmean	normal mean
nsd	normal standard deviation (non-negative)
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk between the lower and upper thresholds and GPD for upper and lower tails. The user can pre-specify `phiul` and `phiur` permitting a parameterised value for the lower and upper tail fraction respectively. Alternatively, when `phiul=TRUE` or `phiur=TRUE` the corresponding tail fraction is estimated as from the normal bulk model.

Notice that the tail fraction cannot be 0 or 1, and the sum of upper and lower tail fractions `phiul+phiur<1`, so the lower threshold must be less than the upper, `ul<ur`.

The cumulative distribution function now has three components. The lower tail with tail fraction ϕ_{ul} defined by the normal bulk model (`phiul=TRUE`) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)G_l(x).$$

where $H(x)$ is the normal cumulative distribution function (i.e. `pnorm(ur, nmean, nsd)`). The $G_l(X)$ is the conditional GPD cumulative distribution function with negated data and threshold, i.e. `dgpd(-x, -ul, sigmaul, xil, phiul)`. The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = H(x).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = H(u_r) + [1 - H(u_r)]G(x)$$

where $G(X)$.

The cumulative distribution function for the pre-specified tail fractions ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}G_l(x).$$

The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

See gpd for details of GPD upper tail component, dnorm for details of normal bulk component and dnormgpd for normal with GPD extreme value mixture model.

Value

dgng gives the density, pgng gives the cumulative distribution function, qgng gives the quantile function and rgng gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main input (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rgng any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default sample size for rgng is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

normgpd, gpd and dnorm

Other gng: fgng, lgng, nlgng

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgng(1000, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgng(xx, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgng(xx), type = "l")
lines(xx, pgng(xx, xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgng(xx, xil = -0.3, xir = -0.3), col = "blue")
legend("topleft", paste("Symmetric xil=xir=",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)
```

```
x = rgng(1000, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgng(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2))

plot(xx, dgng(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2), type = "l", ylim = c(0, 0.4))
lines(xx, dgng(xx, xil = -0.3, phiul = 0.3, xir = 0.3, phiur = 0.3), col = "red")
lines(xx, dgng(xx, xil = -0.3, phiul = TRUE, xir = 0.3, phiur = TRUE), col = "blue")
legend("topleft", c("phiul = phiur = 0.2", "phiul = phiur = 0.3", "Bulk Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gngcon *Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model with Continuity Constraints*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails with Continuity Constraints at the lower and upper threshold. The parameters are the normal mean `nmean` and standard deviation `nsd`, lower tail (threshold `ul`, GPD shape `xil` and tail fraction `phiul`) and upper tail (threshold `ur`, GPD shape `xiR` and tail fraction `phiuR`).

Usage

```
dgngcon(x, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
  log = FALSE)

pgngcon(q, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
  lower.tail = TRUE)

qgngcon(p, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
  lower.tail = TRUE)

rgngcon(n = 1, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE)
```

Arguments

`x`	quantile
`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)

ul	lower tail threshold
xil	lower tail GPD shape parameter
phiul	probability of being below lower threshold (0,1)
ur	upper tail threshold
xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold (0,1)
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk between the lower and upper thresholds and GPD for upper and lower tails with Continuity Constraints at the lower and upper threshold. The user can pre-specify `phiul` and `phiur` permitting a parameterised value for the lower and upper tail fraction respectively. Alternatively, when `phiul=TRUE` or `phiur=TRUE` the corresponding tail fraction is estimated as from the normal bulk model.

Notice that the tail fraction cannot be 0 or 1, and the sum of upper and lower tail fractions `phiul+phiur<1`, so the lower threshold must be less than the upper, `ul<ur`.

The cumulative distribution function now has three components. The lower tail with tail fraction ϕ_{ul} defined by the normal bulk model (`phiul=TRUE`) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)G_l(x).$$

where $H(x)$ is the normal cumulative distribution function (i.e. `pnorm(ur, nmean, nsd)`). The $G_l(X)$ is the conditional GPD cumulative distribution function with negated data and threshold, i.e. `dgpd(-x, -ul, sigmaul, xil, phiul)`. The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = H(x).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = H(u_r) + [1 - H(u_r)]G(x)$$

where $G(X)$.

The cumulative distribution function for the pre-specified tail fractions ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}G_l(x).$$

The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

The continuity constraint at the `ul` means that:

$$\phi_{ul} g_l(x) = (1 - \phi_{ul} - \phi_{ur}) h(u_l) / (H(u_r) - H(u_l)).$$

The GPD scale parameter `sigmaul` is replaced by:

$$\sigma_u l = \phi_{ul} (H(u_r) - H(u_l)) / h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u l = H(u_l) / h(u_l)$$

.

The continuity Constraint at the `ur` means that:

$$\phi_{ur} g_r(x) = (1 - \phi_{ul} - \phi_{ur}) h(u_l) / (H(u_r) - H(u_l)).$$

By rearrangement, the GPD scale parameter `sigmaur` is then:

$$\sigma_u r = \phi_{ur} (H(u_r) - H(u_l)) / h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

where $h(x)$, $g_l(x)$ and $g_r(x)$ are the normal and conditional GPD density functions for lower and upper tail (i.e. `dnorm(x, nmean, nsd)`, `dgpd(-x, -ul, sigmaul, xil, phiul)`, and `dgpd(x, ur, sigmaur, xir`
In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u r = [1 - H(u_r)] / h(u_r)$$

.

See `gpd` for details of GPD upper tail component, `dnorm` for details of normal bulk component, `dnormgpd` for normal with GPD extreme value mixture model and `dgng` for normal bulk with GPD upper and lower tails extreme value mixture model.

Value

`dgngcon` gives the density, `pgngcon` gives the cumulative distribution function, `qgngcon` gives the quantile function and `rgngcon` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rgngcon` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rgngcon` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

gng, normgpd, gpd and dnorm

Other gngcon: fgngcon, lgngcon, nlgngcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgngcon(1000, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgngcon(xx, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgngcon(xx), type = "l")
lines(xx, pgngcon(xx, xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgngcon(xx, xil = -0.3, xir = -0.3), col = "blue")
legend("topleft", paste("Symmetric xil=xir=",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rgngcon(1000, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgngcon(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2))

plot(xx, dgngcon(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2), type = "l", ylim = c(0, 0.4))
lines(xx, dgngcon(xx, xil = -0.3, phiul = 0.3, xir = 0.3, phiur = 0.3), col = "red")
lines(xx, dgngcon(xx, xil = -0.3, phiul = TRUE, xir = 0.3, phiur = TRUE), col = "blue")
legend("topleft", c("phiul = phiur = 0.2", "phiul = phiur = 0.3", "Bulk Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

gpd *Generalised Pareto Distribution (GPD)*

Description

Density, cumulative distribution function, quantile function and random number generation for the GPD conditional on being above a threshold `u` with parameters `sigmau` and `xi`. Unconditional quantities are provided when the probability `phiu` of being above the threshold `u` is given.

Usage

```
dgpd(x, u = 0, sigmau = 1, xi = 0, phiu = 1, log = FALSE)

pgpd(q, u = 0, sigmau = 1, xi = 0, phiu = 1,
  lower.tail = TRUE)

qgpd(p, u = 0, sigmau = 1, xi = 0, phiu = 1,
  lower.tail = TRUE)

rgpd(n = 1, u = 0, sigmau = 1, xi = 0, phiu = 1)
```

Arguments

x	quantile
q	quantile
p	cumulative probability
n	sample size (non-negative integer)
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1]
log	logical, if TRUE then log density
lower.tail	logical, if FALSE then upper tail probabilities

Details

The GPD with parameters scale σ_u and shape ξ has conditional density given by

$$f(x|X > u) = 1/\sigma_u[1 + \xi(x - u)/\sigma_u]^{-1/\xi - 1}$$

for non-zero ξ, $x > u$ and $\sigma_u > 0$. Further, $[1 + \xi(x - u)/\sigma_u] > 0$ which for $\xi < 0$ implies $u < x \leq u - \sigma_u/\xi$. In the special case of $\xi = 0$, which is treated as $|\xi| < 1e - 6$, it reduces to the exponential:

$$f(x|X > u) = 1/\sigma_u exp(-(x - u)/\sigma_u).$$

The unconditional density is obtained by mutltiplying this by the survival probability (or *tail fraction*) $\phi_u = P(X > u)$ giving $f(x) = \phi_u f(x|X > u)$.

The syntax of these functions are similar to those of the evd package, so most code using these functions can simply be reused. The key difference is the introduction of phiu to permit output of unconditional quantities.

Value

dgpd gives the density, pgpd gives the cumulative distribution function, qgpd gives the quantile function and rgpd gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main inputs (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rgpd any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default threshold u=0 and tail fraction phiu=1 which assumes the user will default to inputting excesses above u, rather than exceedance. The default sample size for rgpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Some key differences arise for phiu=1 and phiu<1 (see examples below):

1. For phiu=1 the dgpd evaluates as zero for quantiles below the threshold u and pgpd evaluates over $[0, 1]$.
2. For phiu=1 then pgpd evaluates as zero below the threshold u. For phiu<1 it evaluates as $1 - \phi_u$ at the threshold and NA below the threshold.
3. For phiu=1 the quantiles from qgpd are above threshold and equal to threshold for phiu=0. For phiu<1 then within upper tail, p > 1 - phiu, it will give conditional quantiles above threshold, but when below the threshold, p <= 1 - phiu, these are set to NA.
4. When simulating GPD variates using rgpd if phiu=1 then all values are above the threshold. For phiu<1 then a standard uniform U is simulated and the variate will be classified as above the threshold if $U < \phi$, and below the threshold otherwise. This is equivalent to a binomial random variable for simulated number of exceedances. Those above the threshold are then simulated from the conditional GPD and those below the threshold and set to NA.

These conditions are intuitive and consistent with evd, which assumes missing data are below threshold.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution Based on GPD functions in the evd package.

See Also

evd and fpot

Other gpd: fgpd, lgpd, nlgpd

Examples

```
par(mfrow=c(2,2))
x = rgpd(1000) # simulate sample from GPD
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgpd(xx))
```

```
# three tail behaviours
plot(xx, pgpd(xx), type = "l")
lines(xx, pgpd(xx, xi = 0.3), col = "red")
lines(xx, pgpd(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

# GPD when xi=0 is exponential, and demonstrating phiu
x = rexp(1000)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgpd(xx, u = 0, sigmau = 1, xi = 0), lwd = 2)
lines(xx, dgpd(xx, u = 0.5, phiu = 1 - pexp(0.5)), col = "red", lwd = 2)
lines(xx, dgpd(xx, u = 1.5, phiu = 1 - pexp(1.5)), col = "blue", lwd = 2)
legend("topright", paste("u =",c(0, 0.5, 1.5)),
  col=c("black", "red", "blue"), lty = 1, lwd = 2)

# Quantile function and phiu
p = pgpd(xx)
plot(qgpd(p), p, type = "l")
lines(xx, pgpd(xx, u = 2), col = "red")
lines(xx, pgpd(xx, u = 5, phiu = 0.2), col = "blue")
legend("bottomright", c("u = 0 phiu = 1","u = 2 phiu = 1","u = 5 phiu = 0.2"),
  col=c("black", "red", "blue"), lty = 1)
```

---

hpd *Hybrid Pareto Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the hybrid Pareto extreme value mixture model. The parameters are the normal mean `nmean` and standard deviation `nsd` and GPD shape `xi`.

Usage

```
dhpd(x, nmean = 0, nsd = 1, xi = 0, log = FALSE)

phpd(q, nmean = 0, nsd = 1, xi = 0, lower.tail = TRUE)

qhpd(p, nmean = 0, nsd = 1, xi = 0, lower.tail = TRUE)

rhpd(n = 1, nmean = 0, nsd = 1, xi = 0)
```

Arguments

`x`	quantile
`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density
`q`	quantile

lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail which is continuous in its zeroth and first derivative at the threshold, but with one important difference to all the other mixture models.

The hybrid Pareto does not include the usual tail fraction phiu scaling, i.e. so the GPD is not treated as a conditional model for the exceedances. The unscaled GPD is simply spliced with the normal truncated at the threshold, with no rescaling to account for the proportion above the threshold being applied. The parameters have to adjust for the lack of tail fraction scaling.

The two continuity constraints lead to the threshold u and GPD scale sigmau being replaced by a function of the normal mean, standard deviation and GPD shape parameters.

The continuity constraint on its first derivative at the threshold means that $h'(u) = g'(u)$. Then the Lambert W function is used for replacing the threshold u and GPD scale sigmau in terms of the normal mean, standard deviation and GPD shape xi. The cumulative distribution function defined upto the threshold $x \leq u$, given by:

$$F(x) = H(x)/r$$

and above the threshold $x > u$:

$$F(x) = (H(u) + G(x))/r$$

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. pnorm(x, nmean, nsd), pgpd(x, u, sigmau, xi)). The normalisation constant r ensures a proper density and is given byr = 1 + pnorm(u, mean = nmean, sd = nsd), i.e. the 1 comes from integration of the unscaled GPD and the second term is from the usual normal component.

The continuity constraint on the density at the threshold means that $h(u) = g(u)$ where $h(x)$ and $g(x)$ are the normal and unscaled GPD density functions (i.e. dnorm(u, nmean, nsd) and dgpd(u, u, sigmau, xi)).

See gpd for details of GPD upper tail component and dnorm for details of normal bulk component.

Value

dhpd gives the density, phpd gives the cumulative distribution function, qhpd gives the quantile function and rhpd gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main inputs (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rhpdcon any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default sample size for rhpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

gpd, dnorm, dnormgpd and dnormgpdcon. The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the hybrid Pareto hpareto and mixture of hybrid Paretos hparetomixt, which are more flexible as they also permit the model to be truncated at zero.

Other hpd: fhpd, lhpd, nlhpd

Examples

```
## Not run:
par(mfrow = c(2, 2))
xx = seq(-5, 20, 0.01)
f1 = dhpd(xx, 0,1, 0.4)
plot(xx, f1, type = "l")
abline(v = 0.4942921)

# three tail behaviours
plot(xx, phpd(xx), type = "l")
lines(xx, phpd(xx, xi = 0.3), col = "red")
lines(xx, phpd(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

sim = rhpd(10000, nmean = 0, nsd = 1.5, xi = 0.2)
hist(sim, freq = FALSE, 100, xlim = c(-5, 20), ylim = c(0, 0.2))
lines(xx, dhpd(xx, 0, 1.5, 0.2), col = "blue")

plot(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = 0), type = "l")
lines(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = 0.2), col = "red")
lines(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = -0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

hpdcon *Hybrid Pareto Extreme Value Mixture Model with Single Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the hybrid Pareto extreme value mixture model with a single continuiuty constraint. The parameters are the normal mean `nmean` and standard deviation `nsd`, threshold `u` and GPD shape `xi`.

Usage

```
dhpdcon(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, log = FALSE)

phpdcon(q, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, lower.tail = TRUE)

qhpdcon(p, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, lower.tail = TRUE)

rhpdcon(n = 1, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0)
```

Arguments

`x`	quantile
`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)
`u`	threshold
`xi`	shape parameter
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail which is continuous at the threshold, but with one important difference to all the other mixture models.

The hybrid Pareto does not include the usual tail fraction `phiu` scaling, i.e. so the GPD is not treated as a conditional model for the exceedances. The unscaled GPD is simply spliced with the normal truncated at the threshold, with no rescaling to account for the proportion above the threshold being applied. The parameters have to adjust for the lack of tail fraction scaling.

The continuity constraint leads to the GPD scale `sigmau` being replaced by a function of the normal mean, standard deviation and threshold parameters.

The cumulative distribution function defined upto the threshold $x \leq u$, given by:

$$F(x) = H(x)/r$$

and above the threshold $x > u$:

$$F(x) = (H(u) + G(x))/r$$

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x, nmean, nsd)`, `pgpd(x, u, sigmau, xi)`). The normalisation constant r ensures a proper density and is given by`r = 1 + pnorm(u, mean = nmean, sd = nsd)`, i.e. the 1 comes from integration of the unscaled GPD and the second term is from the usual normal component.

The continuity constraint on the density at the threshold means that $h(u) = g(u)$ where $h(x)$ and $g(x)$ are the normal and unscale GPD density functions (i.e. `dnorm(x, nmean, nsd)` and `dgpd(x, u, sigmau, xi)`).

The continuity constraint on its first derivative at the threshold means that $h'(u) = g'(u)$. Then the Lambert W function is used for replacing the threshold u and GPD scale sigmau in terms of the normal mean, standard deviation and GPD shape xi.

See gpd for details of GPD upper tail component and dnorm for details of normal bulk component.

Value

dhpdcon gives the density, phpdcon gives the cumulative distribution function, qhpdcon gives the quantile function and rhpdcon gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rhpd` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for rhpd is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

gpd, dnorm, dnormgpd and dnormgpdcon. The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the hybrid Pareto hpareto and mixture of hybrid Paretos hparetomixt, which are more flexible as they also permit the model to be truncated at zero.

Other hpdcon: fhpdcon, lhpdcon, nlhpdcon

Examples

```
## Not run:
par(mfrow = c(2, 2))
xx = seq(-5, 20, 0.01)
f1 = dhpdcon(xx, nmean = 0,1, nsd = 0.4, u=0.5)
plot(xx, f1, type = "l")

# three tail behaviours
plot(xx, phpdcon(xx), type = "l")
lines(xx, phpdcon(xx, xi = 0.3), col = "red")
lines(xx, phpdcon(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

sim = rhpdcon(1000, nmean = 0, nsd = 1.5, u = 0.5, xi = 0.2)
hist(sim, freq = FALSE, 100, xlim = c(-5, 20), ylim = c(0, 0.2))
lines(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 0.5, xi = 0.2), col = "blue")

plot(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 0.5, xi = 0), type = "l")
lines(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 0.5, xi = 0.2), col = "red")
lines(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 0.5, xi = -0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`internal` *Internal Functions*

Description

Internal functions not designed to be used directly, but are all exported to make them visible to users.

Usage

```
kdenx(x, kerncentres, lambda)

pkdenx(x, kerncentres, lambda)

simplebckdenx(x, kerncentres, lambda)

simplepbckdenx(x, kerncentres, lambda)
```

```
renormbckdenx(x, kerncentres, lambda)

renormpbckdenx(x, kerncentres, lambda)

reflectbckdenx(x, kerncentres, lambda)

reflectpbckdenx(x, kerncentres, lambda)

pxb(x, lambda)

beta1bckdenx(x, kerncentres, lambda, xmax)

beta1pbckdenx(x, kerncentres, lambda, xmax)

beta2bckdenx(x, kerncentres, lambda, xmax)

beta2pbckdenx(x, kerncentres, lambda, xmax)

gamma1bckdenx(x, kerncentres, lambda)

gamma1pbckdenx(x, kerncentres, lambda)

gamma2bckdenx(x, kerncentres, lambda)

gamma2pbckdenx(x, kerncentres, lambda)

copulabckdenx(x, kerncentres, lambda, xmax)

copulapbckdenx(x, kerncentres, lambda, xmax)

logpbckdenx(x, kerncentres, lambda, offset)

nnbckdenx(x, kerncentres, lambda)

nnpbckdenx(x, kerncentres, lambda)
```

Arguments

x	quantile
kerncentres	kernel centres (typically sample data)
lambda	bandwidth for KDE
xmax	upper bound on support, for copula and beta based KDE's only
offset	offset added to kernel centres, log transform based KDE

Details

Internal functions not designed to be used directly. No error checking of the inputs is carried out, so user must be know what they are doing. They are undocumented, but are made visible to the user.

Mostly, these are used in the kernel density estimation functions.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>. Based on code by Anna MacDonald produced for MATLAB.

See Also

kden and bckden.

---

kden *Kernel Density Estimation Using Normal Kernel*

Description

Density, cumulative distribution function, quantile function and random number generation for the kernel density estimation using the normal kernel with a constant bandwidth `lambda`. The kernel centres (typically the data) are given by `kerncentres`.

Usage

```
dkden(x, kerncentres, lambda = NULL, log = FALSE)

pkden(q, kerncentres, lambda = NULL, lower.tail = TRUE)

qkden(p, kerncentres, lambda = NULL, lower.tail = TRUE)

rkden(n = 1, kerncentres, lambda = NULL)
```

Arguments

`lambda`	bandwidth for normal kernel (standard deviation of normal)
`kerncentres`	kernel centres (typically sample data)
`x`	quantile
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Kernel density estimation using normal density as kernel.

The density function dkden produces exactly the same density estimate as `density` when a sequence of `x` values are provided, see examples. The latter function is far more efficient in this situation as it takes advantage of the computational savings from doing the kernel smoothing in the spectral domain, where the convolution becomes a multiplication. So even after accounting for applying the (Fast) Fourier Transform (FFT) and its inverse it is much more efficient especially for a large sample size.

However, this KDE function applies the less efficient convolution using the classic definition:

$$\hat{f}_(x) = \frac{1}{n}\sum_{j=1}^{n} K(\frac{x - x_j}{\lambda})$$

where $K(.)$ is the density function for the standard normal. Thus is no restriction on the values x can take. Computationally for a particular x the density is then just `mean(dnorm(x, kerncentres, lambda))` for the density and `mean(pnorm(x, kerncentres, lambda))` for cumulative distribution function. The random number generation is achieved by treating the KDE as a mixture model with equal probability of coming from each kernel, given by `rnorm(rep(1, n), sample(kerncentres, n, replace = TRUE), sd`. The `sample()` function decides which kernel each of the n generated samples comes from and gives the normal random number generator the kernel center as its mean, with `lambda` as the kernel standard deviation.

The quantile function is rather more complicated as there is no closed form solution, so is typically obtained by approximation or numerical solution to $P(X \leq x_p) = p$ to find x_p. The quantile function qkden evaluates the KDE cumulative distribution function over the range from `c(min(kerncentre) - 5*lambda, max(kerncentre) - 5*lambda)` as for normal kernel the probability of being outside this range is of the order `1e-7`. Outside of the range the quantiles are set to `-Inf` for lower tail and `Inf` for upper tail. A sequence of values of length fifty times the number of kernels is first calculated. Spline based interpolation using splinefun, with default `fmm` method, is then used to approximate the quantile function. This is a similar approach to that taken by Matt Wand in the qkde in the ks package.

If no bandwidth is provided `lambda=NULL` then the normal reference rule is used, from the function bw.nrd0, which is consistent with the density function. At least two kernel centres must be provided as the variance needs to be estimated.

Value

dkden gives the density, pkden gives the cumulative distribution function, qkden gives the quantile function and rkden gives a random sample.

Note

Unlike all the other extreme value mixture model functions the kden functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length.

The kernel centres `kerncentres` can either be a single datapoint or a vector of data. The kernel centres (`kerncentres`) and locations to evaluate density (`x`) and cumulative distribution function (`q`) would usually be different.

Default values are provided for all inputs, except for the fundamentals `kerncentres`, `x`, `q` and `p`. The default sample size for rkden is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The normal mean `nmean` and GPD threshold `u` will also require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>. Based on code by Anna MacDonald produced for MATLAB.

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

density, bw.nrd0 and dkde in ks package.

Other kden: fkden, lkden, nlkden

Examples

```
## Not run:
nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
plot(xx, dnorm(xx))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = bw.nrd0(x))*0.05)
lines(xx, dkden(xx, x), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "KDE Using evmix", "KDE Using density function"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "green"))

# Estimate bandwidth using cross-validation likelihood
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 5))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$lambda)*0.05)
lines(xx,dnorm(xx), col = "black")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
lines(density(x, bw = fit$lambda), lwd = 2, lty = 2,  col = "blue")
legend("topright", c("True Density", "KDE fitted evmix",
"KDE Using density, default bandwidth", "KDE Using density, c-v likelihood bandwidth"),
lty = c(1, 1, 2, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

plot(xx, pnorm(xx), type = "l")
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$lambda)*0.05)
lines(xx, pkden(xx, x), lwd = 2, col = "red")
lines(xx, pkden(xx, x, lambda = fit$lambda), lwd = 2, col = "green")
# green and blue (quantile) function should be same
p = seq(0, 1, 0.001)
lines(qkden(p, x, lambda = fit$lambda), p, lwd = 2, lty = 2, col = "blue")
```

```
legend("topleft", c("True Density", "KDE using evmix, normal reference rule",
"KDE using evmix, c-v likelihood","KDE quantile function, c-v likelihood"),
lty = c(1, 1, 1, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

xnew = rkden(1000, x, lambda = fit$lambda)
hist(xnew, breaks = 200, freq = FALSE, xlim = c(-5, 5))
rug(xnew)
lines(xx,dnorm(xx), col = "black")
lines(xx, dkden(xx, x), lwd = 2, col = "red")
legend("topright", c("True Density", "KDE Using evmix"),
lty = c(1, 2), lwd = c(1, 2), col = c("black", "red"))

## End(Not run)
```

---

kdengpd *Kernel Density Estimation Using Normal Kernel and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the kernel density estimation using normal kernel for the bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the bandwidth `lambda`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dkdengpd(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, log = FALSE)

pkdengpd(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qkdengpd(p, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rkdengpd(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phiu = TRUE)
```

Arguments

`x`	quantile
`kerncentres`	kernel centres (typically sample data)
`lambda`	bandwidth for normal kernel (standard deviation of normal)

log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1]

Details

Extreme value mixture model combining kernel density estimation Using normal kernel for the bulk below the threshold and GPD for upper tail. The user can pre-specify phiu permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the kernel density estimation using normal kernel (phiu=TRUE), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the kernel density estimator and conditional GPD cumulative distribution functions (i.e. mean(pnorm(x, kerncentres, lambda)) and pgpd(x, u, sigmau, xi)).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

See gpd for details of GPD upper tail component and dnorm for details of normal bulk component.

Value

dkdengpd gives the density, pkdengpd gives the cumulative distribution function, qkdengpd gives the quantile function and rkdengpd gives a random sample.

Note

Unlike all the other extreme value mixture model functions the kdengpd functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length. The kerncentres can also be a scalar or vector.

The kernel centres kerncentres can either be a single datapoint or a vector of data. The kernel centres (kerncentres) and locations to evaluate density (x) and cumulative distribution function (q) would usually be different.

Default values are provided for all inputs, except for the fundamentals kerncentres, x, q and p. The default sample size for rkdengpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

gpd and dnorm

Other kdengpd: fkdengpd

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(500, 0, 1)
xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpd(xx, kerncentres, u = 1.2, sigmau = 0.56, xi = 0.1))

plot(xx, pkdengpd(xx, kerncentres), type = "l")
lines(xx, pkdengpd(xx, kerncentres, xi = 0.3), col = "red")
lines(xx, pkdengpd(xx, kerncentres, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1, cex = 0.5)

kerncentres=rnorm(1000, 0, 1)
x = rkdengpd(1000, kerncentres, phiu = 0.1, u = 1.2, sigmau = 0.56, xi = 0.1)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dkdengpd(xx, kerncentres, phiu = 0.1))
```

```
plot(xx, dkdengpd(xx, kerncentres, xi=0, phiu = 0.2), type = "l")
lines(xx, dkdengpd(xx, kerncentres, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dkdengpd(xx, kerncentres, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

kdengpdcon *Kernel Density Estimation Using Normal Kernel and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the kernel density estimation using normal kernel for the bulk distribution upto the threshold and conditional GPD above threshold and continuous at threshold. The parameters are the bandwidth `lambda`, threshold `u` GPD and shape `xi` and tail fraction `phiu`.

Usage

```
dkdengpdcon(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0,
  phiu = TRUE, log = FALSE)

pkdengpdcon(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qkdengpdcon(p, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rkdengpdcon(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0,
  phiu = TRUE)
```

Arguments

`x`	quantile
`kerncentres`	kernel centres (typically sample data)
`lambda`	bandwidth for normal kernel (standard deviation of normal)
`u`	threshold
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1]
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining kernel density estimation using normal kernel for the bulk below the threshold and GPD for upper tail, with a constraint to be continuous at the threshold. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the kernel density estimation using normal kernel (`phiu=TRUE`), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the kernel and conditional GPD cumulative distribution functions (i.e. `mean(pnorm(x, kerncentres, lambda))` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - mean(H(u))$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the KDE and conditional GPD density functions. The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u)/[1 - \phi_u]h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)]/h(u)$$

.

See gpd for details of GPD upper tail component and dkden for details of KDE of bulk component.

Value

dkdengpdcon gives the density, pkdengpdcon gives the cumulative distribution function, qkdengpdcon gives the quantile function and rkdengpdcon gives a random sample.

Note

Unlike all the other extreme value mixture model functions the kdengpdcon functions have not been vectorised as this is not appropriate. The main inputs (`x`, `p` or `q`) must be either a scalar or a vector, which also define the output length. The `kerncentres` can also be a scalar or vector.

The kernel centres `kerncentres` can either be a single datapoint or a vector of data. The kernel centres (`kerncentres`) and locations to evaluate density (`x`) and cumulative distribution function (`q`) would usually be different.

Default values are provided for all inputs, except for the fundamentals `kerncentres`, `x`, `q` and `p`. The default sample size for rkdengpdcon is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The KDE bandwidth lambda will not require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

kdengpd, kden, gpd and dnorm

Other kdengpdcon: fkdengpdcon, lkdengpdcon, nlkdengpdcon

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(500, 0, 1)
xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpdcon(xx, kerncentres, u = 1.2, xi = 0.1))

plot(xx, pkdengpdcon(xx, kerncentres), type = "l")
lines(xx, pkdengpdcon(xx, kerncentres, xi = 0.3), col = "red")
lines(xx, pkdengpdcon(xx, kerncentres, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1, cex = 0.5)

kerncentres=rnorm(1000, 0, 1)
x = rkdengpdcon(1000, kerncentres, phiu = 0.1, u = 1.2, xi = 0.1)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dkdengpdcon(xx, kerncentres, phiu = 0.1))

plot(xx, dkdengpdcon(xx, kerncentres, xi=0, phiu = 0.2), type = "l")
lines(xx, dkdengpdcon(xx, kerncentres, xi=-0.2, phiu = 0.2), col = "red")
```

```
lines(xx, dkdengpdcon(xx, kerncentres, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

lbckden *Cross-validation Log-likelihood of Boundary Corrected Kernel Density Estimation*

Description

Cross-validation log-likelihood and negative log-likelihood for boundary corrected kernel density estimation, by treating it as a mixture model.

Usage

```
lbckden(x, lambda = NULL, extracentres = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, log = TRUE)

nlbckden(lambda, x, extracentres = NULL,
  bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = 0, xmax = Inf, finitelik = FALSE)
```

Arguments

x	quantile
lambda	scalar value of fixed bandwidth, or NULL (default)
bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
offset	offset added to kernel centres, for logtrans
xmax	upper bound on support, for copula and beta kernels only
log	logical, if TRUE then log density
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The cross-validation likelihood functions for the boundary corrected kernel density estimator, as used in the maximum likelihood fitting function fbckden.

They are designed to be used for MLE in fbckden but are available for wider usage, e.g. constructing your own extreme value mixture models.

All of the boundary correction methods available in bckden are permitted.

See fkden and fgpd for full details.

The cross-validation likelihood is obtained by leaving each point out in turn, obtaining the usual KDE and evaluate at the point left out:

$$L(\lambda)\prod_{i=1}^{n}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i)=\frac{1}{(n-1)\lambda}\sum_{j=1:j\neq i}^{n}K(\frac{x_i-x_j}{\lambda})$$

is the KDE obtained when the ith datapoint is dropped but is evaluated at x_i.

Normally for likelihood estimation of the bandwidth the kernel centres and the data where the likelihood is evaluated are the same. However, when using KDE for extreme value mixture modelling the likelihood only those data in the bulk of the distribution should contribute to the likelihood, but all the data (including those beyond the threshold) should contribute to the density estimate. The `extracentres` option allows the use to specify extra kernel centres used in estimating the density, but not evaluated in the likelihood. The default is to just use the existing data, so `extracentres=NULL`.

Log-likelihood calculations are carried out in `lbckden`, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for `lbckden`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function `lbckden` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

Value

`lbckden` gives cross-validation (log-)likelihood and `nbclkden` gives the negative cross-validation log-likelihood.

Warning

See warning in `fbckden`

Note

Invalid bandwidth parameter will give `0` for likelihood, `log(0)=-Inf` for cross-validation log-likelihood and `-log(0)=Inf` for negative cross-validation log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott `<carl.scarrott@canterbury.ac.nz>`

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

density

`lbckdengpd` *Cross-validation Log-likelihood of Boundary Corrected Kernel Density Estimators for the Bulk and GPD Tail Extreme Value Mixture Model*

Description

Cross-validation log-likelihood and negative log-likelihood for the Boundary Corrected Kernel Density Estimators for the Bulk and GPD Tail Extreme Value Mixture Model.

Usage

```
lbckdengpd(x, lambda = NULL, u = 0, sigmau = 1, xi = 0,
  phiu = TRUE, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = 0, xmax = Inf, log = TRUE)
```

Arguments

x	quantile
phiu	logical
bcmethod	boundary correction approach
proper	logical, should density be renormalised to integrate to unity, simple boundary correction only
nn	non-negativity correction, so simple boundary correction only
offset	offset added to kernel centres, for logtrans
xmax	upper bound on support, for copula and beta kernels only
lambda	scalar value of fixed bandwidth, or NULL (default)
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The cross-validation likelihood functions for the boundary corrected kernel density estimators for the bulk for the bulk below the threshold and GPD for upper tail. As used in the maximum likelihood fitting function fbckdengpd.

They are designed to be used for MLE in fbckdengpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fbckden, fkden and fgpd for full details.

Cross-validation likelihood is used for boundary corrected kernel density component, but standard likelihood is used for GPD component. The cross-validation likelihood for the KDE is obtained by leaving each point out in turn, evaluating the KDE at the point left out:

$$L(\lambda)\prod_{i=1}^{nb}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i)=\frac{1}{(n-1)\lambda}\sum_{j=1:j\neq i}^{n}K(\frac{x_i-x_j}{\lambda})$$

is the boundary corrected KDE obtained when the ith datapoint is dropped out and then evaluated at that dropped datapoint at x_i. Notice that the coundary corrected KDE sum is indexed over all datapoints ($j=1,...,n$, except datapoint i) whether they are below the threshold or in the upper tail. But the likelihood product is evaluated only for those data below the threshold ($i=1,...,n_b$). So the $j=n_b+1,...,n$ datapoints are extra kernel centres from the data in the upper tails which are used in the boundary corrected KDE but the likelihood is not evaluated there.

Log-likelihood calculations are carried out in lbckdengpd, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for lbckdengpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function lbckdengpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lbckdengpd gives cross-validation (log-)likelihood and nlbckdengpd gives the negative cross-validation log-likelihood.

Warning

See warning in fkden

Note

Invalid bandwidth parameter will give 0 for likelihood, log(0)=-Inf for cross-validation log-likelihood and -log(0)=Inf for negative cross-validation log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

bckden, kden, gpd and density

---

lbetagpd — *Log-likelihood of beta Bulk and GPD Tail Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold.

Usage

```
lbetagpd(x, bshape1 = 1, bshape2 = 1,
  u = qbeta(0.9, bshape1, bshape2),
  sigmau = sqrt(bshape1 * bshape2/(bshape1 + bshape2)^2/(bshape1 + bshape2 + 1)),
  xi = 0, phiu = TRUE, log = TRUE)

nlbetagpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

phiu	probability of being above threshold [0,1] or logical
x	vector of sample data
pvector	vector of initial values mixture model parameters (bshape1, bshape2, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
bshape1	beta shape 1 (non-negative)
bshape2	beta shape 2 (non-negative)
u	threshold over $(0, 1)$
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with beta bulk and GPD tail, as used in the maximum likelihood fitting function fbetagpd.

Non-positive data are ignored. Values above 1 must come from GPD component, as threshold u<1..

They are designed to be used for MLE in fbetagpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fbetagpd and fgpd for full details.

Log-likelihood calculations are carried out in lbetagpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lbetagpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions betagpd the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by beta distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lbetagpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lbetagpd gives (log-)likelihood and nlbetagpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lbetagpd and nlnormgpd. The lbetagpd also has the usual defaults for the other parameters, but nlbetagpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Beta_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

MacDonald, A. (2012). Extreme value mixture modelling with medical and industrial applications. PhD thesis, University of Canterbury, New Zealand. http://ir.canterbury.ac.nz/bitstream/10092/6679/1/thesis_fulltext.pdf

See Also

lgpd and gpd

Other betagpd: betagpd, dbetagpd, fbetagpd, pbetagpd, qbetagpd, rbetagpd

---

ldwm *Log-likelihood of dynamically weighted mixture model*

Description

Log-likelihood and negative log-likelihood for the dynamically weighted mixture model

Usage

```
ldwm(x, wshape = 1, wscale = 1, cmu = 1, ctau = 1,
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, log = TRUE)

nldwm(pvector, x, finitelik = FALSE)
```

Arguments

x	vector of sample data
pvector	vector of initial values of mixture model parameters (wshape, wscale, cmu, ctau, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
wshape	Weibull shape (non-negative)
wscale	Weibull scale (non-negative)
cmu	Cauchy location
ctau	Cauchy scale
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The likelihood functions for the dynamically weighted mixture model fdwm.

Non-positive data are ignored.

They are designed to be used for MLE in fdwm but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fdwm and fgpd for full details.

Log-likelihood calculations are carried out in ldwm, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for ldwm, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function ldwm carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

ldwm gives (log-)likelihood and nldwm gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Frigessi, A., O. Haug, and H. Rue (2002). A dynamic mixture model for unsupervised tail estimation without threshold selection. Extremes 5 (3), 219-235

See Also

lgpd and gpd

Other dwm: fdwm

---

lgammagpd *Log-likelihood of gamma Bulk and GPD Tail Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold.

Usage

```
lgammagpd(x, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale),
  sigmau = sqrt(gshape) * gscale, xi = 0, phiu = TRUE,
  log = TRUE)

nlgammagpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

phiu	probability of being above threshold [0,1] or logical
x	vector of sample data
pvector	vector of initial values mixture model parameters (gshape, gscale, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
gshape	gamma shape (non-negative)
gscale	gamma scale (non-negative)
u	threshold (non-negative)
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with gamma bulk and GPD tail, as used in the maximum likelihood fitting function fgammagpd.

They are designed to be used for MLE in fgammagpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

Negative data are ignored.

See fgammagpd and fgpd for full details.

Log-likelihood calculations are carried out in lgammagpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lgammagpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions gammagpd the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by gamma distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lgammagpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgammagpd gives (log-)likelihood and nlgammagpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lgammagpd and nlnormgpd. The lgammagpd also has the usual defaults for the other parameters, but nlgammagpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Gamma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other gammagpd: dgammagpd, fgammagpd, gammagpd, pgammagpd, qgammagpd, rgammagpd

---

lgammagpdcon *Log-likelihood of gamma Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with gamma for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
lgammagpdcon(x, gshape = 1, gscale = 1,
  u = qgamma(0.9, gshape, 1/gscale), xi = 0, phiu = TRUE,
  log = TRUE)

nlgammagpdcon(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

x	vector of sample data
gshape	gamma shape (non-negative)
gscale	gamma scale (non-negative)
u	threshold (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1] or logical

log	logical, if TRUE then log density
pvector	vector of initial values mixture model parameters (gshape, gscale, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for the extreme value mixture model with gamma bulk and GPD tail, as used in the maximum likelihood fitting function fgammagpdcon.

They are designed to be used for MLE in fgammagpdcon but are available for wider usage, e.g. constructing your own extreme value mixture models.

Negative data are ignored.

See fgammagpdcon and fgpd for full details.

Log-likelihood calculations are carried out in lgammagpdcon, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lgammagpdcon, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions gammagpdcon the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by gamma distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lgammagpdcon carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgammagpdcon gives (log-)likelihood and nlgammagpdcon gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lgammagpdcon and nlnormgpd. The lgammagpdcon also has the usual defaults for the other parameters, but nlgammagpdcon has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Gamma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgammagpd, lgpd and gpd

Other gammagpdcon: dgammagpdcon, fgammagpdcon, gammagpdcon, pgammagpdcon, qgammagpdcon, rgammagpdcon

---

lgkg *Cross-validation Log-likelihood of Kernel Density Estimation for Bulk and GPD for Both Upper and Lower Tails in Extreme Value Mixture Model*

Description

Cross-validation log-likelihood and negative log-likelihood for the kernel density estimation using normal kernel bulk and GPD upper and lower tails extreme value mixture model.

Usage

```
lgkg(x, lambda = NULL, ul = as.vector(quantile(x, 0.1)),
  sigmaul = 1, xil = 0, phiul = TRUE,
  ur = as.vector(quantile(x, 0.9)), sigmaur = 1, xir = 0,
  phiur = TRUE, log = TRUE)

nlgkg(pvector, x, phiul = TRUE, phiur = TRUE,
  finitelik = FALSE)
```

Arguments

x	vector of sample data
phiul	logical
phiur	logical
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
lambda	bandwidth for normal kernel (standard deviation of normal)
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (non-negative)
xil	lower tail GPD shape parameter

ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (non-negative)
xir	upper tail GPD shape parameter
log	logical, if TRUE then log density

Details

The cross-validation likelihood functions for the extreme value mixture model with kernel density estimation using normal kernel for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails. As used in the maximum likelihood fitting function fgkg.

They are designed to be used for MLE in fgkg but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fkdengpd, fkden and fgpd for full details.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD components. The cross-validation likelihood for the KDE is obtained by leaving each point out in turn, evaluating the KDE at the point left out:

$$L(\lambda)\prod_{i=1}^{nb}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i)=\frac{1}{(n-1)\lambda}\sum_{j=1:j\neq i}^{n}K(\frac{x_i-x_j}{\lambda})$$

is the KDE obtained when the ith datapoint is dropped out and then evaluated at that dropped datapoint at x_i. Notice that the KDE sum is indexed over all datapoints ($j=1,...,n$, except datapoint i) whether they are between the thresholds or in the tails. But the likelihood product is evaluated only for those data between the thresholds ($i=1,...,n_b$). So the $j=n_b+1,...,n$ datapoint are extra kernel centres from the data in the tails which are used in the KDE but the likelihood is not evaluated there.

Log-likelihood calculations are carried out in lgkg, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for lgkg, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function lgkg carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgkg gives cross-validation (log-)likelihood and nlgkg gives the negative cross-validation log-likelihood.

Warning

See warning in fkden

Note

Invalid bandwidth parameter will give 0 for likelihood, log(0)=-Inf for cross-validation log-likelihood and -log(0)=Inf for negative cross-validation log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

gkg, kdengpd, kden, gpd and density.

Other gkg: dgkg, fgkg, gkg, pgkg, qgkg, rgkg

---

lgng *Log-likelihood of Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with normal for bulk distribution between the lower and upper thresholds with conditional GPD for the two tails.

Usage

```
lgng(x, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), sigmaur = nsd, xir = 0,
  phiur = TRUE, log = TRUE)

nlgng(pvector, x, phiul = TRUE, phiur = TRUE,
  finitelik = FALSE)
```

Arguments

phiul	probability of being above threshold (0, 1) or logical
phiur	probability of being above threshold (0, 1) or logical
x	vector of sample data
pvector	vector of initial values of mixture model parameters or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

nmean	normal mean
nsd	normal standard deviation (non-negative)
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (non-negative)
xil	lower tail GPD shape parameter
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (non-negative)
xir	upper tail GPD shape parameter
log	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with normal bulk and GPD for the two tails, as used in the maximum likelihood fitting function fgng.

They are designed to be used for MLE in fgng but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fgng, gng and fgpd for full details.

Log-likelihood calculations are carried out in lgng, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lgng, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fractions phiul and phiur are treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions gng the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lgng carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgng gives (log-)likelihood and nlgng gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

Default values for the tail fractions phiul=TRUE and phiur=TRUE is given in both lgng and nlgng. The lgng also has the usual defaults for the other parameters, but nlgng has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

lnormgpd, lgpd and gpd

Other gng: dgng, fgng, gng, pgng, qgng, rgng

---

lgngcon — *Log-likelihood of Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model with Continuity Constraints*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with normal for bulk distribution between the lower and upper thresholds with conditional GPD for the two tails with continuity constraints.

Usage

```
lgngcon(x, nmean = 0, nsd = 1,
  ul = qnorm(0.1, nmean, nsd), xil = 0, phiul = TRUE,
  ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
  log = TRUE)

nlgngcon(pvector, x, phiul = TRUE, phiur = TRUE,
  finitelik = FALSE)
```

Arguments

x	vector of sample data
nmean	normal mean
nsd	normal standard deviation (non-negative)
ul	lower tail threshold
xil	lower tail GPD shape parameter
phiul	probability of being above threshold (0, 1) or logical
ur	upper tail threshold
xir	upper tail GPD shape parameter

phiur	probability of being above threshold (0, 1) or logical
log	logical, if TRUE then log density
pvector	vector of initial values of mixture model parameters or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for the extreme value mixture model with normal bulk and GPD for the two tails, as used in the maximum likelihood fitting function fgngcon.

They are designed to be used for MLE in fgngcon but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fgngcon, gngcon and fgpd for full details.

Log-likelihood calculations are carried out in lgngcon, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lgngcon, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fractions phiul and phiur are treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions gngcon the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lgngcon carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgngcon gives (log-)likelihood and nlgngcon gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

Default values for the tail fractions phiul=TRUE and phiur=TRUE is given in both lgngcon and nlgngcon. The lgngcon also has the usual defaults for the other parameters, but nlgngcon has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Zhao, X., Scarrott, C.J. Reale, M. and Oxley, L. (2010). Extreme value modelling for forecasting the market crisis. Applied Financial Econometrics 20(1), 63-72.

See Also

lgng, lnormgpd, lgpd and gpd

Other gngcon: dgngcon, fgngcon, gngcon, pgngcon, qgngcon, rgngcon

---

lgpd *Log-likelihood of Generalised Pareto Distribution (GPD)*

Description

Log-likelihood and negative log-likelihood for the GPD conditional on being above a threshold u with parameters scale sigmau and shape xi. Unconditional likelihood also provided when the probability phiu of being above the threshold u is given.

Usage

```
lgpd(x, u = 0, sigmau = 1, xi = 0, phiu = 1, log = TRUE)

nlgpd(pvector, x, u = 0, phiu = 1, finitelik = FALSE)
```

Arguments

x	quantile
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1]
log	logical, if TRUE then log density
pvector	vector of initial values of GPD parameters (sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The GPD likelihood functions for the exceedances of the threshold u as used in the maximum likelihood fitting function fgpd.

They are designed to be used for MLE in fgpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fgpd and dgpd for full details.

Log-likelihood calculations are carried out in lgpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lgpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

Unlike the MLE function fgpd, the phiu must be in range $[0, 1]$ and cannot be NULL. Specify phiu=1 for conditional likelihood (default) and phiu<1 for unconditional likelihood.

The function lgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lgpd gives (log-)likelihood and nlgpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for the GPD, the likelihood functions only permits a scalar value for scale and shape parameters, phiu and threshold u. Only the data is a vector.

Default values for the threshold u=0 and tail fraction phiu=1 are given in both lgpd and nlgpd, assuming the user will default to entering excesses above the threshold, rather than exceeedances. The lgpd has the usual defaults for the GPD scale and shape parameters, but nlgpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Based on GPD likelihood function in the evd package.

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

See Also

dgpd, fpot and fitdistr

Other gpd: dgpd, fgpd, gpd, pgpd, qgpd, rgpd

lhpd *Log-likelihood of Hybrid Pareto Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the hybrid Pareto extreme value mixture model

Usage

```
lhpd(x, nmean = 0, nsd = 1, xi = 0, log = TRUE)

nlhpd(pvector, x, finitelik = FALSE)
```

Arguments

x	vector of sample data
nmean	normal mean
nsd	normal standard deviation (non-negative)
xi	shape parameter
log	logical, if TRUE then log density
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for hybrid Pareto extreme value mixture model, as used in the maximum likelihood fitting function fhpd.

They are designed to be used for MLE in fhpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fhpd and fgpd for full details.

Log-likelihood calculations are carried out in lhpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lhpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function lhpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lhpd gives (log-)likelihood and nlhpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

The lhpd also has the usual defaults for the other parameters, but nlhpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

lgpd and gpd. The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the likelihood of the hybrid Pareto hpareto.negloglike and fitting hpareto.fit.

Other hpd: dhpd, fhpd, hpd, phpd, qhpd, rhpd

---

lhpdcon — *Log-likelihood of Hybrid Pareto Extreme Value Mixture Model with Single Continuity Constraint*

Description

Log-likelihood and negative log-likelihood for the hybrid Pareto extreme value mixture model with a single continuity constraint

Usage

```
lhpdcon(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, log = TRUE)

nlhpdcon(pvector, x, finitelik = FALSE)
```

Arguments

x	vector of sample data
nmean	normal mean
nsd	normal standard deviation (non-negative)
u	threshold
xi	shape parameter
log	logical, if TRUE then log density

pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for hybrid Pareto extreme value mixture model with single continuity constraint, as used in the maximum likelihood fitting function fhpdcon.

They are designed to be used for MLE in fhpdcon but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fhpd and fgpd for full details.

Log-likelihood calculations are carried out in lhpdcon, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lhpdcon, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function lhpdcon carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lhpdcon gives (log-)likelihood and nlhpdcon gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

The lhpdcon also has the usual defaults for the other parameters, but nlhpdcon has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Carreau, J. and Y. Bengio (2008). A hybrid Pareto model for asymmetric fat-tailed data: the univariate case. Extremes 12 (1), 53-76.

See Also

lhpd, lnormgpdcon, codelgpd and gpd. The condmixt package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the likelihood of the hybrid Pareto hpareto.negloglike and fitting hpareto.fit.

Other hpdcon: dhpdcon, fhpdcon, hpdcon, phpdcon, qhpdcon, rhpdcon

---

lkden *Cross-validation Log-likelihood of Kernel Density Estimator Using normal Kernel*

Description

Cross-validation log-likelihood and negative log-likelihood for the kernel density estimator using a normal kernel by treating it as a mixture model.

Usage

```
lkden(x, lambda = NULL, extracentres = NULL, log = TRUE)

nlkden(lambda, x, extracentres = NULL, finitelik = FALSE)
```

Arguments

x	vector of sample data
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
lambda	bandwidth for normal kernel (standard deviation of normal)
log	logical, if TRUE then log density

Details

The cross-validation likelihood functions for the kernel density estimator using a normal density for kernel, as used in the maximum likelihood fitting function fkden.

They are designed to be used for MLE in fkden but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fkden and fgpd for full details.

Cross-validation likelihood is used for kernel density component, obtained by leaving each point out in turn and evaluating the KDE at the point left out:

$$L(\lambda)\prod_{i=1}^{n}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i)=\frac{1}{(n-1)\lambda}\sum_{j=1:j\neq i}^{n}K(\frac{x_i-x_j}{\lambda})$$

is the KDE obtained when the ith datapoint is dropped out and then evaluated at that dropped datapoint at x_i.

Normally for likelihood estimation of the bandwidth the kernel centres and the data where the likelihood is evaluated are the same. However, when using KDE for extreme value mixture modelling the likelihood only those data in the bulk of the distribution should contribute to the likelihood, but all the data (including those beyond the threshold) should contribute to the density estimate. The `extracentres` option allows the use to specify extra kernel centres used in estimating the density, but not evaluated in the likelihood. The default is to just use the existing data, so `extracentres=NULL`.

Log-likelihood calculations are carried out in `lkden`, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for `lkden`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function `lkden` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

Value

`lkden` gives cross-validation (log-)likelihood and `nlkden` gives the negative cross-validation log-likelihood.

Warning

See warning in `fkden`

Note

Invalid bandwidth parameter will give `0` for likelihood, `log(0)=-Inf` for cross-validation log-likelihood and `-log(0)=Inf` for negative cross-validation log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

density

Other kden: dkden, fkden, kden, pkden, qkden, rkden

---

lkdengpd *Cross-validation Log-likelihood of Kernel Density Estimator Using Normal Kernel and GPD Tail Extreme Value Mixture Model*

Description

Cross-validation log-likelihood and negative log-likelihood for the kernel density estimator using a normal kernels and GPD Tail Extreme Value Mixture Model.

Usage

```
lkdengpd(x, lambda = NULL, u = 0, sigmau = 1, xi = 0,
  phiu = TRUE, log = TRUE)

nlkdengpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

x	vector of sample data
phiu	logical
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
lambda	bandwidth for normal kernel (standard deviation of normal)
u	threshold
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The cross-validation likelihood functions for the kernel density estimator using normal kernel for the bulk below the threshold and GPD for upper tail. As used in the maximum likelihood fitting function fkdengpd.

They are designed to be used for MLE in fkdengpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fkden and fgpd for full details.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD component. The cross-validation likelihood for the KDE is obtained by leaving each point out in turn, evaluating the KDE at the point left out:

$$L(\lambda)\prod_{i=1}^{nb}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i) = \frac{1}{(n-1)\lambda} \sum_{j=1:j\neq i}^{n} K(\frac{x_i - x_j}{\lambda})$$

is the KDE obtained when the ith datapoint is dropped out and then evaluated at that dropped datapoint at x_i. Notice that the KDE sum is indexed over all datapoints ($j = 1, ..., n$, except datapoint i) whether they are below the threshold or in the upper tail. But the likelihood product is evaluated only for those data below the threshold ($i = 1, ..., n_b$). So the $j = n_b + 1, ..., n$ datapoints are extra kernel centres from the data in the upper tails which are used in the KDE but the likelihood is not evaluated there.

Log-likelihood calculations are carried out in `lkdengpd`, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for `lkdengpd`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function `lkdengpd` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

Value

`lkdengpd` gives cross-validation (log-)likelihood and `nlkdengpd` gives the negative cross-validation log-likelihood.

Warning

See warning in `fkden`

Note

Invalid bandwidth parameter will give `0` for likelihood, `log(0)=-Inf` for cross-validation log-likelihood and `-log(0)=Inf` for negative cross-validation log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

kdengpd, kden, gpd and density

---

lkdengpdcon — *Cross-validation Log-likelihood of Kernel Density Estimator Using Normal Kernel and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

Description

Cross-validation log-likelihood and negative log-likelihood for the kernel density estimator using normal kernel for the bulk distribution upto the threshold and conditional GPD above threshold and continuous at threshold

Usage

```
lkdengpdcon(x, lambda = NULL, u = 0, xi = 0, phiu = TRUE,
  log = TRUE)

nlkdengpdcon(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

x	vector of sample data
phiu	logical
pvector	vector of initial values of mixture model parameters (nmean, nsd, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
lambda	bandwidth for normal kernel (standard deviation of normal)
u	threshold
xi	shape parameter
log	logical, if TRUE then log density

Details

The cross-validation likelihood functions for the kernel density estimator using normal kernel for the bulk below the threshold and GPD for upper tail, with a constraint to be continuous at the threshold. As used in the maximum likelihood fitting function fkdengpdcon.

They are designed to be used for MLE in fkdengpdcon but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fkdengpd, fkden and fgpd for full details.

Cross-validation likelihood is used for kernel density component, but standard likelihood is used for GPD component. The cross-validation likelihood for the KDE is obtained by leaving each point out in turn, evaluating the KDE at the point left out:

$$L(\lambda)\prod_{i=1}^{nb}\hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i) = \frac{1}{(n-1)\lambda} \sum_{j=1:j\neq i}^{n} K(\frac{x_i - x_j}{\lambda})$$

is the KDE obtained when the ith datapoint is dropped out and then evaluated at that dropped datapoint at x_i. Notice that the KDE sum is indexed over all datapoints ($j = 1, ..., n$, except datapoint i) whether they are below the threshold or in the upper tail. But the likelihood product is evaluated only for those data below the threshold ($i = 1, ..., n_b$). So the $j = n_b + 1, ..., n$ datapoints are extra kernel centres from the data in the upper tails which are used in the KDE but the likelihood is not evaluated there.

Log-likelihood calculations are carried out in `lkdengpdcon`, which takes bandwidth in the same form as distribution functions. The negative log-likelihood is a wrapper for `lkdengpdcon`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The function `lkdengpdcon` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

Value

`lkdengpdcon` gives cross-validation (log-)likelihood and `nlkdengpdcon` gives the negative cross-validation log-likelihood.

Warning

See warning in `fkden`

Note

Invalid bandwidth parameter will give `0` for likelihood, `log(0)=-Inf` for cross-validation log-likelihood and `-log(0)=Inf` for negative cross-validation log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott `<carl.scarrott@canterbury.ac.nz>`

References

http://en.wikipedia.org/wiki/Kernel_density_estimation

http://en.wikipedia.org/wiki/Cross-validation_(statistics)

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Bowman, A.W. (1984). An alternative method of cross-validation for the smoothing of density estimates. Biometrika 71(2), 353-360.

Duin, R.P.W. (1976). On the choice of smoothing parameters for Parzen estimators of probability density functions. IEEE Transactions on Computers C25(11), 1175-1179.

MacDonald, A., Scarrott, C.J., Lee, D., Darlow, B., Reale, M. and Russell, G. (2011). A flexible extreme value mixture model. Computational Statistics and Data Analysis 55(6), 2137-2157.

See Also

kdengpd, kden, gpd and density

Other kdengpdcon: dkdengpdcon, fkdengpdcon, kdengpdcon, pkdengpdcon, qkdengpdcon, rkdengpdcon

---

`llognormgpd` *Log-likelihood of Log-Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with log-normal for bulk distribution upto the threshold and conditional GPD above threshold.

Usage

```
llognormgpd(x, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), sigmau = lnsd, xi = 0,
  phiu = TRUE, log = TRUE)

nllognormgpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

`phiu`	probability of being above threshold [0,1] or logical
`x`	vector of sample data
`pvector`	vector of initial values of mixture model parameters (`lnmean`, `lnsd`, `u`, `sigmau`, `xi`) or `NULL`
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`lnmean`	mean on log scale
`lnsd`	standard deviation on log scale (non-negative)
`u`	threshold
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with log-normal bulk and GPD tail, as used in the maximum likelihood fitting function `flognormgpd`.

They are designed to be used for MLE in `flognormgpd` but are available for wider usage, e.g. constructing your own extreme value mixture models.

Negative data are ignored.

See `flognormgpd`, `fnormgpd` and `fgpd` for full details.

Log-likelihood calculations are carried out in `llognormgpd`, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for `llognormgpd`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first

input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions lognormgpd the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by log-normal distribution $\phi_u = 1-H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function llognormgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

llognormgpd gives (log-)likelihood and nllognormgpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both llognormgpd and nllognormgpd. The llognormgpd also has the usual defaults for the other parameters, but nllognormgpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

lgpd and gpd

Other lognormgpd: dlognormgpd, flognormgpd, lognormgpd, plognormgpd, qlognormgpd, rlognormgpd

`llognormgpdcon` *Log-likelihood of Log-Normal Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with log-normal for bulk distribution upto the threshold and conditional GPD above threshold and with a continuity constraint.

Usage

```
llognormgpdcon(x, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), xi = 0, phiu = TRUE,
  log = TRUE)

nllognormgpdcon(pvector, x, phiu = TRUE,
  finitelik = FALSE)
```

Arguments

`x`	vector of sample data
`lnmean`	mean on log scale
`lnsd`	standard deviation on log scale (non-negative)
`u`	threshold
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or logical
`log`	logical, if TRUE then log density
`pvector`	vector of initial values of mixture model parameters (`lnmean`, `lnsd`, `u`, `sigmau`, `xi`) or `NULL`
`finitelik`	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for the extreme value mixture model with log-normal bulk and GPD tail with continuity constraint, as used in the maximum likelihood fitting function `flognormgpdcon`.

They are designed to be used for MLE in `flognormgpdcon` but are available for wider usage, e.g. constructing your own extreme value mixture models.

Negative data are ignored.

See `flognormgpdcon`, `flognormgpd`, `fnormgpd` and `fgpd` for full details.

Log-likelihood calculations are carried out in `llognormgpdcon`, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for `llognormgpdcon`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction `phiu` is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions `lognormgpdcon` the `phiu` must be either logical (`TRUE` or `FALSE`) or numerical in range $(0, 1)$. The default is to specify `phiu=TRUE` so that the tail fraction is specified by

log-normal distribution $\phi_u = 1-H(u)$, or `phiu=FALSE` to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric `phiu` as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function `llognormgpdcon` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

Value

`llognormgpdcon` gives (log-)likelihood and `nllognormgpdcon` gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction `phiu=TRUE` is given in both `llognormgpdcon` and `nllognormgpdcon`. The `llognormgpdcon` also has the usual defaults for the other parameters, but `nllognormgpdcon` has no defaults.

Invalid parameters will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

`lognormgpd`, `lgpd` and `gpd`

Other lognormgpdcon: `dlognormgpdcon`, `flognormgpdcon`, `lognormgpdcon`, `plognormgpdcon`, `qlognormgpdcon`, `rlognormgpdcon`

`lnormgpd` *Log-likelihood of Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold.

Usage

```
lnormgpd(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
  phiu = TRUE, log = TRUE)

nlnormgpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

`phiu`	probability of being above threshold [0,1] or logical
`x`	vector of sample data
`pvector`	vector of initial values of mixture model parameters (`nmean`, `nsd`, `u`, `sigmau`, `xi`) or `NULL`
`finitelik`	logical, should log-likelihood return finite value for invalid parameters
`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)
`u`	threshold
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with normal bulk and GPD tail, as used in the maximum likelihood fitting function `fnormgpd`.

They are designed to be used for MLE in `fnormgpd` but are available for wider usage, e.g. constructing your own extreme value mixture models.

See `fnormgpd` and `fgpd` for full details.

Log-likelihood calculations are carried out in `lnormgpd`, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for `lnormgpd`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction `phiu` is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions `normgpd` the `phiu` must be either logical (`TRUE` or `FALSE`) or numerical in range $(0, 1)$. The default is to specify `phiu=TRUE` so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$, or `phiu=FALSE` to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric `phiu` as pre-specified probability $(0, 1)$.

Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lnormgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lnormgpd gives (log-)likelihood and nlnormgpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lnormgpd and nlnormgpd. The lnormgpd also has the usual defaults for the other parameters, but nlnormgpd has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lgpd and gpd

Other normgpd: dnormgpd, fnormgpd, normgpd, pnormgpd, qnormgpd, rnormgpd

`lnormgpdcon` *Log-likelihood of Normal Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint

Usage

```
lnormgpdcon(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, phiu = TRUE,
  log = TRUE)

nlnormgpdcon(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

`x`	vector of sample data
`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)
`u`	threshold
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or logical
`log`	logical, if TRUE then log density
`pvector`	vector of initial values of mixture model parameters (`nmean`, `nsd`, `u`, `sigmau`, `xi`) or `NULL`
`finitelik`	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for the extreme value mixture model with normal bulk and GPD tail, as used in the maximum likelihood fitting function `fnormgpdcon`.

They are designed to be used for MLE in `fnormgpdcon` but are available for wider usage, e.g. constructing your own extreme value mixture models.

See `fnormgpdcon` and `fgpd` for full details.

Log-likelihood calculations are carried out in `lnormgpdcon`, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for `lnormgpdcon`, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction `phiu` is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions `normgpdcon` the `phiu` must be either logical (`TRUE` or `FALSE`) or numerical in range $(0, 1)$. The default is to specify `phiu=TRUE` so that the tail fraction is specified by normal distribution $\phi_u = 1 - H(u)$, or `phiu=FALSE` to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric `phiu` as pre-specified probability $(0, 1)$.

Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lnormgpdcon carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lnormgpdcon gives (log-)likelihood and nlnormgpdcon gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lnormgpdcon and nlnormgpdcon. The lnormgpdcon also has the usual defaults for the other parameters, but nlnormgpdcon has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lnormgpd, lgpd and gpd

Other normgpdcon: dnormgpdcon, fnormgpdcon, normgpdcon, pnormgpdcon, qnormgpdcon, rnormgpdcon

lognormgpd *Log-Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with log-normal for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the normal mean `lnmean` and standard deviation `lnsd`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dlognormgpd(x, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), sigmau = lnsd, xi = 0,
  phiu = TRUE, log = FALSE)

plognormgpd(q, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), sigmau = lnsd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qlognormgpd(p, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), sigmau = lnsd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rlognormgpd(n = 1, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), sigmau = lnsd, xi = 0,
  phiu = TRUE)
```

Arguments

`lnmean`	mean on log scale
`lnsd`	standard deviation on log scale (non-negative)
`x`	quantile
`u`	threshold
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or TRUE
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining log-normal distribution for the bulk below the threshold and GPD for upper tail. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the log-normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the log-normal bulk model (`phiu=TRUE`), upto the threshold $0 < x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the log-normal and conditional GPD cumulative distribution functions (i.e. `plnorm(x, meanlog = lnmean, sdlog = lnsd)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 < x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The gamma is defined on the non-negative reals, so the threshold must be non-negative.

See `gpd` for details of GPD upper tail component and `dlnorm` for details of log-normal bulk component.

Value

`dlognormgpd` gives the density, `plognormgpd` gives the cumulative distribution function, `qlognormgpd` gives the quantile function and `rlognormgpd` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rlognormgpd` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rlognormgpd` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

gpd and dlnorm

Other lognormgpd: flognormgpd, llognormgpd, nllognormgpd

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rlognormgpd(1000)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpd(xx))

# three tail behaviours
plot(xx, plognormgpd(xx), type = "l")
lines(xx, plognormgpd(xx, xi = 0.3), col = "red")
lines(xx, plognormgpd(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rlognormgpd(1000, u = 2, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpd(xx, u = 2, phiu = 0.2))

plot(xx, dlognormgpd(xx, u = 2, xi=0, phiu = 0.2), type = "l")
lines(xx, dlognormgpd(xx, u = 2, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dlognormgpd(xx, u = 2, xi=0.2, phiu = 0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

lognormgpdcon *Log-Normal Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with log-normal for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint. The parameters are the normal mean `lnmean` and standard deviation `lnsd`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dlognormgpdcon(x, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), xi = 0, phiu = TRUE,
  log = FALSE)

plognormgpdcon(q, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qlognormgpdcon(p, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rlognormgpdcon(n = 1, lnmean = 0, lnsd = 1,
  u = qlnorm(0.9, lnmean, lnsd), xi = 0, phiu = TRUE)
```

Arguments

`x`	quantile
`lnmean`	mean on log scale
`lnsd`	standard deviation on log scale (non-negative)
`u`	threshold
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or TRUE
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining log-normal distribution for the bulk below the threshold and GPD for upper tail with a continuity constraint. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the log-normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the log-normal bulk model (`phiu=TRUE`), upto the threshold $0 < x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the log-normal and conditional GPD cumulative distribution functions (i.e. `plnorm(x, meanlog = lnmean, sdlog = lnsd)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 < x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the log-normal and conditional GPD density functions (i.e. `dlnorm(x, nmean, nsd)` and `dgpd(x, u, sigmau, xi)`). The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u)/[1 - \phi_u]h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)]/h(u)$$

.

The gamma is defined on the non-negative reals, so the threshold must be non-negative.

See `gpd` for details of GPD upper tail component and `dlnorm` for details of log-normal bulk component.

Value

`dlognormgpdcon` gives the density, `plognormgpdcon` gives the cumulative distribution function, `qlognormgpdcon` gives the quantile function and `rlognormgpdcon` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rlognormgpd` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rlognormgpdcon` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Log-normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Solari, S. and Losada, M.A. (2004). A unified statistical model for hydrological variables including the selection of threshold for the peak over threshold method. Water Resources Research. 48, W10541.

See Also

lognormgpd, gpd and dlnorm

Other lognormgpdcon: flognormgpdcon, llognormgpdcon, nllognormgpdcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rlognormgpdcon(1000)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpdcon(xx))

# three tail behaviours
plot(xx, plognormgpdcon(xx), type = "l")
lines(xx, plognormgpdcon(xx, xi = 0.3), col = "red")
lines(xx, plognormgpdcon(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rlognormgpdcon(1000, u = 2, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpdcon(xx, u = 2, phiu = 0.2))

plot(xx, dlognormgpdcon(xx, u = 2, xi=0, phiu = 0.2), type = "l")
lines(xx, dlognormgpdcon(xx, u = 2, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dlognormgpdcon(xx, u = 2, xi=0.2, phiu = 0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

lweibullgpd *Log-likelihood of Weibull Bulk and GPD Tail Extreme Value Mixture Model*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold.

Usage

```
lweibullgpd(x, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, phiu = TRUE, log = TRUE)

nlweibullgpd(pvector, x, phiu = TRUE, finitelik = FALSE)
```

Arguments

phiu	probability of being above threshold [0,1] or logical
x	vector of sample data
pvector	vector of initial values mixture model parameters (wshape, wscale, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
wshape	Weibull shape (non-negative)
wscale	Weibull scale (non-negative)
u	threshold (non-negative)
sigmau	scale parameter (non-negative)
xi	shape parameter
log	logical, if TRUE then log density

Details

The likelihood functions for the extreme value mixture model with Weibull bulk and GPD tail, as used in the maximum likelihood fitting function fweibullgpd.

Non-positive data are ignored.

They are designed to be used for MLE in fweibullgpd but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fweibullgpd and fgpd for full details.

Log-likelihood calculations are carried out in lweibullgpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lweibullgpd, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions weibullgpd the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by Weibull distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lweibullgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lweibullgpd gives (log-)likelihood and nlweibullgpd gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction `phiu=TRUE` is given in both `lweibullgpd` and `nlnormgpd`. The `lweibullgpd` also has the usual defaults for the other parameters, but `nlweibullgpd` has no defaults.

Invalid parameters will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

See `fgpd` for explanation of `finitelik`.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

`lgpd` and `gpd`

Other weibullgpd: `dweibullgpd`, `fweibullgpd`, `pweibullgpd`, `qweibullgpd`, `rweibullgpd`, `weibullgpd`

---

lweibullgpdcon *Log-likelihood of Weibull Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Log-likelihood and negative log-likelihood for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint.

Usage

```
lweibullgpdcon(x, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale), xi = 0, phiu = TRUE,
  log = TRUE)

nlweibullgpdcon(pvector, x, phiu = TRUE,
  finitelik = FALSE)
```

Arguments

x	vector of sample data
wshape	Weibull shape (non-negative)
wscale	Weibull scale (non-negative)
u	threshold (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1] or logical
log	logical, if TRUE then log density
pvector	vector of initial values mixture model parameters (wshape, wscale, u, sigmau, xi) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters

Details

The likelihood functions for the extreme value mixture model with Weibull bulk and GPD tail, as used in the maximum likelihood fitting function fweibullgpdcon.

Non-positive data are ignored.

They are designed to be used for MLE in fweibullgpdcon but are available for wider usage, e.g. constructing your own extreme value mixture models.

See fweibullgpdcon and fgpd for full details.

Log-likelihood calculations are carried out in lweibullgpdcon, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for lweibullgpdcon, designed towards making it useable for optimisation (e.g. parameters are given a vector as first input). The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations.

Unlike the distribution functions weibullgpdcon the phiu must be either logical (TRUE or FALSE) or numerical in range $(0, 1)$. The default is to specify phiu=TRUE so that the tail fraction is specified by Weibull distribution $\phi_u = 1 - H(u)$, or phiu=FALSE to treat the tail fraction as an extra parameter estimated using the sample proportion. Specify a numeric phiu as pre-specified probability $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

The function lweibullgpdcon carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

Value

lweibullgpdcon gives (log-)likelihood and nlweibullgpdcon gives the negative log-likelihood.

Note

Unlike all the distribution functions for this mixture model, the likelihood functions only permits a scalar value for all the parameters. Only the data is a vector.

A default value for the tail fraction phiu=TRUE is given in both lweibullgpdcon and nlnormgpd. The lweibullgpdcon also has the usual defaults for the other parameters, but nlweibullgpdcon has no defaults.

Invalid parameters will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

See fgpd for explanation of finitelik.

Error checking of the inputs is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

lweibullgpd, lgpd and gpd

Other weibullgpdcon: dweibullgpdcon, fweibullgpdcon, pweibullgpdcon, qweibullgpdcon, rweibullgpdcon, weibullgpdcon

---

mgammagpd *Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with mixture of gammas for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the gamma shape gshape and scale gscale, threshold u GPD scale sigmau and shape xi and tail fraction phiu.

Usage

```
dmgammagpd(x, mgshape = list(1), mgscale = list(1),
  mgweights = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]),
  sigmau = sqrt(mgshape[[1]]) * mgscale[[1]], xi = 0,
  phiu = TRUE, log = FALSE)

pmgammagpd(q, mgshape = list(1), mgscale = list(1),
  mgweights = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]),
  sigmau = sqrt(mgshape[[1]]) * mgscale[[1]], xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qmgammagpd(p, mgshape = list(1), mgscale = list(1),
  mgweights = NULL,
```

```
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]),
  sigmau = sqrt(mgshape[[1]]) * mgscale[[1]], xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rmgammagpd(n = 1, mgshape = list(1), mgscale = list(1),
  mgweights = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]),
  sigmau = sqrt(mgshape[[1]]) * mgscale[[1]], xi = 0,
  phiu = TRUE)
```

Arguments

mgshape	mgamma shape (non-negative) as list
mgscale	mgamma scale (non-negative) as list
mgweights	mgamma weights (positive) as list or NULL
x	quantile
u	threshold (non-negative)
sigmau	scale parameter (non-negative)
xi	shape parameter
phiu	probability of being above threshold [0,1] or TRUE
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining mixture of gammas for the bulk below the threshold and GPD for upper tail. The parameters are input as a list, with one parameter object in the list for each gamma component. There must be the same number of components in mgshape and mgscale. The number of objects in the parameters lists determines the number of components. The parameter object for each gamma component can either be a scalar or vector, consistent with the other mixture models

If mgweights=NULL then assumes equal weights for each component. Otherwise, mgweights must be a list of the same length as mgshape and mgscale, filled with positive values. In the latter case, the weights are rescaled to sum to unity.

The user can pre-specify phiu permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the gamma bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the gamma bulk model (phiu=TRUE), upto the threshold $0 < x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the mixture of gammas and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 < x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The gamma is defined on the non-negative reals, so the threshold must be non-negative.

See gammagpd for details of simpler parametric mixture model with single gamma for bulk component and GPD for upper tail.

Value

dmgammagpd gives the density, pmgammagpd gives the cumulative distribution function, qmgammagpd gives the quantile function and rmgammagpd gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`, and the parameters can be vectorised within the list. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rmgammagpd` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for rmgammagpd is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Gamma_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

do Nascimento, F.F., Gamerman, D. and Lopes, H.F. (2011). A semiparametric Bayesian approach to extreme value estimation. Statistical Computing, 22(2), 661-675.

See Also

gammagpd, mgammagpd, gpd and dgamma

`mrlplot` *Mean Residual Life Plot*

Description

Plots the sample mean residual life (MRL) plot.

Usage

```
mrlplot(data, tlim = NULL, nt = min(100, length(data)),
  p.or.n = FALSE, alpha = 0.05, ylim = NULL,
  legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Mean Residual Life Plot", xlab = "Threshold u",
  ylab = "Mean Excess", ...)
```

Arguments

`data`	vector of sample data
`tlim`	vector of (lower, upper) limits of range of threshold to plot MRL, or `NULL` to use default values
`nt`	number of thresholds for which to evaluate MRL
`alpha`	logical, significance level (0, 1)
`p.or.n`	logical, should tail fraction (`FALSE`) or number of exceedances (`TRUE`) be given on upper x-axis
`ylim`	y-axis limits or `NULL`
`legend.loc`	location of legend (see `legend`)
`try.thresh`	vector of threshold to fit GPD using MLE and show theoretical MRL
`main`	title of plot
`xlab`	x-axis label
`ylab`	y-axis label
`...`	further arguments to be passed to the plotting functions

Details

Plots the sample mean residual life plot, which is also known as the mean excess plot.

The mean residual life above a threshold u is given by `mean(x[x > u]) - u`, i.e. the sample mean of the exceedances less the threshold. If the generalised Pareto distribution (GPD)is an appropriate model for the excesses above u, then for any higher thresholds $v > u$ the MRL will be linear with intercept $(\sigma_u - \xi * u)/(1 - \xi)$ and gradient $\xi/(1 - \xi)$.

Symmetric central limit theorem based confidence intervals are provided for all mean excesses, provided there are at least 5 exceedances. The sampling density for the MRL is shown by a greyscale image, where lighter greys indicate low density.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The GPD is fitted to the excesses using maximum likelihood estimation. The estimated parameters are used to plot the linear function for all higher thresholds using a solid line. The threshold should set as low as possible,

so a dashed line is shown below the pre-chosen threshold. If the MRL is similar to the dashed line then a lower threshold may be chosen.

If no threshold limits are provided `tlim = NULL` then the lowest threshold is set to be just below the median data point and the maximum threshold is set to the 6th largest datapoint.

The range of permitted thresholds is just below the minimum datapoint and the second largest value. If there are less unique values of data within the threshold range than the number of threshold evalations requested, then instead of a sequence of thresholds they will be set to each unique datapoint, i.e. the MRL will only be evaluated where there is data.

The missing (`NA` and `NaN`) and non-finite values are ignored.

The lower x-axis is the threshold and an upper axis either gives the number of exceedances (`p.or.n = FALSE`) or proportion of excess (`p.or.n = TRUE`). Note that unlike the gpd related functions the missing values are ignored, so do not add to the lower tail fraction. But ignoring the missing values is consistent with all the other mixture model functions.

Value

`mrlplot` gives the mean residual life plot. It also returns a matrix containing columns of the threshold, number of exceedances, mean excess, standard devation of excesses and $100(1-\alpha)\%$ confidence interval. The standard deviation and confidence interval are `NA` for less than 5 exceedances.

Note

If the user specifies the threshold range, the thresholds above the second largest are dropped. A warning message is given if any thresholds have at most 5 exceedances, in which case the confidence interval is not calculated as it is unreliable due to small sample. If there are less than 10 exceedances of the minimum threshold then the function will stop.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Coles S.G. (2004). An Introduction to the Statistical Modelling of Extreme Values. Springer-Verlag: London.

See Also

`gpd` and `mrlplot` from `evd` library

Examples

```
x = rnorm(1000)
mrlplot(x)
mrlplot(x, tlim = c(0, 2.2))
mrlplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))
mrlplot(x, tlim = c(0, 3), try.thresh = c(0.5, 1, 1.5))
```

normgpd *Normal Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the normal mean `nmean` and standard deviation `nsd`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dnormgpd(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
  phiu = TRUE, log = FALSE)

pnormgpd(q, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qnormgpd(p, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rnormgpd(n = 1, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
  phiu = TRUE)
```

Arguments

`nmean`	normal mean
`nsd`	normal standard deviation (non-negative)
`phiu`	probability of being above threshold [0,1] or TRUE
`x`	quantile
`u`	threshold
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the normal bulk model (`phiu=TRUE`), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x, nmean, nsd)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

See `gpd` for details of GPD upper tail component and `dnorm` for details of normal bulk component.

Value

`dnormgpd` gives the density, `pnormgpd` gives the cumulative distribution function, `qnormgpd` gives the quantile function and `rnormgpd` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rnormgpd` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rnormgpd` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The normal mean `nmean` and GPD threshold `u` will also require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd and dnorm

Other normgpd: fnormgpd, lnormgpd, nlnormgpd

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnormgpd(1000)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx))

# three tail behaviours
plot(xx, pnormgpd(xx), type = "l")
lines(xx, pnormgpd(xx, xi = 0.3), col = "red")
lines(xx, pnormgpd(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rnormgpd(1000, phiu = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx, phiu = 0.2))

plot(xx, dnormgpd(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dnormgpd(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dnormgpd(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

normgpdcon *Normal Bulk and GPD Tail Extreme Value Mixture Model with a Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint. The parameters are the normal mean `nmean` and standard deviation `nsd`, threshold `u` and GPD shape `xi` and tail fraction `phiu`.

Usage

```
dnormgpdcon(x, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, phiu = TRUE,
  log = FALSE)

pnormgpdcon(q, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qnormgpdcon(p, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rnormgpdcon(n = 1, nmean = 0, nsd = 1,
  u = qnorm(0.9, nmean, nsd), xi = 0, phiu = TRUE)
```

Arguments

x	quantile
nmean	normal mean
nsd	normal standard deviation (non-negative)
u	threshold
xi	shape parameter
phiu	probability of being above threshold [0,1] or TRUE
log	logical, if TRUE then log density
q	quantile
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail with a continuity constraint. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the normal bulk model (`phiu=TRUE`), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x, nmean, nsd)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the normal and conditional GPD density functions (i.e. `dnorm(x, nmean, nsd)` and `dgpd(x, u, sigmau, xi)`). The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u)/[1 - \phi_u]h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)]/h(u)$$

.

See gpd for details of GPD upper tail component and dnorm for details of normal bulk component.

Value

dnormgpdcon gives the density, pnormgpdcon gives the cumulative distribution function, qnormgpdcon gives the quantile function and rnormgpdcon gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rnormgpdcon` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for rnormgpdcon is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The normal mean `nmean` and GPD threshold `u` will also require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd, dnorm and dnormgpd

Other normgpdcon: fnormgpdcon, lnormgpdcon, nlnormgpdcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnormgpdcon(1000)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpdcon(xx))

# three tail behaviours
plot(xx, pnormgpdcon(xx), type = "l")
lines(xx, pnormgpdcon(xx, xi = 0.3), col = "red")
lines(xx, pnormgpdcon(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rnormgpdcon(1000, phiu = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpdcon(xx, phiu = 0.2))

plot(xx, dnormgpdcon(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dnormgpdcon(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dnormgpdcon(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

tcplot *Parameter Threshold Stability Plots*

Description

Plots the sample mean residual life (MRL) plot.

Usage

```
tcplot(data, tlim = NULL, nt = min(100, length(data)),
  p.or.n = FALSE, alpha = 0.05, ylim.xi = NULL,
  ylim.sigmau = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE), ...)

tshapeplot(data, tlim = NULL,
  nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Shape Threshold Stability Plot",
```

```
  xlab = "Threshold u", ylab = "Shape Parameter", ...)

tscaleplot(data, tlim = NULL,
  nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Modified Scale Threshold Stability Plot",
  xlab = "Threshold u",
  ylab = "Modified Scale Parameter", ...)
```

Arguments

`ylim.xi`	y-axis limits for shape parameter or `NULL`
`ylim.sigmau`	y-axis limits for scale parameter or `NULL`
`data`	vector of sample data
`tlim`	vector of (lower, upper) limits of range of threshold to plot MRL, or `NULL` to use default values
`nt`	number of thresholds for which to evaluate MRL
`p.or.n`	logical, should tail fraction (`FALSE`) or number of exceedances (`TRUE`) be given on upper x-axis
`alpha`	logical, significance level (0, 1)
`legend.loc`	location of legend (see `legend`)
`try.thresh`	vector of threshold to fit GPD using MLE and show theoretical MRL
`...`	further arguments to be passed to the plotting functions
`ylim`	y-axis limits or `NULL`
`main`	title of plot
`xlab`	x-axis label
`ylab`	y-axis label

Details

The MLE of the (modified) GPD scale and shape (xi) parameters are plotted against as reange of possible threshold. Known as the threshold stability plots (Coles, 2001). The modified scale parameter is $\sigma_u - u\xi$. If the GPD is a suitable model for a threshold u then for all higher threshold $v > u$ it will also be suitable, with the shape and modified scale being constant.

In practice there is sample uncertainty in the parameter estimates, which must be taken into account when choosing a threshold.

The usual asymptotic Wald confidence intervals are shown based on the observed information matrix to measure this undertainty. The sampling density of the Wald normal approximation is shown by a greyscale image, where lighter greys indicate low density.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The GPD is fitted to the excesses using maximum likelihood estimation. The estimated parameters are plot as a horizontal line which is solid above this threshold where the parameter from smaller tail fraction should be the same if the GPD is a good model (upto sample uncertainty). The threshold should always be chosen to be as low as possible to reduce sample uncertainty. Therefore, below the pre-chosen threshold, where the GPD should not be a good model, the line is dashed and the parameter estimates should now deviate from the dashed line (otherwise a lower threshold could be used). If no threshold limits are provided `tlim = NULL` then the lowest threshold is set to be just below the median data point and

the maximum threshold is set to the 11th largest datapoint. This is a slightly lower order statistic compared to that used in the MRL plot `mrlplot` function to account for the fact the maximum likelihood estimation is likely to be very unreliable with 10 or fewer datapoints.

The range of permitted thresholds is just below the minimum datapoint and the second largest value. If there are less unique values of data within the threshold range than the number of threshold evalations requested, then instead of a sequence of thresholds they will be set to each unique datapoint, i.e. MLE will only be applied where there is data.

The missing (`NA` and `NaN`) and non-finite values are ignored.

The lower x-axis is the threshold and an upper axis either gives the number of exceedances (`p.or.n = FALSE`) or proportion of excess (`p.or.n = TRUE`). Note that unlike the gpd related functions the missing values are ignored, so do not add to the lower tail fraction. But ignoring the missing values is consistent with all the other mixture model functions.

Value

`tshapeplot` and `tscaleplot` produces the threshold stability plot for the shape and scale parameter respectively. They also returns a matrix containing columns of the threshold, number of exceedances, MLE shape/scale and their standard devation and $100(1-\alpha)\%$ Wald confidence interval. Where the observed information matrix is not obtainable the standard deviation and confidence intervals are `NA`. For the `tscaleplot` the modified scale quantities are also provided. `tcplot` produces both plots on one graph and outputs a merged dataframe of results.

Note

If the user specifies the threshold range, the thresholds above the sixth largest are dropped. A warning message is given if any thresholds have at most 10 exceedances, in which case the maximum likelihood estimation is unreliable. If there are less than 10 exceedances of the minimum threshold then the function will stop.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Coles S.G. (2004). An Introduction to the Statistical Modelling of Extreme Values. Springer-Verlag: London.

See Also

`mrlplot` and `tcplot` from `evd` library

Examples

```
## Not run:
x = rnorm(1000)
tcplot(x)
```

```
tshapeplot(x, tlim = c(0, 2))
tscaleplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))
tcplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))

## End(Not run)
```

`weibullgpd` *Weibull Bulk and GPD Tail Extreme Value Mixture Model*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the Weibull shape `wshape` and scale `wscale`, threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

Usage

```
dweibullgpd(x, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, phiu = TRUE, log = FALSE)

pweibullgpd(q, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, phiu = TRUE, lower.tail = TRUE)

qweibullgpd(p, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, phiu = TRUE, lower.tail = TRUE)

rweibullgpd(n = 1, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  xi = 0, phiu = TRUE)
```

Arguments

`wshape`	Weibull shape (non-negative)
`wscale`	Weibull scale (non-negative)
`u`	threshold (non-negative)
`x`	quantile
`sigmau`	scale parameter (non-negative)
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or TRUE
`log`	logical, if TRUE then log density
`q`	quantile

lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probability
n	sample size (non-negative integer)

Details

Extreme value mixture model combining Weibull distribution for the bulk below the threshold and GPD for upper tail. The user can pre-specify phiu permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the Weibull bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the Weibull bulk model (phiu=TRUE), upto the threshold $0 \leq x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the Weibull and conditional GPD cumulative distribution functions (i.e. pweibull(x, wshape, wscale) and pgpd(x, u, sigmau, xi)).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 \leq x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The gamma is defined on the positive reals, so the threshold must be positive.

See gpd for details of GPD upper tail component and dweibull for details of Weibull bulk component.

Value

dweibullgpd gives the density, pweibullgpd gives the cumulative distribution function, qweibullgpd gives the quantile function and rweibullgpd gives a random sample.

Note

All inputs are vectorised except log and lower.tail. The main inputs (x, p or q) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of rweibullgpd any input vector must be of length n.

Default values are provided for all inputs, except for the fundamentals x, q and p. The default sample size for rweibullgpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x and q are passed through as is and infinite values are set to NA.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd and dweibull

Other weibullgpd: fweibullgpd, lweibullgpd, nlweibullgpd

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rweibullgpd(1000)
xx = seq(-1, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx))

# three tail behaviours
plot(xx, pweibullgpd(xx), type = "l")
lines(xx, pweibullgpd(xx, xi = 0.3), col = "red")
lines(xx, pweibullgpd(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rweibullgpd(1000, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx, phiu = 0.2))

plot(xx, dweibullgpd(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dweibullgpd(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dweibullgpd(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

weibullgpdcon *Weibull Bulk and GPD Tail Extreme Value Mixture Model with Continuity Constraint*

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold with a continuity constraint. The parameters are the Weibull shape `wshape` and scale `wscale`, threshold `u` and GPD shape `xi` and tail fraction `phiu`.

Usage

```
dweibullgpdcon(x, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale), xi = 0, phiu = TRUE,
  log = FALSE)

pweibullgpdcon(q, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qweibullgpdcon(p, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rweibullgpdcon(n = 1, wshape = 1, wscale = 1,
  u = qweibull(0.9, wshape, wscale), xi = 0, phiu = TRUE)
```

Arguments

`x`	quantile
`wshape`	Weibull shape (non-negative)
`wscale`	Weibull scale (non-negative)
`u`	threshold (non-negative)
`xi`	shape parameter
`phiu`	probability of being above threshold [0,1] or TRUE
`log`	logical, if TRUE then log density
`q`	quantile
`lower.tail`	logical, if FALSE then upper tail probabilities
`p`	cumulative probability
`n`	sample size (non-negative integer)

Details

Extreme value mixture model combining Weibull distribution for the bulk below the threshold and GPD for upper tail with a continuity constraint. The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the Weibull bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the Weibull bulk model (`phiu=TRUE`), upto the threshold $0 \leq x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the Weibull and conditional GPD cumulative distribution functions (i.e. `pweibull(x, wshape, wscale)` and `pgpd(x, u, sigmau, xi)`).

The cumulative distribution function for pre-specified ϕ_u, upto the threshold $0 \leq x \leq u$, is given by:

$$F(x) = (1 - \phi_u)H(x)/H(u)$$

and above the threshold $x > u$:

$$F(x) = \phi_u + [1 - \phi_u]G(x)$$

Notice that these definitions are equivalent when $\phi_u = 1 - H(u)$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the Weibull and conditional GPD density functions (i.e. `dweibull(x, wshape, wscale)` and `dgpd(x, u, sigmau, xi)`). The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u)/[1 - \phi_u]h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)]/h(u)$$

.

The gamma is defined on the positive reals, so the threshold must be positive.

See `gpd` for details of GPD upper tail component and `dweibull` for details of Weibull bulk component.

Value

`dweibullgpdcon` gives the density, `pweibullgpdcon` gives the cumulative distribution function, `qweibullgpdcon` gives the quantile function and `rweibullgpdcon` gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rweibullgpdcon` any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for `rweibullgpdcon` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x` and `q` are passed through as is and infinite values are set to `NA`.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from http://www.ine.pt/revstat/pdf/rs120102.pdf

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

gpd, dweibull and dweibullgpd

Other weibullgpdcon: fweibullgpdcon, lweibullgpdcon, nlweibullgpdcon

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rweibullgpdcon(1000)
xx = seq(-1, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpdcon(xx))

# three tail behaviours
plot(xx, pweibullgpdcon(xx), type = "l")
lines(xx, pweibullgpdcon(xx, xi = 0.3), col = "red")
lines(xx, pweibullgpdcon(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =",c(0, 0.3, -0.3)),
  col=c("black", "red", "blue"), lty = 1)

x = rweibullgpdcon(1000, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpdcon(xx, phiu = 0.2))

plot(xx, dweibullgpdcon(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dweibullgpdcon(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dweibullgpdcon(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

Index