# R documentation

of all in 'man'

September 3, 2019

## R topics documented:

---

`evmix-package` *Extreme Value Mixture Modelling, Threshold Estimation and Boundary Corrected Kernel Density Estimation*

---

## Description

Functions for Extreme Value Mixture Modelling, Threshold Estimation and Boundary Corrected Kernel Density Estimation

## Details

| | |
|---|---|
| Package: | evmix |
| Type: | Package |
| Version: | 2.12 |
| Date: | 2019-09-02 |
| 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 and a wide choice of kernels, 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)

Carl Scarrott, Yang Hu and Alfadino Akbar, University of Canterbury, New Zealand <carl.scarrott@canterbury.ac.n

## 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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

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 Using a Variety of Approaches*

---

## Description

Density, cumulative distribution function, quantile function and random number generation for boundary corrected kernel density estimators using a variety of approaches (and different kernels) with a constant bandwidth `lambda`.

## Usage

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

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

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

rbckden(n = 1, kerncentres, lambda = NULL, bw = NULL,
  kernel = "gaussian", bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = NULL, xmax = NULL)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default` = `"gaussian"`) |
| `bcmethod` | boundary correction method |
| `proper` | logical, whether density is renormalised to integrate to unity (where needed) |
| `nn` | non-negativity correction method (simple boundary correction only) |
| `offset` | offset added to kernel centres (logtrans only) or `NULL` |
| `xmax` | upper bound on support (copula and beta kernels only) or `NULL` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Boundary corrected kernel density estimation (BCKDE) with improved bias properties near the boundary compared to standard KDE available in `kden` functions. The user chooses from a wide range of boundary correction methods designed to cope with a lower bound at zero and potentially also both upper and lower bounds.

Some boundary correction methods require a secondary correction for negative density estimates of which two methods are implemented. Further, some methods don't necessarily give a density which integrates to one, so an option is provided to renormalise to be proper.

It assumes there is a lower bound at zero, so prior transformation of data is required for a alternative lower bound (possibly including negation to allow for an upper bound).

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

Certain boundary correction methods use the standard kernels which are defined in the kernels help documentation with the "gaussian" as the default choice.

The quantile function is rather complicated as there is no closed form solution, so is obtained by numerical approximation of the inverse cumulative distribution function $P(X \leq q) = p$ to find $q$. The quantile function qbckden evaluates the KDE cumulative distribution function over the range from c(0,max(kerncentre) + lambda), or c(0,max(kerncentre) + 5*lambda) for normal kernel. Outside of this range the quantiles are set to 0 for lower tail and Inf (or xmax where appropriate) for upper tail. A sequence of values of length fifty times the number of kernels (upto a maximum of 1000) 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 implemented. Hence, a bandwidth must always be specified and you should consider using fbckden function for cross-validation MLE for bandwidth.

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.

## Boundary Correction Methods

Renormalisation to a proper density is assumed by default proper=TRUE. This correction is needed for bcmethod="renorm", "simple", "beta1", "beta2", "gamma1" and "gamma2" which all require numerical integration. Renormalisation will not be carried out for other methods, even when proper=TRUE.

Non-negativity correction is only relevant for the bcmethod="simple" approach. The Jones and Foster (1996) method is applied nn="jf96" by default. This method can occassionally give an extra boundary bias for certain populations (e.g. Gamma(2, 1)), see paper for details. Non-negative values can simply be zeroed (nn="zero"). Renormalisation should always be applied after non-negativity correction. Non-negativity correction will not be carried out for other methods, even when requested by user.

The non-negative correction is applied before renormalisation, when both requested.

The boundary correction methods implemented are listed below. The first set can use any type of kernel (see kernels help documentation):

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. Renormalisation and non-negativity correction may be required.

bcmethod="cutnorm" applies cut and normalisation method of Gasser and Muller (1979), where the kernels themselves are individually truncated at the boundary and renormalised to unity.

bcmethod="renorm" applies first order correction method discussed in Diggle (1985), where the kernel density estimate is locally renormalised near boundary. Renormalisation may be required.

bcmethod="reflect" applies 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 cause extra artefacts at the boundary.

bcmethod="logtrans" applies KDE on the log-scale and then back-transforms (with explicit normalisation) following Marron and Ruppert (1992). This is the approach implemented in the ks package. As the KDE is applied on the log scale, the effective bandwidth on the original scale is non-constant. The offset option is only used for this method and is commonly used to offset zero kernel centres in log transform to prevent log(0).

All the following boundary correction methods do not use kernels in their usual sense, so ignore the kernel input:

bcmethod="beta1" and "beta2" uses the beta and modified beta kernels of Chen (1999) respectively. The xmax rescales the beta kernels to be defined on the support [0, xmax] rather than unscaled [0, 1]. Renormalisation will be required.

bcmethod="gamma1" and "gamma2" uses the gamma and modified gamma kernels of Chen (2000) respectively. Renormalisation will be required.

bcmethod="copula" uses the bivariate normal copula based kernesl of Jones and Henderson (2007). As with the bcmethod="beta1" and "beta2" methods the xmax rescales the copula kernels to be defined on the support [0, xmax] rather than [0, 1]. In this case the bandwidth is defined as $lambda = 1 - \rho^2$, so the bandwidth is limited to $(0, 1)$.

## Warning

The "simple", "renorm", "beta1", "beta2", "gamma1" and "gamma2" boundary correction methods may require renormalisation using numerical integration which can be very slow. In particular, the numerical integration is extremely slow for the kernel="uniform", due to the adaptive quadrature in the integrate function being particularly slow for functions with step-like behaviour.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of 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 not NULL for in other methods. The offset option is only relevant for the "logtrans" method, so a warning is produced if this is not NULL for in other methods.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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/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.

Gasser, T. and Muller, H. (1979). Kernel estimation of regression functions. In "Lecture Notes in Mathematics 757, edited by Gasser and Rosenblatt, Springer.

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

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kden: fbckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other bckden: bckdengpdcon, bckdengpd, fbckdengpdcon, fbckdengpd, fbckden, fkden, kden

Other bckdengpd: bckdengpdcon, bckdengpd, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpdcon, bckdengpd, fbckdengpdcon, fbckdengpd, fbckden, fkdengpdcon, gkgcon, kdengpdcon

Other fbckden: fbckden

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

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 = 1), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
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", ylim = c(0, 0.8))
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.1, bcmethod = "beta2", proper = TRUE, 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=1000
x = rgamma(n, shape = 1, scale = 2)
xx = seq(-0.5, 15, 0.01)
plot(xx, dgamma(xx, shape = 1, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = TRUE),
  lwd = 2, col = "purple")
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = FALSE),
  lwd = 2, col = "red", lty = 2)
legend("topright", c("True Density", "Simple BC with renomalisation",
"Simple BC without renomalisation"),
lty = 1, lwd = c(1, 2, 2), col = c("black", "purple", "red"))

## End(Not run)
```

---

bckdengpd *Boundary Corrected Kernel Density Estimate 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 boundary corrected kernel density estimate for 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, bw = NULL,
  kernel = "gaussian", bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = NULL, xmax = NULL, log = FALSE)

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

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `bcmethod` | boundary correction method |
| `proper` | logical, whether density is renormalised to integrate to unity (where needed) |
| `nn` | non-negativity correction method (simple boundary correction only) |
| `offset` | offset added to kernel centres (logtrans only) or `NULL` |
| `xmax` | upper bound on support (copula and beta kernels only) or `NULL` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining boundary corrected kernel density (BCKDE) estimate for the bulk below the threshold and GPD for upper tail. The user chooses from a wide range of boundary correction methods designed to cope with a lower bound at zero and potentially also both upper and lower bounds.

Some boundary correction methods require a secondary correction for negative density estimates of which two methods are implemented. Further, some methods don't necessarily give a density which integrates to one, so an option is provided to renormalise to be proper.

It assumes there is a lower bound at zero, so prior transformation of data is required for a alternative lower bound (possibly including negation to allow for an upper bound).

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the BCKDE bulk model.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

The cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the BCKDE (`phiu=TRUE`), upto the threshold $x \le u$, 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 BCKDE and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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 $\phi_u = 1 - H(u)$.

Unlike the standard KDE, there is no general rule-of-thumb bandwidth for all the BCKDE, with only certain methods having a guideline in the literature, so none have been implemented. Hence, a bandwidth must always be specified and you should consider using `fbckdengpd` of `fbckden` function for cross-validation MLE for bandwidth.

See `gpd` for details of GPD upper tail component and `dbckden` for details of BCKDE bulk component.

## Value

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

## Boundary Correction Methods

See `dbckden` for details of BCKDE methods.

## Warning

The "simple", "renorm", "beta1", "beta2", "gamma1" and "gamma2" boundary correction methods may require renormalisation using numerical integration which can be very slow. In particular, the numerical integration is extremely slow for the kernel="uniform", due to the adaptive quadrature in the integrate function being particularly slow for functions with step-like behaviour.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of 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.

The xmax option is only relevant for the beta and copula methods, so a warning is produced if this is not NULL for in other methods. The offset option is only relevant for the "logtrans" method, so a warning is produced if this is not NULL for in other methods.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters or kernel centres.

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.

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

### See Also

gpd, kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kdengpd: fbckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, gkg, kdengpdcon, kdengpd, kden

Other bckden: bckdengpdcon, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkden, kden

Other bckdengpd: bckdengpdcon, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpdcon, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpdcon, gkgcon, kdengpdcon

Other fbckdengpd: fbckdengpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

kerncentres=rgamma(500, shape = 1, scale = 2)
xx = seq(-0.1, 10, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)")
abline(v = quantile(kerncentres, 0.9))

plot(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", type = "l")
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, xi = 0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "red")
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, xi = -0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", 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, bcmethod = "reflect")
xx = seq(0.01, 3.5, 0.01)
hist(x, breaks = 100, freq = FALSE)
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, phiu = TRUE, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)")

lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, xi=-0.2, phiu = 0.1, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)", col = "red")
lines(xx, dbckdengpd(xx, kerncentres, lambda = 0.1, xi=0.2, phiu = 0.1, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)", col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

bckdengpdcon *Boundary Corrected Kernel Density Estimate and GPD Tail Extreme Value Mixture Model With Single Continuity Constraint*

## Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with boundary corrected kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the bandwidth `lambda`, threshold `u` GPD shape `xi` and tail fraction `phiu`.

## Usage

```
dbckdengpdcon(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL,
  log = FALSE)

pbckdengpdcon(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL,
  lower.tail = TRUE)

qbckdengpdcon(p, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL,
  lower.tail = TRUE)

rbckdengpdcon(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `u` | threshold |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `bcmethod` | boundary correction method |
| `proper` | logical, whether density is renormalised to integrate to unity (where needed) |

| | |
|---|---|
| `nn` | non-negativity correction method (simple boundary correction only) |
| `offset` | offset added to kernel centres (logtrans only) or `NULL` |
| `xmax` | upper bound on support (copula and beta kernels only) or `NULL` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining boundary corrected kernel density (BCKDE) estimate for the bulk below the threshold and GPD for upper tail with continuity at threshold. The user chooses from a wide range of boundary correction methods designed to cope with a lower bound at zero and potentially also both upper and lower bounds.

Some boundary correction methods require a secondary correction for negative density estimates of which two methods are implemented. Further, some methods don't necessarily give a density which integrates to one, so an option is provided to renormalise to be proper.

It assumes there is a lower bound at zero, so prior transformation of data is required for a alternative lower bound (possibly including negation to allow for an upper bound).

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the BCKDE bulk model.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

The cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the BCKDE (`phiu=TRUE`), upto the threshold $x \le u$, 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 BCKDE and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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 BCKDE and conditional GPD density functions respectively. 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

$$\sigma_u = [1 - H(u)]/h(u)$$

.

Unlike the standard KDE, there is no general rule-of-thumb bandwidth for all the BCKDE, with only certain methods having a guideline in the literature, so none have been implemented. Hence, a bandwidth must always be specified and you should consider using fbckdengpdcon of fbckden function for cross-validation MLE for bandwidth.

See gpd for details of GPD upper tail component and dbckden for details of BCKDE bulk component.

### Value

dbckdengpdcon gives the density, pbckdengpdcon gives the cumulative distribution function, qbckdengpdcon gives the quantile function and rbckdengpdcon gives a random sample.

### Boundary Correction Methods

See dbckden for details of BCKDE methods.

### Warning

The "simple", "renorm", "beta1", "beta2", "gamma1" and "gamma2" boundary correction methods may require renormalisation using numerical integration which can be very slow. In particular, the numerical integration is extremely slow for the kernel="uniform", due to the adaptive quadrature in the integrate function being particularly slow for functions with step-like behaviour.

### Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

### Note

Unlike most of the other extreme value mixture model functions the bckdengpdcon 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 rbckdengpdcon is 1.

The xmax option is only relevant for the beta and copula methods, so a warning is produced if this is not NULL for in other methods. The offset option is only relevant for the "logtrans" method, so a warning is produced if this is not NULL for in other methods.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters or kernel centres.

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.

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

gpd, kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kdengpdcon: fbckdengpdcon, fgkgcon, fkdengpdcon, fkdengpd, gkgcon, kdengpdcon, kdengpd

Other bckden: bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkden, kden

Other bckdengpd: bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpdcon, gkgcon, kdengpdcon

Other fbckdengpdcon: fbckdengpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

kerncentres=rgamma(500, shape = 1, scale = 2)
xx = seq(-0.1, 10, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)")
abline(v = quantile(kerncentres, 0.9))

plot(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", type = "l")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = 0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "red")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = -0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", 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 = rbckdengpdcon(1000, kerncentres, lambda = 0.1, phiu = TRUE, bcmethod = "reflect")
xx = seq(0.01, 3.5, 0.01)
hist(x, breaks = 100, freq = FALSE)
lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, phiu = TRUE, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)")

lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, xi=-0.2, phiu = 0.1, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)", col = "red")
lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, xi=0.2, phiu = 0.1, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)", 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 1 `bshape1` and shape 2 `bshape2`, 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

| | |
|---|---|
| `x` | quantiles |

| | |
|---|---|
| bshape1 | beta shape 1 (positive) |
| bshape2 | beta shape 2 (positive) |
| u | threshold over $(0, 1)$ |
| sigmau | scale parameter (positive) |
| xi | shape parameter |
| phiu | probability of being above threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive 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 $\phi_u$. Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the beta bulk model.

The usual beta distribution is defined over $[0, 1]$, 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 $\phi_u$ defined by the upper tail fraction of the beta bulk model (phiu=TRUE), upto the threshold $0 \le x \le u < 1$, 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,bshape1,bshape2) and pgpd(x,u,sigmau,xi)).

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 \le x \le u < 1$, 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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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: `betagpdcon`, `fbetagpdcon`, `fbetagpd`

Other betagpdcon: `betagpdcon`, `fbetagpdcon`, `fbetagpd`

Other fbetagpd: `fbetagpd`

## Examples

```
## Not run:
set.seed(1)
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("bottomright", 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)
```

---

betagpdcon *Beta Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

---

## 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 with continuity at threshold. The parameters are the beta shape 1 `bshape1` and shape 2 `bshape2`, threshold `u` GPD shape `xi` and tail fraction `phiu`.

## Usage

```
dbetagpdcon(x, bshape1 = 1, bshape2 = 1, u = qbeta(0.9, bshape1,
  bshape2), xi = 0, phiu = TRUE, log = FALSE)

pbetagpdcon(q, bshape1 = 1, bshape2 = 1, u = qbeta(0.9, bshape1,
  bshape2), xi = 0, phiu = TRUE, lower.tail = TRUE)

qbetagpdcon(p, bshape1 = 1, bshape2 = 1, u = qbeta(0.9, bshape1,
  bshape2), xi = 0, phiu = TRUE, lower.tail = TRUE)

rbetagpdcon(n = 1, bshape1 = 1, bshape2 = 1, u = qbeta(0.9,
  bshape1, bshape2), xi = 0, phiu = TRUE)
```

## Arguments

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

## Details

Extreme value mixture model combining beta distribution for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the beta bulk model.

The usual beta distribution is defined over $[0, 1]$, 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 $\phi_u$ defined by the upper tail fraction of the beta bulk model (`phiu=TRUE`), upto the threshold $0 \le x \le u < 1$, 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,bshape1,bshape2)` and `pgpd(x,u,sigmau,xi)`).

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 \le x \le u < 1$, 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 $\phi_u = 1 - H(u)$.

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the beta and conditional GPD density functions (i.e. `dbeta(x,bshape1,bshape2)` and `dgpd(x,u,sigmau,xi)`) respectively. 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 `dbeta` for details of beta bulk component.

## Value

`dbetagpdcon` gives the density, `pbetagpdcon` gives the cumulative distribution function, `qbetagpdcon` gives the quantile function and `rbetagpdcon` 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 `rbetagpdcon` 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 rbetagpdcon is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: betagpd, fbetagpdcon, fbetagpd

Other betagpdcon: betagpd, fbetagpdcon, fbetagpd

Other fbetagpdcon: fbetagpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

x = rbetagpdcon(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, dbetagpdcon(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2))

# three tail behaviours
plot(xx, pbetagpdcon(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2), type = "l")
lines(xx, pbetagpdcon(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2, xi = 0.3), col = "red")
lines(xx, pbetagpdcon(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 = rbetagpdcon(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, dbetagpdcon(xx, bshape1 = 2, bshape2 = 0.6, u = 0.7, phiu = 0.5))

plot(xx, dbetagpdcon(xx, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5, xi=0), type = "l")
lines(xx, dbetagpdcon(xx, bshape1 = 2, bshape2 = 0.8, u = 0.7, phiu = 0.5, xi=-0.2), col = "red")
```

```
lines(xx, dbetagpdcon(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)
```

---

checking *Internal functions for checking function input arguments*

---

## Description

Functions for checking the input arguments to functions, so that main functions are more concise. They will stop when an inappropriate input is found.

These function are visible and operable by the user. But they should be used with caution, as no checks on the input validity are carried out.

For likelihood functions you will often not want to stop on finding a non-positive values for positive parameters, in such cases use `check.param` rather than `check.posparam`.

## Usage

```
check.param(param, allowvec = FALSE, allownull = FALSE,
  allowmiss = FALSE, allowna = FALSE, allowinf = FALSE)

check.posparam(param, allowvec = FALSE, allownull = FALSE,
  allowmiss = FALSE, allowna = FALSE, allowinf = FALSE,
  allowzero = FALSE)

check.quant(x, allownull = FALSE, allowna = FALSE, allowinf = FALSE)

check.prob(prob, allownull = FALSE, allowna = FALSE)

check.n(n, allowzero = FALSE)

check.logic(logicarg, allowvec = FALSE, allowna = FALSE)

check.nparam(ns, nparam = 1, allownull = FALSE, allowmiss = FALSE)

check.inputn(inputn, allowscalar = FALSE, allowzero = FALSE)

check.text(textarg, allowvec = FALSE, allownull = FALSE)

check.phiu(phiu, allowvec = FALSE, allownull = FALSE,
  allowfalse = FALSE)

check.optim(method)

check.control(control)

check.bcmethod(bcmethod)
```

```
check.nn(nn)

check.offset(offset, bcmethod, allowzero = FALSE)

check.design.knots(beta, xrange, nseg, degree, design.knots)
```

## Arguments

| | |
|---|---|
| param | scalar or vector of parameters |
| allowvec | logical, where TRUE permits vector |
| allownull | logical, where TRUE permits NULL values |
| allowmiss | logical, where TRUE permits missing input |
| allowna | logical, where TRUE permits NA and NaN values |
| allowinf | logical, where TRUE permits +/-Inf values |
| allowzero | logical, where TRUE permits zero values (positive vs non-negative) |
| x | scalar or vector of quantiles |
| prob | scalar or vector of probability |
| n | scalar sample size |
| logicarg | logical input argument |
| ns | vector of lengths of parameter vectors |
| nparam | acceptable length of (non-scalar) vectors of parameter vectors |
| inputn | vector of input lengths |
| allowscalar | logical, where TRUE permits scalar (as opposed to vector) values |
| textarg | character input argument |
| phiu | scalar or vector of phiu (logical, NULL or 0-1 exclusive) |
| allowfalse | logical, where TRUE permits FALSE (and TRUE) values |
| method | optimisation method (see optim) |
| control | optimisation control list (see optim) |
| bcmethod | boundary correction method |
| nn | non-negativity correction method (simple boundary correction only) |
| offset | offset added to kernel centres (logtrans only) or NULL |
| beta | vector of B-spline coefficients (required) |
| xrange | vector of minimum and maximum of B-spline (support of density) |
| nseg | number of segments between knots |
| degree | degree of B-splines (0 is constant, 1 is linear, etc.) |
| design.knots | spline knots for splineDesign function |

## Value

The checking functions will stop on errors and return no value. The only exception is the check.inputn which outputs the maximum vector length.

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

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

| | |
|---|---|
| `x` | quantiles |
| `wshape` | Weibull shape (positive) |
| `wscale` | Weibull scale (positive) |
| `cmu` | Cauchy location |
| `ctau` | Cauchy scale |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `qinit` | scalar or vector of initial values for the quantile estimate |
| `n` | sample size (positive 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. `dweibull(x,wshape,wscale` and `dgpd(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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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/Cauchy_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

gpd, dcauchy and dweibull

Other fdwm: fdwm

### Examples

```
## Not run:
set.seed(1)
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, sigmau = 1, xi = 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, alpha = 0.05, N = 1000,
  legend = FALSE, ...)

rlplot(modelfit, upperfocus = TRUE, alpha = 0.05, N = 1000,
  legend = TRUE, rplim = NULL, rllim = NULL, ...)

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

pplot(modelfit, upperfocus = 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? |
| `alpha` | significance level over range (0, 1), or `NULL` for no CI |
| `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 |
| `rplim` | return period range |
| `rllim` | return level range |

## 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 default behaviour of this function is to use

$$(i - 0.5)/n$$

as the estimate for the $i$th order statistic of the given sample of size $n$.

The return level plot has the quantile ($q$ where $P(X \geq q) = p$ on the $y$-axis, for a particular survival probability $p$. The return period $t = 1/p$ is shown on the $x$-axis. The return level is given by:

$$q = u + \sigma_u[(\phi_u t)^\xi - 1]/\xi$$

for $\xi \neq 0$. But in the case of $\xi = 0$ this simplifies to

$$q = u + \sigma_u log(\phi_u t)$$

which is linear when plotted against the return period on a logarithmic scale. The special case of exponential/Type I ($\xi = 0$) upper tail behaviour will be linear on this scale. This is the same tranformation as in the GPD/POT diagnostic plot function `plot.uvevd` in the `evd` package, from which these functions were derived.

The crosses are the empirical quantiles/return levels (i.e. the ordered sample data) against their corresponding transformed empirical return period (from `ppoints`). The solid line is the theoretical return level (quantile) function using the estimated 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.uvevd 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.

## Acknowledgments

Based on the GPD/POT diagnostic function plot.uvevd in the evd package for which Stuart Coles' and Alec Stephenson's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

## 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

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, plot.uvevd and gpd.diag.

### Examples

```
## Not run:
set.seed(1)

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 Using a Variety of Approaches*

---

### Description

Maximum likelihood estimation for fitting boundary corrected kernel density estimator using a variety of approaches (and many possible kernels), by treating it as a mixture model.

### Usage

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

lbckden(x, lambda = NULL, bw = NULL, kernel = "gaussian",
  extracentres = NULL, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = NULL, xmax = NULL, log = TRUE)

nlbckden(lambda, x, bw = NULL, kernel = "gaussian",
  extracentres = NULL, bcmethod = "simple", proper = TRUE,
  nn = "jf96", offset = NULL, xmax = NULL, finitelik = FALSE)
```

### Arguments

| | |
|---|---|
| x | vector of sample data |
| linit | initial value for bandwidth (as kernel half-width) or NULL |
| bwinit | initial value for bandwidth (as kernel standard deviations) or NULL |

| | |
|---|---|
| kernel | kernel name (default = "gaussian") |
| extracentres | extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL |
| bcmethod | boundary correction method |
| proper | logical, whether density is renormalised to integrate to unity (where needed) |
| nn | non-negativity correction method (simple boundary correction only) |
| offset | offset added to kernel centres (logtrans only) or NULL |
| xmax | upper bound on support (copula and beta kernels only) or NULL |
| add.jitter | logical, whether jitter is needed for rounded kernel centres |
| 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 |
| lambda | bandwidth for kernel (as half-width of kernel) or NULL |
| bw | bandwidth for kernel (as standard deviations of kernel) or NULL |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

The boundary corrected kernel density estimator using a variety of approaches (and many possible kernels) is fitted to the entire dataset using cross-validation maximum likelihood estimation. The estimated bandwidth, variance and standard error are automatically output.

The log-likelihood and negative log-likelihood are also provided for wider usage, e.g. constructing your own extreme value mixture models or profile likelihood functions. The parameter lambda must be specified in the negative log-likelihood nlbckden.

Log-likelihood calculations are carried out in lbckden, which takes bandwidths as inputs 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. lambda given as first input).

The alternate bandwidth definitions are discussed in the kernels, with the lambda used here but bw also output. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels help documentation with the "gaussian" as the default choice.

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 implemented. Hence, a bandwidth must always be specified.

The simple, renorm, beta1, beta2 gamma1 and gamma2 density estimates require renormalisation, achieved by numerical integration, so is very time consuming.

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

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 $i$th 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`.

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

`fbckden` gives leave one out cross-validation (log-)likelihood and `lbckden` gives the negative log-likelihood. `nlbckden` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | (jittered) data vector `x` |
| `kerncentres`: actual kernel centres used `x` | |
| `init`: | `linit` for lambda |
| `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 lambda (kernel half-width) |
| `bw`: | MLE of bw (kernel standard deviations) |
| `kernel`: | kernel name |
| `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 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 the `fbckdengpd` function for a heavy upper tail. See MacDonald et al (2013).

### Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

### Note

An initial bandwidth must be provided, so `linit` and `bwinit` cannot both be `NULL`

The extra kernel centres `extracentres` can either be a vector of data or `NULL`.

Invalid parameter ranges will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

Infinite and missing sample values are dropped.

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, jitter, density and bw.nrd0

Other kden: bckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other bckden: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fkden, kden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fkdengpdcon, gkgcon, kdengpdcon

Other fbckden: bckden

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

nk=500
x = rgamma(nk, shape = 1, scale = 2)
xx = seq(-1, 10, 0.01)

# cut and normalize is very quick
fit = fbckden(x, linit = 0.2, bcmethod = "cutnorm")
hist(x, nk/5, freq = FALSE)
rug(x)
lines(xx, dgamma(xx, shape = 1, scale = 2), col = "black")
# but cut and normalize does not always work well for boundary correction
lines(xx, dbckden(xx, x, lambda = fit$lambda, bcmethod = "cutnorm"), lwd = 2, col = "red")
# Handily, the bandwidth usually works well for other approaches as well
lines(xx, dbckden(xx, x, lambda = fit$lambda, bcmethod = "simple"), lwd = 2, col = "blue")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "BC KDE using cutnorm",
  "BC KDE using simple", "KDE Using density"),
  lty = c(1, 1, 1, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "blue", "green"))

# By contrast simple boundary correction is very slow
# a crude trick to speed it up is to ignore the normalisation and non-negative correction,
# which generally leads to bandwidth being biased high
fit = fbckden(x, linit = 0.2, bcmethod = "simple", proper = FALSE, nn = "none")
hist(x, nk/5, freq = FALSE)
rug(x)
lines(xx, dgamma(xx, shape = 1, scale = 2), col = "black")
lines(xx, dbckden(xx, x, lambda = fit$lambda, bcmethod = "simple"), lwd = 2, col = "blue")
```

```
lines(density(x), lty = 2, lwd = 2, col = "green")

# but ignoring upper tail in likelihood works a lot better
q75 = qgamma(0.75, shape = 1, scale = 2)
fitnotail = fbckden(x[x <= q75], linit = 0.1,
   bcmethod = "simple", proper = FALSE, nn = "none", extracentres = x[x > q75])
lines(xx, dbckden(xx, x, lambda = fitnotail$lambda, bcmethod = "simple"), lwd = 2, col = "red")
legend("topright", c("True Density", "BC KDE using simple", "BC KDE (upper tail ignored)",
   "KDE Using density"),
   lty = c(1, 1, 1, 2), lwd = c(1, 2, 2, 2), col = c("black", "blue", "red", "green"))

## End(Not run)
```

---

fbckdengpd *MLE Fitting of Boundary Corrected Kernel Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with boundary corrected kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

lbckdengpd(x, lambda = NULL, u = 0, sigmau = 1, xi = 0,
  phiu = TRUE, bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL,
  log = TRUE)

nlbckdengpd(pvector, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, finitelik = FALSE)

proflubckdengpd(u, pvector, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)

nlubckdengpd(pvector, u, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in `useq`, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in `useq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `kernel` | kernel name (`default = "gaussian"`) |
| `bcmethod` | boundary correction method |
| `proper` | logical, whether density is renormalised to integrate to unity (where needed) |
| `nn` | non-negativity correction method (simple boundary correction only) |
| `offset` | offset added to kernel centres (logtrans only) or `NULL` |
| `xmax` | upper bound on support (copula and beta kernels only) or `NULL` |
| `add.jitter` | logical, whether jitter is needed for rounded kernel centres |
| `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` |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

The extreme value mixture model with boundary corrected kernel density estimate (BCKDE) for 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.

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is (`lambda`, `u`, `sigmau`, `xi`) if threshold is also estimated and (`lambda`, `sigmau`, `xi`) for profile likelihood or fixed threshold approach.

Negative data are ignored.

Cross-validation likelihood is used for BCKDE, but standard likelihood is used for GPD component. See help for `fkden` for details, type `help fkden`.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default used in the likelihood fitting. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

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 implemented. Hence, a bandwidth must always be specified.

The simple, renorm, beta1, beta2 gamma1 and gamma2 boundary corrected kernel density estimates require renormalisation, achieved by numerical integration, so are very time consuming.

## Value

lbckdengpd, nlbckdengpd, and nlubckdengpd give the log-likelihood, negative log-likelihood and profile likelihood for threshold. Profile likelihood for single threshold is given by proflubckdengpd. fbckdengpd returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| lambda: | MLE of lambda (kernel half-width) |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |
| bw: | MLE of bw (kernel standard deviations) |
| kernel: | kernel name |
| 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 scaled beta or copula |

## Boundary Correction Methods

See dbckden for details of BCKDE methods.

## Warning

See important warnings about cross-validation likelihood estimation in fkden, type help fkden.

See important warnings about boundary correction approaches in dbckden, type help bckden.

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

See notes in fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

No default initial values for parameter vector are provided, so will stop evaluation if pvector is left as NULL. Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

The data and kernel centres are both vectors. Infinite, missing and negative sample values (and kernel centres) are dropped.

## Author(s)

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

## References

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

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kdengpd: bckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, gkg, kdengpdcon, kdengpd, kden

Other bckden: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckden, fkden, kden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckden, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckden, fkdengpdcon, gkgcon, kdengpdcon

Other fbckdengpd: bckdengpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rgamma(500, 2, 1)
xx = seq(-0.1, 10, 0.01)
y = dgamma(xx, 2, 1)

# Bulk model based tail fraction
pinit = c(0.1, quantile(x, 0.9), 1, 0.1) # initial values required for BCKDE
fit = fbckdengpd(x, pvector = pinit, bcmethod = "cutnorm")
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10))
lines(xx, y)
with(fit, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, bcmethod = "cutnorm"), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = fbckdengpd(x, phiu = FALSE, pvector = pinit, bcmethod = "cutnorm")
with(fit2, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, phiu, bc = "cutnorm"), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
pinit = c(0.1, 1, 0.1) # notice threshold dropped from initial values
fitu = fbckdengpd(x, useq = seq(1, 6, length = 20), pvector = pinit, bcmethod = "cutnorm")
fitfix = fbckdengpd(x, useq = seq(1, 6, length = 20), fixedu = TRUE, pv = pinit, bc = "cutnorm")

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10))
lines(xx, y)
with(fit, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, bc = "cutnorm"), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, bc = "cutnorm"), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, bc = "cutnorm"), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

fbckdengpdcon *MLE Fitting of Boundary Corrected Kernel Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

---

### Description

Maximum likelihood estimation for fitting the extreme value mixture model with boundary corrected kernel density estimate for bulk distribution upto the threshold and conditional GPD above thresholdwith continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

### Usage

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

lbckdengpdcon(x, lambda = NULL, u = 0, xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", bcmethod = "simple",
  proper = TRUE, nn = "jf96", offset = NULL, xmax = NULL,
  log = TRUE)

nlbckdengpdcon(pvector, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, finitelik = FALSE)

proflubckdengpdcon(u, pvector, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)

nlubckdengpdcon(pvector, u, x, phiu = TRUE, kernel = "gaussian",
  bcmethod = "simple", proper = TRUE, nn = "jf96", offset = NULL,
  xmax = NULL, finitelik = FALSE)
```

### Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| kernel | kernel name (default = "gaussian") |
| bcmethod | boundary correction method |
| proper | logical, whether density is renormalised to integrate to unity (where needed) |
| nn | non-negativity correction method (simple boundary correction only) |
| offset | offset added to kernel centres (logtrans only) or NULL |
| xmax | upper bound on support (copula and beta kernels only) or NULL |

| | |
|---|---|
| add.jitter | logical, whether jitter is needed for rounded kernel centres |
| 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 |
| lambda | bandwidth for kernel (as half-width of kernel) or NULL |
| u | scalar threshold value |
| xi | scalar shape parameter |
| bw | bandwidth for kernel (as standard deviations of kernel) or NULL |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

The extreme value mixture model with boundary corrected kernel density estimate (BCKDE) for bulk and GPD tail with continuity at threshold is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dbckdengpdcon for details, type help bckdengpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (lambda, u, xi) if threshold is also estimated and (lambda, xi) for profile likelihood or fixed threshold approach.

Negative data are ignored.

Cross-validation likelihood is used for BCKDE, but standard likelihood is used for GPD component. See help for fkden for details, type help fkden.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default used in the likelihood fitting. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

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 implemented. Hence, a bandwidth must always be specified.

The simple, renorm, beta1, beta2 gamma1 and gamma2 boundary corrected kernel density estimates require renormalisation, achieved by numerical integration, so are very time consuming.

## Value

lbckdengpdcon, nlbckdengpdcon, and nlubckdengpdcon give the log-likelihood, negative log-likelihood and profile likelihood for threshold. Profile likelihood for single threshold is given by proflubckdengpdcon. fbckdengpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |

| | |
|---|---|
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `lambda`: | MLE of lambda (kernel half-width) |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale(estimated from other parameters) |
| `xi`: | MLE of GPD shape |
| `phiu`: | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu`: | standard error of MLE of tail fraction |
| `bw`: | MLE of bw (kernel standard deviations) |
| `kernel`: | kernel name |
| `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 scaled beta or copula |

## Boundary Correction Methods

See `dbckden` for details of BCKDE methods.

## Warning

See important warnings about cross-validation likelihood estimation in `fkden`, type `help fkden`.

See important warnings about boundary correction approaches in `dbckden`, type `help bckden`.

## Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`. Based on code by Anna MacDonald produced for MATLAB.

## Note

See notes in `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

No default initial values for parameter vector are provided, so will stop evaluation if `pvector` is left as `NULL`. Avoid setting the starting value for the shape parameter to `xi=0` as depending on the optimisation method it may be get stuck.

The data and kernel centres are both vectors. Infinite, missing and negative sample values (and kernel centres) are dropped.

## Author(s)

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

### References


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

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.


### See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kdengpdcon: bckdengpdcon, fgkgcon, fkdengpdcon, fkdengpd, gkgcon, kdengpdcon, kdengpd

Other bckden: bckdengpdcon, bckdengpd, bckden, fbckdengpd, fbckden, fkden, kden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpd, fbckden, fkdengpd, gkg, kdengpd, kden

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpd, fbckden, fkdengpdcon, gkgcon, kdengpdcon

Other fbckdengpdcon: bckdengpdcon

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rgamma(500, 2, 1)
xx = seq(-0.1, 10, 0.01)
y = dgamma(xx, 2, 1)

# Continuity constraint
pinit = c(0.1, quantile(x, 0.9), 0.1) # initial values required for BCKDE
fit = fbckdengpdcon(x, pvector = pinit, bcmethod = "cutnorm")
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10))
lines(xx, y)
with(fit, lines(xx, dbckdengpdcon(xx, x, lambda, u, xi, bcmethod = "cutnorm"), col="red"))
```

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

# No continuity constraint
pinit = c(0.1, quantile(x, 0.9), 1, 0.1) # initial values required for BCKDE
fit2 = fbckdengpd(x, pvector = pinit, bcmethod = "cutnorm")
with(fit2, lines(xx, dbckdengpd(xx, x, lambda, u, sigmau, xi, bc = "cutnorm"), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
pinit = c(0.1, 0.1) # notice threshold dropped from initial values
fitu = fbckdengpdcon(x, useq = seq(1, 6, length = 20), pvector = pinit, bcmethod = "cutnorm")
fitfix = fbckdengpdcon(x, useq = seq(1, 6, length = 20), fixedu = TRUE, pv = pinit, bc = "cutnorm")

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10))
lines(xx, y)
with(fit, lines(xx, dbckdengpdcon(xx, x, lambda, u, xi, bc = "cutnorm"), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dbckdengpdcon(xx, x, lambda, u, xi, bc = "cutnorm"), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dbckdengpdcon(xx, x, lambda, u, xi, bc = "cutnorm"), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## 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. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflubetagpd(u, pvector, x, phiu = TRUE, method = "BFGS",
```

```
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlubetagpd(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| bshape1 | scalar beta shape 1 (positive) |
| bshape2 | scalar beta shape 2 (positive) |
| u | scalar threshold over $(0, 1)$ |
| sigmau | scalar scale parameter (positive) |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## 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.

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The full parameter vector is (bshape1, bshape2, u, sigmau, xi) if threshold is also estimated and (bshape1, bshape2, sigmau, xi) for profile likelihood or fixed threshold approach.

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

## Value

Log-likelihood is given by lbetagpd and it's wrappers for negative log-likelihood from nlbetagpd and nlubetagpd. Profile likelihood for single threshold given by proflubetagpd. Fitting function fbetagpd returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |

| | |
|---|---|
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| bshape1: | MLE of beta shape1 |
| bshape2: | MLE of beta shape2 |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

Thanks to Vathy Kamulete of the Royal Bank of Canada for reporting a bug in the likelihood function. See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

When pvector=NULL then the initial values are:

- method of moments estimator of beta parameters assuming entire population is beta; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

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

## References

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

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

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

dbeta, fgpd and gpd

Other betagpd: betagpdcon, betagpd, fbetagpdcon

Other betagpdcon: betagpdcon, betagpd, fbetagpdcon

Other fbetagpd: betagpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rbeta(1000, shape1 = 2, shape2 = 4)
xx = seq(-0.1, 2, 0.01)
y = dbeta(xx, shape1 = 2, shape2 = 4)

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

# Parameterised tail fraction
fit2 = fbetagpd(x, phiu = FALSE)
with(fit2, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fbetagpd(x, useq = seq(0.3, 0.7, length = 20))
fitfix = fbetagpd(x, useq = seq(0.3, 0.7, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, y)
with(fit, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

fbetagpdcon *MLE Fitting of beta Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

lbetagpdcon(x, bshape1 = 1, bshape2 = 1, u = qbeta(0.9, bshape1,
  bshape2), xi = 0, phiu = TRUE, log = TRUE)

nlbetagpdcon(pvector, x, phiu = TRUE, finitelik = FALSE)

proflubetagpdcon(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlubetagpdcon(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `bshape1` | scalar beta shape 1 (positive) |
| `bshape2` | scalar beta shape 2 (positive) |
| `u` | scalar threshold over $(0, 1)$ |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

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

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dbetagpdcon for details, type help betagpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (bshape1, bshape2, u, xi) if threshold is also estimated and (bshape1, bshape2, xi) for profile likelihood or fixed threshold approach.

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

## Value

Log-likelihood is given by lbetagpdcon and it's wrappers for negative log-likelihood from nlbetagpdcon and nlubetagpdcon. Profile likelihood for single threshold given by proflubetagpdcon. Fitting function fbetagpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| bshape1: | MLE of beta shape1 |
| bshape2: | MLE of beta shape2 |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale (estimated from other parameters) |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

When pvector=NULL then the initial values are:

- method of moments estimator of beta parameters assuming entire population is beta; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

dbeta, fgpd and gpd

Other betagpd: betagpdcon, betagpd, fbetagpd

Other betagpdcon: betagpdcon, betagpd, fbetagpd

Other fbetagpdcon: betagpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rbeta(1000, shape1 = 2, shape2 = 4)
xx = seq(-0.1, 2, 0.01)
y = dbeta(xx, shape1 = 2, shape2 = 4)

# Continuity constraint
fit = fbetagpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, y)
with(fit, lines(xx, dbetagpdcon(xx, bshape1, bshape2, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fbetagpd(x, phiu = FALSE)
with(fit2, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fbetagpdcon(x, useq = seq(0.3, 0.7, length = 20))
fitfix = fbetagpdcon(x, useq = seq(0.3, 0.7, length = 20), fixedu = TRUE)
```

```
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, y)
with(fit, lines(xx, dbetagpdcon(xx, bshape1, bshape2, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dbetagpdcon(xx, bshape1, bshape2, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dbetagpdcon(xx, bshape1, bshape2, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), 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, ...)

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 parameters (`wshape`, `wscale`, `cmu`, `ctau`, `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` |
| `wshape` | Weibull shape (positive) |
| `wscale` | Weibull scale (positive) |
| `cmu` | Cauchy location |
| `ctau` | Cauchy scale |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## 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.

The log-likelihood and negative log-likelihood are also provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector `pvector` must be specified in the negative log-likelihood `nldwm`.

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).

Non-negative 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

`ldwm` gives (log-)likelihood and `nldwm` gives the negative log-likelihood. `fdwm` 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 parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `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 |

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.

### Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

### Note

Unlike most of 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.

Infinite and missing sample values are dropped.

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/Cauchy_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

fgpd and gpd

Other fdwm: dwm

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rweibull(1000, shape = 2)
xx = seq(-0.1, 4, 0.01)
y = dweibull(xx, shape = 2)

fit = fdwm(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, ddwm(xx, wshape, wscale, cmu, ctau, sigmau, 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. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflugammagpd(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlugammagpd(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in `useq`, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in `useq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `gshape` | scalar gamma shape (positive) |
| `gscale` | scalar gamma scale (positive) |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

### 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.

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is (`gshape`, `gscale`, `u`, `sigmau`, `xi`) if threshold is also estimated and (`gshape`, `gscale`, `sigmau`, `xi`) for profile likelihood or fixed threshold approach.

Non-positive data are ignored as likelihood is infinite, except for `gshape=1`.

### Value

Log-likelihood is given by `lgammagpd` and it's wrappers for negative log-likelihood from `nlgammagpd` and `nlugammagpd`. Profile likelihood for single threshold given by `proflugammagpd`. Fitting function `fgammagpd` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `gshape`: | MLE of gamma shape |
| `gscale`: | MLE of gamma scale |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale |
| `xi`: | MLE of GPD shape |
| `phiu`: | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu`: | standard error of MLE of tail fraction |

### Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

### Note

When `pvector=NULL` then the initial values are:

- approximation of MLE of gamma parameters assuming entire population is gamma; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

### Author(s)

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

### References

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

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

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

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

dgamma, fgpd and gpd

Other gammagpd: fgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, gammagpd, mgammagpd

Other gammagpdcon: fgammagpdcon, fmgammagpdcon, gammagpdcon, gammagpd, mgammagpdcon

Other mgammagpd: fmgammagpdcon, fmgammagpd, fmgamma, gammagpd, mgammagpdcon, mgammagpd, mgamma

Other fgammagpd: gammagpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rgamma(1000, shape = 2)
xx = seq(-0.1, 8, 0.01)
y = dgamma(xx, shape = 2)

# Bulk model based tail fraction
fit = fgammagpd(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 8))
lines(xx, y)
with(fit, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = fgammagpd(x, phiu = FALSE)
with(fit2, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgammagpd(x, useq = seq(1, 5, length = 20))
```

```
fitfix = fgammagpd(x, useq = seq(1, 5, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 8))
lines(xx, y)
with(fit, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

fgammagpdcon *MLE Fitting of Gamma Bulk and GPD Tail Extreme Value Mixture Model with Single 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 continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflugammagpdcon(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlugammagpdcon(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0,1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |

| | |
|---|---|
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `gshape` | scalar gamma shape (positive) |
| `gscale` | scalar gamma scale (positive) |
| `u` | scalar threshold value |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

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

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The GPD `sigmau` parameter is now specified as function of other parameters, see help for `dgammagpdcon` for details, type `help gammagpdcon`. Therefore, `sigmau` should not be included in the parameter vector if initial values are provided, making the full parameter vector (`gshape`, `gscale`, `u`, `xi`) if threshold is also estimated and (`gshape`, `gscale`, `xi`) for profile likelihood or fixed threshold approach.

Non-positive data are ignored as likelihood is infinite, except for `gshape=1`.

## Value

Log-likelihood is given by `lgammagpdcon` and it's wrappers for negative log-likelihood from `nlgammagpdcon` and `nlugammagpdcon`. Profile likelihood for single threshold given by `proflugammagpdcon`. Fitting function `fgammagpdcon` returns a simple list with the following elements

| | |
|---|---|
| `call:` | `optim` call |
| `x:` | data vector `x` |
| `init:` | `pvector` |
| `fixedu:` | fixed threshold, logical |
| `useq:` | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq:` | profile negative log-likelihood at each threshold in useq |
| `optim:` | complete `optim` output |
| `mle:` | vector of MLE of parameters |
| `cov:` | variance-covariance matrix of MLE of parameters |
| `se:` | vector of standard errors of MLE of parameters |
| `rate:` | `phiu` to be consistent with `evd` |
| `nllh:` | minimum negative log-likelihood |
| `n:` | total sample size |
| `gshape:` | MLE of gamma shape |
| `gscale:` | MLE of gamma scale |
| `u:` | threshold (fixed or MLE) |
| `sigmau:` | MLE of GPD scale (estimated from other parameters) |

| | |
|---|---|
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- approximation of MLE of gamma parameters assuming entire population is gamma; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

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

dgamma, fgpd and gpd

Other gammagpd: fgammagpd, fmgammagpd, fmgamma, gammagpdcon, gammagpd, mgammagpd

Other gammagpdcon: fgammagpd, fmgammagpdcon, gammagpdcon, gammagpd, mgammagpdcon

Other mgammagpdcon: fmgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, mgammagpdcon, mgammagpd, mgamma

Other fgammagpdcon: gammagpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rgamma(1000, shape = 2)
xx = seq(-0.1, 8, 0.01)
y = dgamma(xx, shape = 2)

# Continuity constraint
fit = fgammagpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 8))
lines(xx, y)
with(fit, lines(xx, dgammagpdcon(xx, gshape, gscale, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fgammagpd(x, phiu = FALSE)
with(fit2, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgammagpdcon(x, useq = seq(1, 5, length = 20))
fitfix = fgammagpdcon(x, useq = seq(1, 5, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 8))
lines(xx, y)
with(fit, lines(xx, dgammagpdcon(xx, gshape, gscale, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dgammagpdcon(xx, gshape, gscale, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dgammagpdcon(xx, gshape, gscale, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

fgkg    *MLE Fitting of Kernel Density Estimate for Bulk and GPD for Both Tails Extreme Value Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with kernel density estimate for bulk distribution between thresholds and conditional GPDs beyond thresholds. With options for profile likelihood estimation for both thresholds and fixed threshold approach.

## Usage

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

lgkg(x, lambda = NULL, ul = 0, sigmaul = 1, xil = 0,
  phiul = TRUE, ur = 0, sigmaur = 1, xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian", log = TRUE)

nlgkg(pvector, x, phiul = TRUE, phiur = TRUE, kernel = "gaussian",
  finitelik = FALSE)

proflugkg(ulr, pvector, x, phiul = TRUE, phiur = TRUE,
  kernel = "gaussian", method = "BFGS", control = list(maxit =
  10000), finitelik = TRUE, ...)

nlugkg(pvector, ul, ur, x, phiul = TRUE, phiur = TRUE,
  kernel = "gaussian", finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiul | probability of being below lower threshold $(0, 1)$ or logical, see Details in help for fgng |
| phiur | probability of being above upper threshold $(0, 1)$ or logical, see Details in help for fgng |
| ulseq | vector of lower thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| urseq | vector of upper thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| kernel | kernel name (default = "gaussian") |
| add.jitter | logical, whether jitter is needed for rounded kernel centres |
| 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 |
| lambda | scalar bandwidth for kernel (as half-width of kernel) |
| ul | scalar lower tail threshold |

| | |
|---|---|
| sigmaul | scalar lower tail GPD scale parameter (positive) |
| xil | scalar lower tail GPD shape parameter |
| ur | scalar upper tail threshold |
| sigmaur | scalar upper tail GPD scale parameter (positive) |
| xir | scalar upper tail GPD shape parameter |
| bw | scalar bandwidth for kernel (as standard deviations of kernel) |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| ulr | vector of length 2 giving lower and upper tail thresholds or NULL for default values |

## Details

The extreme value mixture model with kernel density estimate for 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.

See help for fnormgpd and fgkg for details, type help fnormgpd and help fgkg. Only the different features are outlined below for brevity.

The full parameter vector is (lambda, ul, sigmaul, xil, ur, sigmaur, xir) if thresholds are also estimated and (lambda, sigmaul, xil, sigmaur, xir) for profile likelihood or fixed threshold approach.

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD components. See help for fkden for details, type help fkden.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default used in the likelihood fitting. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

The tail fractions phiul and phiur are treated separately to the other parameters, to allow for all their representations. In the fitting functions fgkg and proflugkg they are logical:

- default values phiul=TRUE and phiur=TRUE - tail fractions specified by KDE distribution and survivior functions respectively and standard error is output as NA.
- phiul=FALSE and phiur=FALSE - treated as extra parameters estimated using the MLE which is the sample proportion beyond the thresholds and standard error is output.

In the likelihood functions lgkg, nlgkg and nlugkg it can be logical or numeric:

- logical - same as for fitting functions with default values phiul=TRUE and phiur=TRUE.
- numeric - any value over range $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively. Also, phiul+phiur<1 as bulk must contribute.

If the profile likelihood approach is used, then a grid search over all combinations of both thresholds is carried out. The combinations which lead to less than 5 in any datapoints beyond the thresholds are not considered.

## Value

Log-likelihood is given by lgkg and it's wrappers for negative log-likelihood from nlgkg and nlugkg. Profile likelihood for both thresholds given by proflugkg. Fitting function fgkg returns a simple list with the following elements

| | |
|---|---|
| `call:` | `optim` call |
| `x:` | data vector `x` |
| `init:` | `pvector` |
| `fixedu:` | fixed thresholds, logical |
| `ulseq:` | lower threshold vector for profile likelihood or scalar for fixed threshold |
| `urseq:` | upper threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq:` | profile negative log-likelihood at each threshold pair in (ulseq, urseq) |
| `optim:` | complete `optim` output |
| `mle:` | vector of MLE of parameters |
| `cov:` | variance-covariance matrix of MLE of parameters |
| `se:` | vector of standard errors of MLE of parameters |
| `rate:` | `phiu` to be consistent with `evd` |
| `nllh:` | minimum negative log-likelihood |
| `n:` | total sample size |
| `lambda:` | MLE of lambda (kernel half-width) |
| `ul:` | lower threshold (fixed or MLE) |
| `sigmaul:` | MLE of lower tail GPD scale |
| `xil:` | MLE of lower tail GPD shape |
| `phiul:` | MLE of lower tail fraction (bulk model or parameterised approach) |
| `se.phiul:` | standard error of MLE of lower tail fraction |
| `ur:` | upper threshold (fixed or MLE) |
| `sigmaur:` | MLE of upper tail GPD scale |
| `xir:` | MLE of upper tail GPD shape |
| `phiur:` | MLE of upper tail fraction (bulk model or parameterised approach) |
| `se.phiur:` | standard error of MLE of upper tail fraction |
| `bw:` | MLE of bw (kernel standard deviations) |
| `kernel:` | kernel name |

## Warning

See important warnings about cross-validation likelihood estimation in `fkden`, type `help fkden`.

## Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`. Based on code by Anna MacDonald produced for MATLAB.

## Note

The data and kernel centres are both vectors. Infinite and missing sample values (and kernel centres) are dropped.

When `pvector=NULL` then the initial values are:

- normal reference rule for bandwidth, using the `bw.nrd0` function, which is consistent with the `density` function. At least two kernel centres must be provided as the variance needs to be estimated.
- lower threshold 10% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- upper threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters beyond thresholds.

### Author(s)

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

### References

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

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

### See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kden: bckden, fbckden, fgkgcon, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other kdengpd: bckdengpd, fbckdengpd, fkdengpdcon, fkdengpd, fkden, gkg, kdengpdcon, kdengpd, kden

Other gkg: fgkgcon, fkdengpd, gkgcon, gkg, kdengpd, kden

Other gkgcon: fgkgcon, fkdengpdcon, gkgcon, gkg, kdengpdcon

Other fgkg: gkg

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model based tail fraction
fit = fgkg(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
```

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

# Parameterised tail fraction
fit2 = fgkg(x, phiul = FALSE, phiur = FALSE)
with(fit2, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="blue"))
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)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgkg(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10))
fitfix = fgkg(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")
with(fitu, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="purple"))
abline(v = c(fitu$ul, fitu$ur), col = "purple")
with(fitfix, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="darkgreen"))
abline(v = c(fitfix$ul, fitfix$ur), col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

`fgkgcon` *MLE Fitting of Kernel Density Estimate for Bulk and GPD for Both Tails with Single Continuity Constraint at Both Thresholds Extreme Value Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with kernel density estimate for bulk distribution between thresholds and conditional GPDs for both tails with continuity at thresholds. With options for profile likelihood estimation for both thresholds and fixed threshold approach.

## Usage

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

```
lgkgcon(x, lambda = NULL, ul = 0, xil = 0, phiul = TRUE, ur = 0,
  xir = 0, phiur = TRUE, bw = NULL, kernel = "gaussian",
  log = TRUE)

nlgkgcon(pvector, x, phiul = TRUE, phiur = TRUE, kernel = "gaussian",
  finitelik = FALSE)

proflugkgcon(ulr, pvector, x, phiul = TRUE, phiur = TRUE,
  kernel = "gaussian", method = "BFGS", control = list(maxit =
  10000), finitelik = TRUE, ...)

nlugkgcon(pvector, ul, ur, x, phiul = TRUE, phiur = TRUE,
  kernel = "gaussian", finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiul` | probability of being below lower threshold $(0, 1)$ or logical, see Details in help for `fgng` |
| `phiur` | probability of being above upper threshold $(0, 1)$ or logical, see Details in help for `fgng` |
| `ulseq` | vector of lower thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `urseq` | vector of upper thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in `ulseq`/`urseq`, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in `ulseq`/`urseq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `kernel` | kernel name (`default = "gaussian"`) |
| `add.jitter` | logical, whether jitter is needed for rounded kernel centres |
| `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` |
| `lambda` | scalar bandwidth for kernel (as half-width of kernel) |
| `ul` | scalar lower tail threshold |
| `xil` | scalar lower tail GPD shape parameter |
| `ur` | scalar upper tail threshold |
| `xir` | scalar upper tail GPD shape parameter |
| `bw` | scalar bandwidth for kernel (as standard deviations of kernel) |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |
| `ulr` | vector of length 2 giving lower and upper tail thresholds or `NULL` for default values |

## Details

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

See help for fnormgpd and fgng for details, type help fnormgpd and help fgng. Only the different features are outlined below for brevity.

The GPD sigmaul and sigmaur parameters are now specified as function of other parameters, see help for dgkgcon for details, type help gkgcon. Therefore, sigmaul and sigmaur should not be included in the parameter vector if initial values are provided, making the full parameter vector The full parameter vector is (lambda, ul, xil, ur, xir) if thresholds are also estimated and (lambda, xil, xir) for profile likelihood or fixed threshold approach.

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD components. See help for fkden for details, type help fkden.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default used in the likelihood fitting. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

The tail fractions phiul and phiur are treated separately to the other parameters, to allow for all their representations. In the fitting functions fgkgcon and proflugkgcon they are logical:

- default values phiul=TRUE and phiur=TRUE - tail fractions specified by KDE distribution and survivior functions respectively and standard error is output as NA.
- phiul=FALSE and phiur=FALSE - treated as extra parameters estimated using the MLE which is the sample proportion beyond the thresholds and standard error is output.

In the likelihood functions lgkgcon, nlgkgcon and nlugkgcon it can be logical or numeric:

- logical - same as for fitting functions with default values phiul=TRUE and phiur=TRUE.
- numeric - any value over range $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively. Also, phiul+phiur<1 as bulk must contribute.

If the profile likelihood approach is used, then a grid search over all combinations of both thresholds is carried out. The combinations which lead to less than 5 in any datapoints beyond the thresholds are not considered.

## Value

Log-likelihood is given by lgkgcon and it's wrappers for negative log-likelihood from nlgkgcon and nlugkgcon. Profile likelihood for both thresholds given by proflugkgcon. Fitting function fgkgcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed thresholds, logical |
| ulseq: | lower threshold vector for profile likelihood or scalar for fixed threshold |
| urseq: | upper threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold pair in (ulseq, urseq) |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |

| | |
|---|---|
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| lambda: | MLE of lambda (kernel half-width) |
| ul: | lower threshold (fixed or MLE) |
| sigmaul: | MLE of lower tail GPD scale (estimated from other parameters) |
| xil: | MLE of lower tail GPD shape |
| phiul: | MLE of lower tail fraction (bulk model or parameterised approach) |
| se.phiul: | standard error of MLE of lower tail fraction |
| ur: | upper threshold (fixed or MLE) |
| sigmaur: | MLE of upper tail GPD scale (estimated from other parameters) |
| xir: | MLE of upper tail GPD shape |
| phiur: | MLE of upper tail fraction (bulk model or parameterised approach) |
| se.phiur: | standard error of MLE of lower tail fraction |
| bw: | MLE of bw (kernel standard deviations) |
| kernel: | kernel name |

## Warning

See important warnings about cross-validation likelihood estimation in fkden, type help fkden.

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

The data and kernel centres are both vectors. Infinite and missing sample values (and kernel centres) are dropped.

When pvector=NULL then the initial values are:

- normal reference rule for bandwidth, using the bw.nrd0 function, which is consistent with the density function. At least two kernel centres must be provided as the variance needs to be estimated.
- lower threshold 10% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- upper threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameters beyond thresholds.

## Author(s)

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

## References

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kden: bckden, fbckden, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fkdengpdcon, fkdengpd, gkgcon, kdengpdcon, kdengpd

Other gkg: fgkg, fkdengpd, gkgcon, gkg, kdengpd, kden

Other gkgcon: fgkg, fkdengpdcon, gkgcon, gkg, kdengpdcon

Other fgkgcon: gkgcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Continuity constraint
fit = fgkgcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgkgcon(xx, x, lambda, ul, xil, phiul,
   ur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")

# No continuity constraint
fit2 = fgkg(x)
with(fit2, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="blue"))
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topleft", c("True Density","No continuity constraint","With continuty constraint"),
```

```
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgkgcon(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10))
fitfix = fgkgcon(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgkgcon(xx, x, lambda, ul, xil, phiul,
   ur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")
with(fitu, lines(xx, dgkgcon(xx, x, lambda, ul, xil, phiul,
   ur, xir, phiur), col="purple"))
abline(v = c(fitu$ul, fitu$ur), col = "purple")
with(fitfix, lines(xx, dgkgcon(xx, x, lambda, ul, xil, phiul,
   ur, xir, phiur), col="darkgreen"))
abline(v = c(fitfix$ul, fitfix$ur), col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fgng — *MLE Fitting of Normal Bulk and GPD for Both Tails Extreme Value Mixture Model*

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution between thresholds and conditional GPDs beyond thresholds. With options for profile likelihood estimation for both thresholds and fixed threshold approach.

## Usage

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

lgng(x, nmean = 0, nsd = 1, ul = 0, sigmaul = 1, xil = 0,
  phiul = TRUE, ur = 0, sigmaur = 1, xir = 0, phiur = TRUE,
  log = TRUE)

nlgng(pvector, x, phiul = TRUE, phiur = TRUE, finitelik = FALSE)

proflugng(ulr, pvector, x, phiul = TRUE, phiur = TRUE,
  method = "BFGS", control = list(maxit = 10000), finitelik = TRUE,
  ...)

nlugng(pvector, ul, ur, x, phiul = TRUE, phiur = TRUE,
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiul | probability of being below lower threshold $(0, 1)$ or logical, see Details in help for fgng |
| phiur | probability of being above upper threshold $(0, 1)$ or logical, see Details in help for fgng |
| ulseq | vector of lower thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| urseq | vector of upper thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| ul | scalar lower tail threshold |
| sigmaul | scalar lower tail GPD scale parameter (positive) |
| xil | scalar lower tail GPD shape parameter |
| ur | scalar upper tail threshold |
| sigmaur | scalar upper tail GPD scale parameter (positive) |
| xir | scalar upper tail GPD shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| ulr | vector of length 2 giving lower and upper tail thresholds or NULL for default values |

## 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.

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The full parameter vector is (nmean, nsd, ul, sigmaul, xil, ur, sigmaur, xir) if thresholds are also estimated and (nmean, nsd, sigmaul, xil, sigmaur, xir) for profile likelihood or fixed threshold approach.

The tail fractions phiul and phiur are treated separately to the other parameters, to allow for all their representations. In the fitting functions fgng and proflugng they are logical:

- default values phiul=TRUE and phiur=TRUE - tail fractions specified by normal distribution pnorm(ul,nmean,nsd) and survivior functions 1-pnorm(ur,nmean,nsd) respectively and standard error is output as NA.
- phiul=FALSE and phiur=FALSE - treated as extra parameters estimated using the MLE which is the sample proportion beyond the thresholds and standard error is output.

In the likelihood functions lgng, nlgng and nlugng it can be logical or numeric:

- logical - same as for fitting functions with default values phiul=TRUE and phiur=TRUE.
- numeric - any value over range $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively. Also, phiul+phiur<1 as bulk must contribute.

If the profile likelihood approach is used, then a grid search over all combinations of both thresholds is carried out. The combinations which lead to less than 5 in any datapoints beyond the thresholds are not considered.

## Value

Log-likelihood is given by lgng and it's wrappers for negative log-likelihood from nlgng and nlugng. Profile likelihood for both thresholds given by proflugng. Fitting function fgng returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed thresholds, logical |
| ulseq: | lower threshold vector for profile likelihood or scalar for fixed threshold |
| urseq: | upper threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold pair in (ulseq, urseq) |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| ul: | lower threshold (fixed or MLE) |
| sigmaul: | MLE of lower tail GPD scale |
| xil: | MLE of lower tail GPD shape |
| phiul: | MLE of lower tail fraction (bulk model or parameterised approach) |
| se.phiul: | standard error of MLE of lower tail fraction |
| ur: | upper threshold (fixed or MLE) |
| sigmaur: | MLE of upper tail GPD scale |
| xir: | MLE of upper tail GPD shape |
| phiur: | MLE of upper tail fraction (bulk model or parameterised approach) |
| se.phiur: | standard error of MLE of upper tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Xin Zhao produced for MATLAB.

## Note

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- lower threshold 10% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- upper threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters beyond threshold.

## Author(s)

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

## References

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

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

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

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.

Mendes, B. and H. F. Lopes (2004). Data driven estimates for mixtures. Computational Statistics and Data Analysis 47(3), 583-598.

## See Also

dnorm, fgpd and gpd

Other normgpd: fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other gng: fgngcon, fitmgng, fnormgpd, gngcon, gng, itmgng, normgpd

Other gngcon: fgngcon, fnormgpdcon, gngcon, gng, normgpdcon

Other fgng: gng

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model based tail fraction
fit = fgng(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")

# Parameterised tail fraction
fit2 = fgng(x, phiul = FALSE, phiur = FALSE)
with(fit2, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="blue"))
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)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgng(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10))
fitfix = fgng(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")
with(fitu, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="purple"))
abline(v = c(fitu$ul, fitu$ur), col = "purple")
with(fitfix, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="darkgreen"))
abline(v = c(fitfix$ul, fitfix$ur), col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

## fgngcon *MLE Fitting of Normal Bulk and GPD for Both Tails with Single Continuity Constraint at Both Thresholds Extreme Value Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution between thresholds and conditional GPDs for both tails with continuity at thresholds. With options for profile likelihood estimation for both thresholds and fixed threshold approach.

## Usage

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

lgngcon(x, nmean = 0, nsd = 1, ul = 0, xil = 0, phiul = TRUE,
  ur = 0, xir = 0, phiur = TRUE, log = TRUE)

nlgngcon(pvector, x, phiul = TRUE, phiur = TRUE, finitelik = FALSE)

proflugngcon(ulr, pvector, x, phiul = TRUE, phiur = TRUE,
  method = "BFGS", control = list(maxit = 10000), finitelik = TRUE,
  ...)

nlugngcon(pvector, ul, ur, x, phiul = TRUE, phiur = TRUE,
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiul | probability of being below lower threshold $(0, 1)$ or logical, see Details in help for fgng |
| phiur | probability of being above upper threshold $(0, 1)$ or logical, see Details in help for fgng |
| ulseq | vector of lower thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| urseq | vector of upper thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| ul | scalar lower tail threshold |
| xil | scalar lower tail GPD shape parameter |
| ur | scalar upper tail threshold |

| | |
|---|---|
| xir | scalar upper tail GPD shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| ulr | vector of length 2 giving lower and upper tail thresholds or NULL for default values |

## Details

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

See help for fnormgpd and fgngfor details, type help fnormgpd and help fgng. Only the different features are outlined below for brevity.

The GPD sigmaul and sigmaur parameters are now specified as function of other parameters, see help for dgngcon for details, type help gngcon. Therefore, sigmaul and sigmaur should not be included in the parameter vector if initial values are provided, making the full parameter vector The full parameter vector is (nmean, nsd, ul, xil, ur, xir) if thresholds are also estimated and (nmean, nsd, xil, xir) for profile likelihood or fixed threshold approach.

If the profile likelihood approach is used, then a grid search over all combinations of both thresholds is carried out. The combinations which lead to less than 5 in any datapoints beyond the thresholds are not considered.

## Value

Log-likelihood is given by lgngcon and it's wrappers for negative log-likelihood from nlgngcon and nlugngcon. Profile likelihood for both thresholds given by proflugngcon. Fitting function fgngcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed thresholds, logical |
| ulseq: | lower threshold vector for profile likelihood or scalar for fixed threshold |
| urseq: | upper threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold pair in (ulseq, urseq) |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| ul: | lower threshold (fixed or MLE) |
| sigmaul: | MLE of lower tail GPD scale (estimated from other parameters) |
| xil: | MLE of lower tail GPD shape |
| phiul: | MLE of lower tail fraction (bulk model or parameterised approach) |
| se.phiul: | standard error of MLE of lower tail fraction |
| ur: | upper threshold (fixed or MLE) |
| sigmaur: | MLE of upper tail GPD scale (estimated from other parameters) |
| xir: | MLE of upper tail GPD shape |
| phiur: | MLE of upper tail fraction (bulk model or parameterised approach) |

se.phiur: standard error of MLE of upper tail fraction

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Xin Zhao produced for MATLAB.

## Note

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- lower threshold 10% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- upper threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameters beyond threshold.

## Author(s)

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

## References

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

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

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

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.

Mendes, B. and H. F. Lopes (2004). Data driven estimates for mixtures. Computational Statistics and Data Analysis 47(3), 583-598.

## See Also

dnorm, fgpd and gpd

Other normgpdcon: fhpdcon, flognormgpdcon, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, normgpdcon, normgpd

Other gng: fgng, fitmgng, fnormgpd, gngcon, gng, itmgng, normgpd

Other gngcon: fgng, fnormgpdcon, gngcon, gng, normgpdcon

Other fgngcon: gngcon

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Continuity constraint
fit = fgngcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgngcon(xx, nmean, nsd, ul, xil, phiul,
   ur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")

# No continuity constraint
fit2 = fgng(x)
with(fit2, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
   ur, sigmaur, xir, phiur), col="blue"))
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topleft", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fgngcon(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10))
fitfix = fgngcon(x, ulseq = seq(-2, -0.2, length = 10),
 urseq = seq(0.2, 2, length = 10), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dgngcon(xx, nmean, nsd, ul, xil, phiul,
   ur, xir, phiur), col="red"))
abline(v = c(fit$ul, fit$ur), col = "red")
with(fitu, lines(xx, dgngcon(xx, nmean, nsd, ul, xil, phiul,
   ur, xir, phiur), col="purple"))
abline(v = c(fitu$ul, fitu$ur), col = "purple")
with(fitfix, lines(xx, dgngcon(xx, nmean, nsd, ul, xil, phiul,
   ur, xir, phiur), col="darkgreen"))
abline(v = c(fitfix$ul, fitfix$ur), col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), 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", control = list(maxit = 10000), finitelik = TRUE,
  ...)

lgpd(x, u = 0, sigmau = 1, xi = 0, phiu = 1, log = TRUE)

nlgpd(pvector, x, u = 0, phiu = 1, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| u | scalar threshold |
| phiu | probability of being above threshold $[0, 1]$ or NULL, see Details |
| pvector | vector of initial values of GPD parameters (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 |
| sigmau | scalar scale parameter (positive) |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## 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 log-likelihood and negative log-likelihood are also provided for wider usage, e.g. constructing your own extreme value mixture model or profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood nlgpd.

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).

The default value for the tail fraction phiu in the fitting function fgpd is NULL, in which case the MLE is calculated using the sample proportion of exceedances. In this case the standard error for phiu is estimated and output as se.phiu, otherwise it is set to NA. Consistent with the evd library the missing values (NA and NaN) are assumed to be below the threshold in calculating the tail fraction.

Otherwise, in the fitting function fgpd the tail fraction phiu can be specified as any value over $(0, 1]$, i.e. excludes $\phi_u = 0$, leading to the unconditional log-likelihood being used for estimation. In this case the standard error will be output as NA.

In the log-likelihood functions lgpd and nlgpd the tail fraction phiu cannot be NULL but can be over the range $[0, 1]$, i.e. which includes $\phi_u = 0$.

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 $\sigma_u$ and shape $\xi$, 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

lgpd gives (log-)likelihood and nlgpd gives the negative log-likelihood. fgpd 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 parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| u: | threshold |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction |
| se.phiu: | standard error of MLE of tail fraction (parameterised approach using sample proportion) |

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from fpot and increase usability.

## Acknowledgments

Based on the gpd.fit and fpot functions in the ismev and evd packages for which their author's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

## Note

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

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 in the fitting fpgd, in which case the MLE assumes that excesses over the threshold are given, rather than exceedances.

The usual default of phiu=1 is given in the likelihood functions lpgd and nlpgd.

The lgpd also has the usual defaults for the other parameters, but nlgpd has no defaults.

Infinite sample values are dropped in fitting function fpgd, but missing values are used to estimate phiu as described above. But in likelihood functions lpgd and nlpgd both infinite and missing values are ignored.

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/Generalized_Pareto_distribution

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

## See Also

dgpd, fpot and fitdistr

Other gpd: gpd

Other fgpd: gpd

## Examples

```
set.seed(1)
par(mfrow = c(2, 1))

# GPD is conditional model for threshold exceedances
# so tail fraction phiu not relevant when only have exceedances
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)
lines(xx, dgpd(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi), col="red")

# but tail fraction phiu is needed for conditional modelling of population tail
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)
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, ...)

lhpd(x, nmean = 0, nsd = 1, xi = 0, log = TRUE)

nlhpd(pvector, x, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| pvector | vector of initial values of parameters (nmean, nsd, 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 |
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## 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.

The log-likelihood and negative log-likelihood are also provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood nlhpd.

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).

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 function lhpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

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

lhpd gives (log-)likelihood and nlhpd gives the negative log-likelihood. fhpd 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 parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| u: | threshold (implicit from other parameters) |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (implied by 1/(1+pnorm(u,nmean,nsd))) |

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 most of 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.

A default value for the tail fraction phiu=TRUE is given. The lhpd also has the usual defaults for the other parameters, but nlhpd has no defaults.

Invalid parameter ranges will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

Infinite and missing sample values are dropped.

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

fgpd 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: fhpdcon, hpdcon, hpd

Other hpdcon: fhpdcon, hpdcon, hpd

Other normgpd: fgng, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other fhpd: hpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Hybrid Pareto provides reasonable fit for some asymmetric heavy upper tailed distributions
# 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)
with(fit, lines(xx, dhpd(xx, nmean, nsd, 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)
```

```
with(fit2, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="blue"))
abline(v = fit2$u)
legend("topright", c("Standard Normal", "Hybrid Pareto", "Normal+GPD Continuous"),
  col=c("black", "red", "blue"), lty = 1)

## 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 model, with only continuity at threshold and not necessarily continuous in first derivative. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

lhpdcon(x, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd), xi = 0,
  log = TRUE)

nlhpdcon(pvector, x, finitelik = FALSE)

profluhpdcon(u, pvector, x, method = "BFGS", control = list(maxit =
  10000), finitelik = TRUE, ...)

nluhpdcon(pvector, u, x, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |

| | |
|---|---|
| u | scalar threshold value |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

The hybrid Pareto model is fitted to the entire dataset using maximum likelihood estimation, with only continuity at threshold and not necessarily continuous in first derivative. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

Note that the key difference between this model (hpdcon) and the normal with GPD tail and continuity at threshold (normgpdcon) is that the latter includes the rescaling of the conditional GPD component by the tail fraction to make it an unconditional tail model. However, for the hybrid Pareto with single continuity constraint use the GPD in it's conditional form with no differential scaling compared to the bulk model.

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The profile likelihood and fixed threshold approach functionality are implemented for this version of the hybrid Pareto as it includes the threshold as a parameter. Whereas the usual hybrid Pareto does not naturally have a threshold parameter.

The GPD sigmau parameter is now specified as function of other parameters, see help for dhpdcon for details, type help hpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (nmean, nsd, u, xi) if threshold is also estimated and (nmean, nsd, xi) for profile likelihood or fixed threshold approach.

## Value

lhpdcon, nlhpdcon, and nluhpdcon give the log-likelihood, negative log-likelihood and profile likelihood for threshold. Profile likelihood for single threshold is given by profluhpdcon. fhpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale (estimated from other parameters) |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (implied by 1/(1+pnorm(u,nmean,nsd))) |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of normal parameters assuming entire population is normal; and
- MLE of GPD parameters above threshold.

Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

## Author(s)

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

## References

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

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

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

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

dnorm, fgpd 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: fhpd, hpdcon, hpd

Other hpdcon: fhpd, hpdcon, hpd

Other normgpdcon: fgngcon, flognormgpdcon, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, normgpdcon, normgpd

Other fhpdcon: hpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Hybrid Pareto provides reasonable fit for some asymmetric heavy upper tailed distributions
# but not for cases such as the normal distribution

# Continuity constraint
fit = fhpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fhpd(x)
with(fit2, lines(xx, dhpd(xx, nmean, nsd, xi), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fhpdcon(x, useq = seq(-2, 2, length = 20))
fitfix = fhpdcon(x, useq = seq(-2, 2, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topleft", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

# Notice that if tail fraction is included a better fit is obtained
fittailfrac = fnormgpdcon(x)

par(mfrow = c(1, 1))
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fittailfrac, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="blue"))
abline(v = fittailfrac$u)
legend("topright", c("Standard Normal", "Hybrid Pareto Continuous", "Normal+GPD Continuous"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

`fitmgng` *MLE Fitting of Normal Bulk and GPD for Both Tails Interval Transition Mixture Model*

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with normal for bulk distribution between thresholds, conditional GPDs beyond thresholds and interval transition. With options for profile likelihood estimation for both thresholds and interval half-width, which can also be fixed.

## Usage

```
fitmgng(x, eseq = NULL, ulseq = NULL, urseq = NULL,
  fixedeu = FALSE, pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

litmgng(x, nmean = 0, nsd = 1, epsilon = nsd, ul = 0,
  sigmaul = 1, xil = 0, ur = 0, sigmaur = 1, xir = 0,
  log = TRUE)

nlitmgng(pvector, x, finitelik = FALSE)

profleuitmgng(eulr, pvector, x, method = "BFGS", control = list(maxit =
  10000), finitelik = TRUE, ...)

nleuitmgng(pvector, epsilon, ul, ur, x, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `eseq` | vector of epsilons (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `ulseq` | vector of lower thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `urseq` | vector of upper thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedeu` | logical, should threshold and epsilon be fixed (at either scalar value in useq and eseq, or estimated from maximum of profile likelihood evaluated at grid of thresholds and epsilons in useq and eseq) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |

| | |
|---|---|
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| epsilon | interval half-width |
| ul | lower tail threshold |
| sigmaul | lower tail GPD scale parameter (positive) |
| xil | lower tail GPD shape parameter |
| ur | upper tail threshold |
| sigmaur | upper tail GPD scale parameter (positive) |
| xir | upper tail GPD shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| eulr | vector of epsilon, lower and upper thresholds considered in profile likelihood |

## Details

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

See ditmgng for explanation of GPD-normal-GPD interval transition model, including mixing functions.

See also help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The full parameter vector is (nmean, nsd, epsilon, ul, sigmaul, xil, ur, sigmaur, xir) if thresholds and interval half-width are also estimated and (nmean, nsd, sigmaul, xil, sigmaur, xir) for profile likelihood or fixed threshold approach.

If the profile likelihood approach is used, then a grid search over all combinations of epsilons and both thresholds are carried out. The combinations which lead to less than 5 in any component outside of the intervals are not considered.

A fixed pair of thresholds and epsilon approach is acheived by setting a single scalar value to each in ulseq, urseq and eseq respectively.

## Value

Log-likelihood is given by litmgng and it's wrappers for negative log-likelihood from nlitmgng and nluitmgng. Profile likelihood for thresholds and interval half-width given by profluitmgng. Fitting function fitmgng returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedeu: | fixed epsilon and threshold, logical |
| ulseq: | lower threshold vector for profile likelihood or scalar for fixed threshold |
| urseq: | upper threshold vector for profile likelihood or scalar for fixed threshold |
| eseq: | interval half-width vector for profile likelihood or scalar for fixed threshold |
| nllheuseq: | profile negative log-likelihood at each combination in (eseq, ulseq, urseq) |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |

| | |
|---|---|
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| epsilon: | MLE of transition half-width |
| ul: | lower threshold (fixed or MLE) |
| sigmaul: | MLE of lower tail GPD scale |
| xil: | MLE of lower tail GPD shape |
| ur: | upper threshold (fixed or MLE) |
| sigmaur: | MLE of upper tail GPD scale |
| xir: | MLE of upper tail GPD shape |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Xin Zhao produced for MATLAB.

## Note

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- lower threshold 10% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- upper threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters beyond threshold.

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

## See Also

fgng, dnorm, fgpd and gpd

Other itmgng: itmgng

Other itmnormgpd: fitmnormgpd, itmgng, itmnormgpd

Other gng: fgngcon, fgng, fnormgpd, gngcon, gng, itmgng, normgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# MLE for complete parameter set (not recommended!)
fit = fitmgng(x)
hist(x, breaks = seq(-6, 6, 0.1), freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, ditmgng(xx, nmean, nsd, epsilon, ul, sigmaul, xil,
                                                     ur, sigmaur, xir), col="red"))
abline(v = fit$ul + fit$epsilon * seq(-1, 1), col = "red")
abline(v = fit$ur + fit$epsilon * seq(-1, 1), col = "darkred")

# Profile likelihood for threshold which is then fixed
fitfix = fitmgng(x, eseq = seq(0, 2, 0.1), ulseq = seq(-2.5, 0, 0.25),
                                         urseq = seq(0, 2.5, 0.25), fixedeu = TRUE)
with(fitfix, lines(xx, ditmgng(xx, nmean, nsd, epsilon, ul, sigmaul, xil,
                                                     ur, sigmaur, xir), col="blue"))
abline(v = fitfix$ul + fitfix$epsilon * seq(-1, 1), col = "blue")
abline(v = fitfix$ur + fitfix$epsilon * seq(-1, 1), col = "darkblue")
legend("topright", c("True Density", "GPD-normal-GPD ITM", "Profile likelihood"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

`fitmnormgpd` *MLE Fitting of Normal Bulk and GPD Tail Interval Transition Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with the normal bulk and GPD tail interval transition mixture model. With options for profile likelihood estimation for threshold and interval half-width, which can both be fixed.

## Usage

```
fitmnormgpd(x, eseq = NULL, useq = NULL, fixedeu = FALSE,
  pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

litmnormgpd(x, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9,
  nmean, nsd), sigmau = nsd, xi = 0, log = TRUE)

nlitmnormgpd(pvector, x, finitelik = FALSE)

profleuitmnormgpd(eu, pvector, x, method = "BFGS", control = list(maxit
```

```
  = 10000), finitelik = TRUE, ...)

nleuitmnormgpd(pvector, epsilon, u, x, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `eseq` | vector of epsilons (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedeu` | logical, should threshold and epsilon be fixed (at either scalar value in `useq` and `eseq`, or estimated from maximum of profile likelihood evaluated at grid of thresholds and epsilons in `useq` and `eseq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `nmean` | scalar normal mean |
| `nsd` | scalar normal standard deviation (positive) |
| `epsilon` | interval half-width |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |
| `eu` | vector of epsilon and threshold pair considered in profile likelihood |

## Details

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

See `ditmnormgpd` for explanation of normal-GPD interval transition model, including mixing functions.

See also help for `fnormgpd` for mixture model fitting details. Only the different features are outlined below for brevity.

The full parameter vector is (`nmean`, `nsd`, `epsilon`, `u`, `sigmau`, `xi`) if threshold and interval half-width are both estimated and (`nmean`, `nsd`, `sigmau`, `xi`) for profile likelihood or fixed threshold and epsilon approach.

If the profile likelihood approach is used, then it is applied to both the threshold and epsilon parameters together. A grid search over all combinations of epsilons and thresholds are considered. The combinations which lead to less than 5 on either side of the interval are not considered.

A fixed threshold and epsilon approach is acheived by setting a single scalar value to each in `useq` and `eseq` respectively.

If the profile likelihood approach is used, then a grid search over all combinations of epsilon and threshold are carried out. The combinations which lead to less than 5 in any any interval are not considered.

## Value

Log-likelihood is given by litmnormgpd and it's wrappers for negative log-likelihood from nlitmnormgpd and nluitmnormgpd. Profile likelihood for threshold and interval half-width given by profluitmnormgpd. Fitting function fitmnormgpd returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedeu: | fixed epsilon and threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| eseq: | epsilon vector for profile likelihood or scalar for fixed epsilon |
| nllheuseq: | profile negative log-likelihood at each combination in (eseq, useq) |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal shape |
| nsd: | MLE of normal scale |
| epsilon: | MLE of transition half-width |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- epsilon is MLE of normal standard deviation;
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

## See Also

fnormgpd, dnorm, fgpd and gpd

Other normgpd: fgng, fhpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other itmnormgpd: fitmgng, itmgng, itmnormgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# MLE for complete parameter set
fit = fitmnormgpd(x)
hist(x, breaks = seq(-6, 6, 0.1), freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, ditmnormgpd(xx, nmean, nsd, epsilon, u, sigmau, xi), col="red"))
abline(v = fit$u + fit$epsilon * seq(-1, 1), col = "red")

# Profile likelihood for threshold which is then fixed
fitfix = fitmnormgpd(x, eseq = seq(0, 2, 0.1), useq = seq(0, 2.5, 0.1), fixedeu = TRUE)
with(fitfix, lines(xx, ditmnormgpd(xx, nmean, nsd, epsilon, u, sigmau, xi), col="blue"))
abline(v = fitfix$u + fitfix$epsilon * seq(-1, 1), col = "blue")
legend("topright", c("True Density", "normal-GPD ITM", "Profile likelihood"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

fitmweibullgpd *MLE Fitting of Weibull Bulk and GPD Tail Interval Transition Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme valeu mixture model with the Weibull bulk and GPD tail interval transition mixture model. With options for profile likelihood estimation for threshold and interval half-width, which can both be fixed.

## Usage

```
fitmweibullgpd(x, eseq = NULL, useq = NULL, fixedeu = FALSE,
  pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

litmweibullgpd(x, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  u = qweibull(0.9, wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 +
```

```
  2/wshape) - (wscale * gamma(1 + 1/wshape))^2), xi = 0, log = TRUE)

nlitmweibullgpd(pvector, x, finitelik = FALSE)

profleuitmweibullgpd(eu, pvector, x, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nleuitmweibullgpd(pvector, epsilon, u, x, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `eseq` | vector of epsilons (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedeu` | logical, should threshold and epsilon be fixed (at either scalar value in `useq` and `eseq`, or estimated from maximum of profile likelihood evaluated at grid of thresholds and epsilons in `useq` and `eseq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `wshape` | scalar Weibull shape (positive) |
| `wscale` | scalar Weibull scale (positive) |
| `epsilon` | interval half-width |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |
| `eu` | vector of epsilon and threshold pair considered in profile likelihood |

## Details

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

See `ditmweibullgpd` for explanation of Weibull-GPD interval transition model, including mixing functions.

See also help for `fnormgpd` for mixture model fitting details. Only the different features are outlined below for brevity.

The full parameter vector is (`wshape`, `wscale`, `epsilon`, `u`, `sigmau`, `xi`) if threshold and interval half-width are both estimated and (`wshape`, `wscale`, `sigmau`, `xi`) for profile likelihood or fixed threshold and epsilon approach.

If the profile likelihood approach is used, then it is applied to both the threshold and epsilon parameters together. A grid search over all combinations of epsilons and thresholds are considered. The combinations which lead to less than 5 on either side of the interval are not considered.

A fixed threshold and epsilon approach is acheived by setting a single scalar value to each in `useq` and `eseq` respectively.

If the profile likelihood approach is used, then a grid search over all combinations of epsilon and threshold are carried out. The combinations which lead to less than 5 in any any interval are not considered.

Negative data are ignored.

## Value

Log-likelihood is given by `litmweibullgpd` and it's wrappers for negative log-likelihood from `nlitmweibullgpd` and `nluitmweibullgpd`. Profile likelihood for threshold and interval half-width given by `profluitmweibullgpd`. Fitting function `fitmweibullgpd` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedeu`: | fixed epsilon and threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `eseq`: | epsilon vector for profile likelihood or scalar for fixed epsilon |
| `nllheuseq`: | profile negative log-likelihood at each combination in (eseq, useq) |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `wshape`: | MLE of Weibull shape |
| `wscale`: | MLE of Weibull scale |
| `epsilon`: | MLE of transition half-width |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale |
| `xi`: | MLE of GPD shape |

## Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

## Note

When `pvector=NULL` then the initial values are:

- MLE of Weibull parameters assuming entire population is Weibull; and
- epsilon is MLE of Weibull standard deviation;
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

## See Also

dweibull, fgpd and gpd

Other weibullgpd: fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpdcon, weibullgpd

Other itmweibullgpd: fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpdcon, weibullgpd

Other fitmweibullgpd: itmweibullgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rweibull(1000, shape = 1, scale = 2)
xx = seq(-0.2, 10, 0.01)
y = dweibull(xx, shape = 1, scale = 2)

# MLE for complete parameter set
fit = fitmweibullgpd(x)
hist(x, breaks = seq(0, 20, 0.1), freq = FALSE, xlim = c(-0.2, 10))
lines(xx, y)
with(fit, lines(xx, ditmweibullgpd(xx, wshape, wscale, epsilon, u, sigmau, xi), col="red"))
abline(v = fit$u + fit$epsilon * seq(-1, 1), col = "red")

# Profile likelihood for threshold which is then fixed
fitfix = fitmweibullgpd(x, eseq = seq(0, 2, 0.1), useq = seq(0.5, 4, 0.1), fixedeu = TRUE)
with(fitfix, lines(xx, ditmweibullgpd(xx, wshape, wscale, epsilon, u, sigmau, xi), col="blue"))
abline(v = fitfix$u + fitfix$epsilon * seq(-1, 1), col = "blue")
legend("topright", c("True Density", "Weibull-GPD ITM", "Profile likelihood"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

---

fkden *Cross-validation MLE Fitting of Kernel Density Estimator, With Variety of Kernels*

---

## Description

Maximum (cross-validation) likelihood estimation for fitting kernel density estimator for a variety of possible kernels, by treating it as a mixture model.

## Usage

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

lkden(x, lambda = NULL, bw = NULL, kernel = "gaussian",
  extracentres = NULL, log = TRUE)

nlkden(lambda, x, bw = NULL, kernel = "gaussian",
  extracentres = NULL, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `linit` | initial value for bandwidth (as kernel half-width) or `NULL` |
| `bwinit` | initial value for bandwidth (as kernel standard deviations) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `extracentres` | extra kernel centres used in KDE, but likelihood contribution not evaluated, or `NULL` |
| `add.jitter` | logical, whether jitter is needed for rounded kernel centres |
| `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` |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

The kernel density estimator (KDE) with one of possible kernels is fitted to the entire dataset using maximum (cross-validation) likelihood estimation. The estimated bandwidth, variance and standard error are automatically output.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` used here but `bw` also output. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` help documentation with the `"gaussian"` as the default choice.

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

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 $i$th 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. Suppose the first `nb` data are below the threshold, followed by `nu` exceedances of the threshold, so $i = 1, \ldots, nb, nb+1, \ldots, nb+nu$. The cross-validation likelihood using the extra kernel centres is then:

$$L(\lambda) \prod_{i=1}^{nb} \hat{f}_{-i}(x_i)$$

where

$$\hat{f}_{-i}(x_i) = \frac{1}{(nb+nu-1)\lambda} \sum_{j=1:j\neq i}^{nb+nu} K(\frac{x_i - x_j}{\lambda})$$

which shows that the complete set of data is used in evaluating the KDE, but only those below the threshold contribute to the cross-validation likelihood. The default is to use the existing data, so `extracentres=NULL`.

The following functions are provided:

- `fkden` - maximum (cross-validation) likelihood fitting with all the above options;
- `lkden` - cross-validation log-likelihood;
- `nlkden` - negative cross-validation log-likelihood;

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions.

The log-likelihood and negative log-likelihood are also provided for wider usage, e.g. constructing your own extreme value mixture models or profile likelihood functions. The parameter `lambda` must be specified in the negative log-likelihood `nlkden`.

Log-likelihood calculations are carried out in `lkden`, which takes bandwidths as inputs 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. `lambda` given as first input).

Defaults values for the bandwidth `linit` and `lambda` are given in the fitting `fkden` and cross-validation likelihood functions `lkden`. The bandwidth `linit` must be specified in the negative log-likelihood function `nlkden`.

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 function `lkden` carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (`log=FALSE`).

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 or for common indicators of lack of convergence (e.g. estimated bandwidth equal to initial value).

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

Log-likelihood is given by `lkden` and it's wrappers for negative log-likelihood from `nlkden`. Fitting function `fkden` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | (jittered) data vector `x` |
| `kerncentres`: | actual kernel centres used `x` |
| `init`: | `linit` for lambda |
| `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 lambda (kernel half-width) |
| `bw`: | MLE of bw (kernel standard deviations) |
| `kernel`: | kernel name |

## 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 $\hat{\lambda} \rightarrow 0$ as $n \rightarrow \infty$, 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 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 trim the data at a suitable threshold to remove the problematic tail from the inference for the bandwidth, using either the `fkdengpd` function for a single heavy tail or the `fgkg` function if both tails are heavy. See MacDonald et al (2013).

### Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

### Note

When linit=NULL then the initial value for the lambda bandwidth is calculated using bw.nrd0 function and transformed using klambda function.

The extra kernel centres extracentres can either be a vector of data or NULL.

Invalid parameter ranges will give 0 for likelihood, log(0)=-Inf for log-likelihood and -log(0)=Inf for negative log-likelihood.

Infinite and missing sample values are dropped.

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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

### See Also

kernels, kfun, jitter, density and bw.nrd0

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, kdengpdcon, kdengpd, kden

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkdengpd, gkg, kdengpdcon, kdengpd, kden

Other bckden: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, kden

Other fkden: kden

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 5), ylim = c(0,0.6))
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$bw), 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"))

par(mfrow = c(2, 1))

# 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 cv-likelihood evaluation, but still include them in KDE within likelihood
# often gives better bandwidth (see MacDonald et al (2011) for justification)
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 — *MLE Fitting of Kernel Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model*

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

lkdengpd(x, lambda = NULL, u = 0, sigmau = 1, xi = 0,
  phiu = TRUE, bw = NULL, kernel = "gaussian", log = TRUE)

nlkdengpd(pvector, x, phiu = TRUE, kernel = "gaussian",
  finitelik = FALSE)

proflukdengpd(u, pvector, x, phiu = TRUE, kernel = "gaussian",
  method = "BFGS", control = list(maxit = 10000), finitelik = TRUE,
  ...)

nlukdengpd(pvector, u, x, phiu = TRUE, kernel = "gaussian",
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| kernel | kernel name (default = "gaussian") |
| add.jitter | logical, whether jitter is needed for rounded kernel centres |
| 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` |
| `lambda` | scalar bandwidth for kernel (as half-width of kernel) |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `bw` | scalar bandwidth for kernel (as standard deviations of kernel) |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

The extreme value mixture model with kernel density estimate for 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.

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is (`lambda`, `u`, `sigmau`, `xi`) if threshold is also estimated and (`lambda`, `sigmau`, `xi`) for profile likelihood or fixed threshold approach.

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD component. See help for `fkden` for details, type `help fkden`.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default used in the likelihood fitting. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

## Value

Log-likelihood is given by `lkdengpd` and it's wrappers for negative log-likelihood from `nlkdengpd` and `nlukdengpd`. Profile likelihood for single threshold given by `proflukdengpd`. Fitting function `fkdengpd` returns a simple list with the following elements

| | |
|---|---|
| `call:` | `optim` call |
| `x:` | data vector `x` |
| `init:` | `pvector` |
| `fixedu:` | fixed threshold, logical |
| `useq:` | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq:` | profile negative log-likelihood at each threshold in useq |
| `optim:` | complete `optim` output |
| `mle:` | vector of MLE of parameters |
| `cov:` | variance-covariance matrix of MLE of parameters |
| `se:` | vector of standard errors of MLE of parameters |
| `rate:` | `phiu` to be consistent with `evd` |
| `nllh:` | minimum negative log-likelihood |
| `n:` | total sample size |
| `lambda:` | MLE of lambda (kernel half-width) |
| `u:` | threshold (fixed or MLE) |
| `sigmau:` | MLE of GPD scale |
| `xi:` | MLE of GPD shape |
| `phiu:` | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu:` | standard error of MLE of tail fraction |

| | |
|---|---|
| bw: | MLE of bw (kernel standard deviations) |
| kernel: | kernel name |

## Warning

See important warnings about cross-validation likelihood estimation in fkden, type help fkden.

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

The data and kernel centres are both vectors. Infinite and missing sample values (and kernel centres) are dropped.

When pvector=NULL then the initial values are:

- normal reference rule for bandwidth, using the bw.nrd0 function, which is consistent with the density function. At least two kernel centres must be provided as the variance needs to be estimated.
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

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

## References

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

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpdcon, fkden, kdengpdcon, kdengpd, kden

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkden, gkg, kdengpdcon, kdengpd, kden

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fgkgcon, fkdengpdcon, gkgcon, kdengpdcon, kdengpd

Other gkg: fgkgcon, fgkg, gkgcon, gkg, kdengpd, kden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, gkg, kdengpd, kden

Other fkdengpd: kdengpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model based tail fraction
fit = fkdengpd(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dkdengpd(xx, x, lambda, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = fkdengpd(x, phiu = FALSE)
with(fit2, lines(xx, dkdengpd(xx, x, lambda, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fkdengpd(x, useq = seq(0, 2, length = 20))
fitfix = fkdengpd(x, useq = seq(0, 2, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dkdengpd(xx, x, lambda, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dkdengpd(xx, x, lambda, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dkdengpd(xx, x, lambda, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fkdengpdcon *MLE Fitting of Kernel Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

lkdengpdcon(x, lambda = NULL, u = 0, xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", log = TRUE)

nlkdengpdcon(pvector, x, phiu = TRUE, kernel = "gaussian",
  finitelik = FALSE)

proflukdengpdcon(u, pvector, x, phiu = TRUE, kernel = "gaussian",
  method = "BFGS", control = list(maxit = 10000), finitelik = TRUE,
  ...)

nlukdengpdcon(pvector, u, x, phiu = TRUE, kernel = "gaussian",
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| kernel | kernel name (default = "gaussian") |
| add.jitter | logical, whether jitter is needed for rounded kernel centres |
| 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 |
| lambda | scalar bandwidth for kernel (as half-width of kernel) |
| u | scalar threshold value |
| xi | scalar shape parameter |
| bw | scalar bandwidth for kernel (as standard deviations of kernel) |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

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

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dkdengpdcon for details, type help kdengpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (lambda, u, xi) if threshold is also estimated and (lambda, xi) for profile likelihood or fixed threshold approach.

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD component. See help for fkden for details, type help fkden.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default used in the likelihood fitting. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

## Value

Log-likelihood is given by lkdengpdcon and it's wrappers for negative log-likelihood from nlkdengpdcon and nlukdengpdcon. Profile likelihood for single threshold given by proflukdengpdcon. Fitting function fkdengpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| lambda: | MLE of lambda (kernel half-width) |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale (estimated from other parameters) |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

| | |
|---|---|
| bw: | MLE of bw (kernel standard deviations) |
| kernel: | kernel name |

## Warning

See important warnings about cross-validation likelihood estimation in fkden, type help fkden.

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd. Based on code by Anna MacDonald produced for MATLAB.

## Note

The data and kernel centres are both vectors. Infinite and missing sample values (and kernel centres) are dropped.

When pvector=NULL then the initial values are:

- normal reference rule for bandwidth, using the bw.nrd0 function, which is consistent with the density function. At least two kernel centres must be provided as the variance needs to be estimated.
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

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

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

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

### See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package. fgpd and gpd.

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpd, fkden, gkg, kdengpdcon, kdengpd, kden

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fgkgcon, fkdengpd, gkgcon, kdengpdcon, kdengpd

Other gkgcon: fgkgcon, fgkg, gkgcon, gkg, kdengpdcon

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, gkgcon, kdengpdcon

Other fkdengpdcon: kdengpdcon

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Continuity constraint
fit = fkdengpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dkdengpdcon(xx, x, lambda, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fkdengpdcon(x)
with(fit2, lines(xx, dkdengpdcon(xx, x, lambda, u, xi), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fkdengpdcon(x, useq = seq(0, 2, length = 20))
fitfix = fkdengpdcon(x, useq = seq(0, 2, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dkdengpdcon(xx, x, lambda, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dkdengpdcon(xx, x, lambda, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dkdengpdcon(xx, x, lambda, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## 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 log-normal for bulk distribution upto the threshold and conditional GPD above threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

llognormgpd(x, lnmean = 0, lnsd = 1, u = qlnorm(0.9, lnmean, lnsd),
  sigmau = sqrt(lnmean) * lnsd, xi = 0, phiu = TRUE, log = TRUE)

nllognormgpd(pvector, x, phiu = TRUE, finitelik = FALSE)

proflulognormgpd(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlulognormgpd(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in `useq`, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in `useq`) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `lnmean` | scalar mean on log scale |
| `lnsd` | scalar standard deviation on log scale (positive) |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## 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.

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is (`lnmean`, `lnsd`, `u`, `sigmau`, `xi`) if threshold is also estimated and (`lnmean`, `lnsd`, `sigmau`, `xi`) for profile likelihood or fixed threshold approach.

Non-positive data are ignored.

## Value

Log-likelihood is given by `llognormgpd` and it's wrappers for negative log-likelihood from `nllognormgpd` and `nlulognormgpd`. Profile likelihood for single threshold given by `proflulognormgpd`. Fitting function `flognormgpd` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `lnmean`: | MLE of log-normal mean |
| `lnsd`: | MLE of log-normal shape |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale |
| `xi`: | MLE of GPD shape |
| `phiu`: | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu`: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

## Note

When `pvector=NULL` then the initial values are:

- MLE of log-normal parameters assuming entire population is log-normal; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

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

## References

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

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

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

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

dlnorm, fgpd and gpd

Other lognormgpd: flognormgpdcon, lognormgpdcon, lognormgpd

Other lognormgpdcon: flognormgpdcon, lognormgpdcon, lognormgpd

Other normgpd: fgng, fhpd, fitmnormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other flognormgpd: lognormgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rlnorm(1000)
xx = seq(-0.1, 10, 0.01)
y = dlnorm(xx)

# Bulk model based tail fraction
fit = flognormgpd(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10), ylim = c(0, 0.8))
lines(xx, y)
with(fit, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = flognormgpd(x, phiu = FALSE)
with(fit2, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
```

```
fitu = flognormgpd(x, useq = seq(1, 5, length = 20))
fitfix = flognormgpd(x, useq = seq(1, 5, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10), ylim = c(0, 0.8))
lines(xx, y)
with(fit, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

`flognormgpdcon` *MLE Fitting of log-normal Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with log-normal for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflulognormgpdcon(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlulognormgpdcon(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |

| | |
|---|---|
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| lnmean | scalar mean on log scale |
| lnsd | scalar standard deviation on log scale (positive) |
| u | scalar threshold value |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

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

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dlognormgpdcon for details, type help lognormgpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (lnmean, lnsd, u, xi) if threshold is also estimated and (lnmean, lnsd, xi) for profile likelihood or fixed threshold approach.

Non-positive data are ignored.

## Value

Log-likelihood is given by llognormgpdcon and it's wrappers for negative log-likelihood from nllognormgpdcon and nlulognormgpdcon. Profile likelihood for single threshold given by proflulognormgpdcon. Fitting function flognormgpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| lnmean: | MLE of log-normal mean |
| lnsd: | MLE of log-normal standard deviation |
| u: | threshold (fixed or MLE) |

| | |
|---|---|
| sigmau: | MLE of GPD scale (estimated from other parameters) |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- MLE of log-normal parameters assuming entire population is log-normal; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

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

dlnorm, fgpd and gpd

Other lognormgpd: flognormgpd, lognormgpdcon, lognormgpd

Other lognormgpdcon: flognormgpd, lognormgpdcon, lognormgpd

Other normgpdcon: fgngcon, fhpdcon, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, normgpdcon, normgpd

Other flognormgpdcon: lognormgpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rlnorm(1000)
xx = seq(-0.1, 10, 0.01)
y = dlnorm(xx)

# Continuity constraint
fit = flognormgpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10), ylim = c(0, 0.8))
lines(xx, y)
with(fit, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = flognormgpd(x, phiu = FALSE)
with(fit2, lines(xx, dlognormgpd(xx, lnmean, lnsd, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = flognormgpdcon(x, useq = seq(1, 5, length = 20))
fitfix = flognormgpdcon(x, useq = seq(1, 5, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10), ylim = c(0, 0.8))
lines(xx, y)
with(fit, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

fmgamma *MLE Fitting of Mixture of Gammas Using EM Algorithm*

---

## Description

Maximum likelihood estimation for fitting the mixture of gammas distribution using the EM algorithm.

## Usage

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

```
lmgamma(x, mgshape, mgscale, mgweight, log = TRUE)

nlmgamma(pvector, x, M, finitelik = FALSE)

nlEMmgamma(pvector, tau, mgweight, x, M, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `M` | number of gamma components in mixture |
| `pvector` | vector of initial values of GPD parameters (`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` |
| `mgshape` | mgamma shape (positive) as vector of length `M` |
| `mgscale` | mgamma scale (positive) as vector of length `M` |
| `mgweight` | mgamma weights (positive) as vector of length `M` |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |
| `tau` | matrix of posterior probability of being in each component (`nxM` where `n` is `length(x)`) |

## Details

The weighted mixture of gammas distribution is fitted to the entire dataset by maximum likelihood estimation using the EM algorithm. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

The expectation step estimates the expected probability of being in each component conditional on gamma component parameters. The maximisation step optimizes the negative log-likelihood conditional on posterior probabilities of each observation being in each component.

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector, as often there are numerous local modes. This is an inherent feature of such models and the EM algorithm. The EM algorithm is guaranteed to reach the maximum of the local mode. Multiple initial values should be considered to find the global maximum. If the `pvector` is input as `NULL` then random component probabilities are simulated as the initial values, so multiple such runs should be run to check the sensitivity to initial values. Alternatives to black-box likelihood optimisers (e.g. simulated annealing), or moving to computational Bayesian inference, are also worth considering.

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector `pvector` must be specified in the negative log-likelihood functions `nlmgamma` and `nlEMmgamma`.

Log-likelihood calculations are carried out in `lmgamma`, which takes parameters as inputs in the same form as the distribution functions. The negative log-likelihood function `nlmgamma` is a wrapper for `lmgamma` designed towards making it useable for optimisation, i.e. `nlmgamma` has complete parameter vector as first input. Similarly, for the maximisation step negative log-likelihood `nlEMmgamma`, which also has the second input as the component probability vector `mgweight`.

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

The function lnormgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

The default optimisation algorithm in the "maximisation step" 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 or for common indicators of lack of convergence (e.g. any estimated parameters same as initial values).

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.

Suppose there are $M$ gamma components with (scalar) shape and scale parameters and weight for each component. Only $M-1$ are to be provided in the initial parameter vector, as the $M$th components weight is uniquely determined from the others.

For the fitting function fmgamma and negative log-likelihood functions the parameter vector pvector is a 3*M-1 length vector containing all $M$ gamma component shape parameters first, followed by the corresponding $M$ gamma scale parameters, then all the corresponding $M-1$ probability weight parameters. The full parameter vector is then c(mgshape,mgscale,mgweight[1:(M-1)]).

For the maximisation step negative log-likelihood functions the parameter vector pvector is a 2*M length vector containing all $M$ gamma component shape parameters first followed by the corresponding $M$ gamma scale parameters. The partial parameter vector is then c(mgshape,mgscale).

For identifiability purposes the mean of each gamma component must be in ascending in order. If the initial parameter vector does not satisfy this constraint then an error is given.

Non-positive data are ignored as likelihood is infinite, except for gshape=1.

## Value

Log-likelihood is given by lmgamma and it's wrapper for negative log-likelihood from nlmgamma. The conditional negative log-likelihood using the posterior probabilities is given by nlEMmgamma. Fitting function fmgammagpd using EM algorithm 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 parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| M: | number of gamma components |
| mgshape: | MLE of gamma shapes |
| mgscale: | MLE of gamma scales |
| mgweight: | MLE of gamma weights |
| EMresults: | EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation st |
| posterior: | posterior probabilites |

## Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

## Note

In the fitting and profile likelihood functions, when `pvector=NULL` then the default initial values are obtained under the following scheme:

- number of sample from each component is simulated from symmetric multinomial distribution;
- sample data is then sorted and split into groups of this size (works well when components have modes which are well separated);
- for data within each component approximate MLE's for the gamma shape and scale parameters are estimated.

The `lmgamma`, `nlmgamma` and `nlEMmgamma` have no defaults.

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`.

Invalid parameter ranges will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

Infinite and missing sample values are dropped.

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

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

## See Also

`dgamma` and `gammamixEM` in `mixtools` package

Other gammagpd: `fgammagpdcon`, `fgammagpd`, `fmgammagpd`, `gammagpdcon`, `gammagpd`, `mgammagpd`

Other mgamma: `fmgammagpdcon`, `fmgammagpd`, `mgammagpdcon`, `mgammagpd`, `mgamma`

Other mgammagpd: `fgammagpd`, `fmgammagpdcon`, `fmgammagpd`, `gammagpd`, `mgammagpdcon`, `mgammagpd`, `mgamma`

Other mgammagpdcon: `fgammagpdcon`, `fmgammagpdcon`, `fmgammagpd`, `gammagpdcon`, `mgammagpdcon`, `mgammagpd`, `mgamma`

Other fmgamma: `mgamma`

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = c(rgamma(1000, shape = 1, scale = 1), rgamma(3000, shape = 6, scale = 2))
xx = seq(-1, 40, 0.01)
y = (dgamma(xx, shape = 1, scale = 1) + 3 * dgamma(xx, shape = 6, scale = 2))/4

# Fit by EM algorithm
fit = fmgamma(x, M = 2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit, lines(xx, dmgamma(xx, mgshape, mgscale, mgweight), col="red"))

## End(Not run)
```

---

fmgammagpd *MLE Fitting of Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model using the EM algorithm.*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with mixture of gammas for bulk distribution upto the threshold and conditional GPD above threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

```
fmgammagpd(x, M, phiu = TRUE, useq = NULL, fixedu = FALSE,
  pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

lmgammagpd(x, mgshape, mgscale, mgweight, u, sigmau, xi, phiu = TRUE,
  log = TRUE)

nlmgammagpd(pvector, x, M, phiu = TRUE, finitelik = FALSE)

nlumgammagpd(pvector, u, x, M, phiu = TRUE, finitelik = FALSE)

nlEMmgammagpd(pvector, tau, mgweight, x, M, phiu = TRUE,
  finitelik = FALSE)

proflumgammagpd(u, pvector, x, M, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nluEMmgammagpd(pvector, u, tau, mgweight, x, M, phiu = TRUE,
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| M | number of gamma components in mixture |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| mgshape | mgamma shape (positive) as vector of length M |
| mgscale | mgamma scale (positive) as vector of length M |
| mgweight | mgamma weights (positive) as vector of length M |
| u | scalar threshold value |
| sigmau | scalar scale parameter (positive) |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| tau | matrix of posterior probability of being in each component (nxM where n is length(x)) |

## Details

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

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The expectation step estimates the expected probability of being in each component conditional on gamma component parameters. The maximisation step optimizes the negative log-likelihood conditional on posterior probabilities of each observation being in each component.

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector, as often there are numerous local modes. This is an inherent feature of such models and the EM algorithm. The EM algorithm is guaranteed to reach the maximum of the local mode. Multiple initial values should be considered to find the global maximum. If the pvector is input as NULL then random component probabilities are simulated as the initial values, so multiple such runs should be run to check the sensitivity to initial values. Alternatives to black-box likelihood optimisers (e.g. simulated annealing), or moving to computational Bayesian inference, are also worth considering.

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood functions nlmgammagpd and nlEMmgammagpd.

Log-likelihood calculations are carried out in `lmgammagpd`, which takes parameters as inputs in the same form as the distribution functions. The negative log-likelihood function `nlmgammagpd` is a wrapper for `lmgammagpd` designed towards making it useable for optimisation, i.e. `nlmgammagpd` has complete parameter vector as first input. Though it is not directly used for optimisation here, as the EM algorithm due to mixture of gammas for the bulk component of this model

The EM algorithm for the mixture of gammas utilises the negative log-likelihood function `nlEMmgammagpd` which takes the posterior probabilities $tau$ and component probabilities `mgweight` as secondary inputs.

The profile likelihood for the threshold `proflumgammagpd` also implements the EM algorithm for the mixture of gammas, utilising the negative log-likelihood function `nluEMmgammagpd` which takes the threshold, posterior probabilities $tau$ and component probabilities `mgweight` as secondary inputs.

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

Suppose there are $M$ gamma components with (scalar) shape and scale parameters and weight for each component. Only $M - 1$ are to be provided in the initial parameter vector, as the $M$th components weight is uniquely determined from the others.

The initial parameter vector `pvector` always has the $M$ gamma component shape parameters followed by the corresponding $M$ gamma scale parameters. However, subsets of the other parameters are needed depending on which function is being used:

- fmgammagpd - `c(mgshape,mgscale,mgweight[1:(M-1)],u,sigmau,xi)`
- nlmgammagpd - `c(mgshape,mgscale,mgweight[1:(M-1)],u,sigmau,xi)`
- nlumgammagpd and proflumgammagpd - `c(mgshape,mgscale,mgweight[1:(M-1)],sigmau,xi)`
- nlEMmgammagpd - `c(mgshape,mgscale,u,sigmau,xi)`
- nluEMmgammagpd - `c(mgshape,mgscale,sigmau,xi)`

Notice that when the component probability weights are included only the first $M - 1$ are specified, as the remaining one can be uniquely determined from these. Where some parameters are left out, they are always taken as secondary inputs to the functions.

For identifiability purposes the mean of each gamma component must be in ascending in order. If the initial parameter vector does not satisfy this constraint then an error is given.

Non-positive data are ignored as likelihood is infinite, except for `gshape=1`.

## Value

Log-likelihood is given by `lmgammagpd` and it's wrappers for negative log-likelihood from `nlmgammagpd` and `nlumgammagpd`. The conditional negative log-likelihoods using the posterior probabilities are `nlEMmgammagpd` and `nluEMmgammagpd`. Profile likelihood for single threshold given by `proflumgammagpd` using EM algorithm. Fitting function `fmgammagpd` using EM algorithm returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |

| | |
|---|---|
| `se:` | vector of standard errors of MLE of parameters |
| `rate:` | `phiu` to be consistent with `evd` |
| `nllh:` | minimum negative log-likelihood |
| `n:` | total sample size |
| `M:` | number of gamma components |
| `mgshape:` | MLE of gamma shapes |
| `mgscale:` | MLE of gamma scales |
| `mgweight:` | MLE of gamma weights |
| `u:` | threshold (fixed or MLE) |
| `sigmau:` | MLE of GPD scale |
| `xi:` | MLE of GPD shape |
| `phiu:` | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu:` | standard error of MLE of tail fraction |
| `EMresults:` | EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation st |
| `posterior:` | posterior probabilites |

## Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

## Note

In the fitting and profile likelihood functions, when `pvector=NULL` then the default initial values are obtained under the following scheme:

- number of sample from each component is simulated from symmetric multinomial distribution;
- sample data is then sorted and split into groups of this size (works well when components have modes which are well separated);
- for data within each component approximate MLE's for the gamma shape and scale parameters are estimated;
- threshold is specified as sample 90% quantile; and
- MLE of GPD parameters above threshold.

The other likelihood functions `lmgammagpd`, `nlmgammagpd`, `nlumgammagpd` and `nlEMmgammagpd` and `nluEMmgammagpd` have no defaults.

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

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

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

dgamma, fgpd and gpd

Other gammagpd: fgammagpdcon, fgammagpd, fmgamma, gammagpdcon, gammagpd, mgammagpd

Other mgamma: fmgammagpdcon, fmgamma, mgammagpdcon, mgammagpd, mgamma

Other mgammagpd: fgammagpd, fmgammagpdcon, fmgamma, gammagpd, mgammagpdcon, mgammagpd, mgamma

Other mgammagpdcon: fgammagpdcon, fmgammagpdcon, fmgamma, gammagpdcon, mgammagpdcon, mgammagpd, mgamma

Other fmgammagpd: mgammagpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

n=1000
x = c(rgamma(n*0.25, shape = 1, scale = 1), rgamma(n*0.75, shape = 6, scale = 2))
xx = seq(-1, 40, 0.01)
y = (0.25*dgamma(xx, shape = 1, scale = 1) + 0.75 * dgamma(xx, shape = 6, scale = 2))

# Bulk model based tail fraction
# very sensitive to initial values, so best to provide sensible ones
fit.noinit = fmgammagpd(x, M = 2)
fit.withinit = fmgammagpd(x, M = 2, pvector = c(1, 6, 1, 2, 0.5, 15, 4, 0.1))
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit.noinit, lines(xx, dmgammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi),
 col="red"))
abline(v = fit.noinit$u, col = "red")
with(fit.withinit, lines(xx, dmgammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi),
 col="green"))
abline(v = fit.withinit$u, col = "green")

# Parameterised tail fraction
fit2 = fmgammagpd(x, M = 2, phiu = FALSE, pvector = c(1, 6, 1, 2, 0.5, 15, 4, 0.1))
with(fit2, lines(xx, dmgammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Default pvector", "Sensible pvector",
 "Parameterised Tail Fraction"), col=c("black", "red", "green", "blue"), lty = 1)

# Fixed threshold approach
fitfix = fmgammagpd(x, M = 2, useq = 15, fixedu = TRUE,
```

```
  pvector = c(1, 6, 1, 2, 0.5, 4, 0.1))

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit.withinit, lines(xx, dmgammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi), col="red"))
abline(v = fit.withinit$u, col = "red")
with(fitfix, lines(xx, dmgammagpd(xx,mgshape, mgscale, mgweight, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density", "Default initial value (90% quantile)",
 "Fixed threshold approach"), col=c("black", "red", "darkgreen"), lty = 1)

## End(Not run)
```

---

fmgammagpdcon *MLE Fitting of Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint using the EM algorithm.*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with mixture of gammas for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

```
fmgammagpdcon(x, M, phiu = TRUE, useq = NULL, fixedu = FALSE,
  pvector = NULL, std.err = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

lmgammagpdcon(x, mgshape, mgscale, mgweight, u, xi, phiu = TRUE,
  log = TRUE)

nlmgammagpdcon(pvector, x, M, phiu = TRUE, finitelik = FALSE)

nlumgammagpdcon(pvector, u, x, M, phiu = TRUE, finitelik = FALSE)

nlEMmgammagpdcon(pvector, tau, mgweight, x, M, phiu = TRUE,
  finitelik = FALSE)

proflumgammagpdcon(u, pvector, x, M, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nluEMmgammagpdcon(pvector, u, tau, mgweight, x, M, phiu = TRUE,
  finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| M | number of gamma components in mixture |

| | |
|---|---|
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| mgshape | mgamma shape (positive) as vector of length M |
| mgscale | mgamma scale (positive) as vector of length M |
| mgweight | mgamma weights (positive) as vector of length M |
| u | scalar threshold value |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |
| tau | matrix of posterior probability of being in each component (nxM where n is length(x)) |

## Details

The extreme value mixture model with weighted mixture of gammas bulk and GPD tail with continuity at threshold is fitted to the entire dataset using maximum likelihood estimation using the EM algorithm. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The expectation step estimates the expected probability of being in each component conditional on gamma component parameters. The maximisation step optimizes the negative log-likelihood conditional on posterior probabilities of each observation being in each component.

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector, as often there are numerous local modes. This is an inherent feature of such models and the EM algorithm. The EM algorithm is guaranteed to reach the maximum of the local mode. Multiple initial values should be considered to find the global maximum. If the pvector is input as NULL then random component probabilities are simulated as the initial values, so multiple such runs should be run to check the sensitivity to initial values. Alternatives to black-box likelihood optimisers (e.g. simulated annealing), or moving to computational Bayesian inference, are also worth considering.

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood functions nlmgammagpdcon and nlEMmgammagpdcon.

Log-likelihood calculations are carried out in lmgammagpdcon, which takes parameters as inputs in the same form as the distribution functions. The negative log-likelihood function nlmgammagpdcon is a wrapper for lmgammagpdcon designed towards making it useable for optimisation, i.e. nlmgammagpdcon has complete parameter vector as first input. Though it is not directly used for optimisation here, as the EM algorithm due to mixture of gammas for the bulk component of this model

The EM algorithm for the mixture of gammas utilises the negative log-likelihood function nlEMmgammagpdcon which takes the posterior probabilities $tau$ and component probabilities mgweight as secondary inputs.

The profile likelihood for the threshold proflumgammagpdcon also implements the EM algorithm for the mixture of gammas, utilising the negative log-likelihood function nluEMmgammagpdcon which takes the threshold, posterior probabilities $tau$ and component probabilities mgweight as secondary inputs.

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

Suppose there are $M$ gamma components with (scalar) shape and scale parameters and weight for each component. Only $M - 1$ are to be provided in the initial parameter vector, as the $M$th components weight is uniquely determined from the others.

The initial parameter vector pvector always has the $M$ gamma component shape parameters followed by the corresponding $M$ gamma scale parameters. However, subsets of the other parameters are needed depending on which function is being used:

- fmgammagpdcon - c(mgshape,mgscale,mgweight[1:(M-1)],u,xi)
- nlmgammagpdcon - c(mgshape,mgscale,mgweight[1:(M-1)],u,xi)
- nlumgammagpdcon and proflumgammagpdcon - c(mgshape,mgscale,mgweight[1:(M-1)],xi)
- nlEMmgammagpdcon - c(mgshape,mgscale,u,xi)
- nluEMmgammagpdcon - c(mgshape,mgscale,xi)

Notice that when the component probability weights are included only the first $M - 1$ are specified, as the remaining one can be uniquely determined from these. Where some parameters are left out, they are always taken as secondary inputs to the functions.

For identifiability purposes the mean of each gamma component must be in ascending in order. If the initial parameter vector does not satisfy this constraint then an error is given.

Non-positive data are ignored as likelihood is infinite, except for gshape=1.

## Value

Log-likelihood is given by lmgammagpdcon and it's wrappers for negative log-likelihood from nlmgammagpdcon and nlumgammagpdcon. The conditional negative log-likelihoods using the posterior probabilities are nlEMmgammagpdcon and nluEMmgammagpdcon. Profile likelihood for single threshold given by proflumgammagpdcon using EM algorithm. Fitting function fmgammagpdcon using EM algorithm returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |

| | |
|---|---|
| `n:` | total sample size |
| `M:` | number of gamma components |
| `mgshape:` | MLE of gamma shapes |
| `mgscale:` | MLE of gamma scales |
| `mgweight:` | MLE of gamma weights |
| `u:` | threshold (fixed or MLE) |
| `sigmau:` | MLE of GPD scale |
| `xi:` | MLE of GPD shape |
| `phiu:` | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu:` | standard error of MLE of tail fraction |
| `EMresults:` | EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation st |
| `posterior:` | posterior probabilites |

## Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

## Note

In the fitting and profile likelihood functions, when `pvector=NULL` then the default initial values are obtained under the following scheme:

- number of sample from each component is simulated from symmetric multinomial distribution;
- sample data is then sorted and split into groups of this size (works well when components have modes which are well separated);
- for data within each component approximate MLE's for the gamma shape and scale parameters are estimated;
- threshold is specified as sample 90% quantile; and
- MLE of GPD shape parameter above threshold.

The other likelihood functions `lmgammagpdcon`, `nlmgammagpdcon`, `nlumgammagpdcon` and `nlEMmgammagpdcon` and `nluEMmgammagpdcon` have no defaults.

## Author(s)

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

## References

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

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

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

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

dgamma, fgpd and gpd

Other gammagpdcon: fgammagpdcon, fgammagpd, gammagpdcon, gammagpd, mgammagpdcon

Other mgamma: fmgammagpd, fmgamma, mgammagpdcon, mgammagpd, mgamma

Other mgammagpd: fgammagpd, fmgammagpd, fmgamma, gammagpd, mgammagpdcon, mgammagpd, mgamma

Other mgammagpdcon: fgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, mgammagpdcon, mgammagpd, mgamma

Other fmgammagpdcon: mgammagpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

n=1000
x = c(rgamma(n*0.25, shape = 1, scale = 1), rgamma(n*0.75, shape = 6, scale = 2))
xx = seq(-1, 40, 0.01)
y = (0.25*dgamma(xx, shape = 1, scale = 1) + 0.75 * dgamma(xx, shape = 6, scale = 2))

# Bulk model based tail fraction
# very sensitive to initial values, so best to provide sensible ones
fit.noinit = fmgammagpdcon(x, M = 2)
fit.withinit = fmgammagpdcon(x, M = 2, pvector = c(1, 6, 1, 2, 0.5, 15, 0.1))
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit.noinit, lines(xx, dmgammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="red"))
abline(v = fit.noinit$u, col = "red")
with(fit.withinit, lines(xx, dmgammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="green"))
abline(v = fit.withinit$u, col = "green")

# Parameterised tail fraction
fit2 = fmgammagpdcon(x, M = 2, phiu = FALSE, pvector = c(1, 6, 1, 2, 0.5, 15, 0.1))
with(fit2, lines(xx, dmgammagpdcon(xx, mgshape, mgscale, mgweight, u, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Default pvector", "Sensible pvector",
 "Parameterised Tail Fraction"), col=c("black", "red", "green", "blue"), lty = 1)

# Fixed threshold approach
fitfix = fmgammagpdcon(x, M = 2, useq = 15, fixedu = TRUE,
   pvector = c(1, 6, 1, 2, 0.5, 0.1))

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit.withinit, lines(xx, dmgammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="red"))
abline(v = fit.withinit$u, col = "red")
```

```
with(fitfix, lines(xx, dmgammagpdcon(xx,mgshape, mgscale, mgweight, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density", "Default initial value (90% quantile)",
 "Fixed threshold approach"), col=c("black", "red", "darkgreen"), 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. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflunormgpd(u, pvector = NULL, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlunormgpd(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |

| | |
|---|---|
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| u | scalar threshold value |
| sigmau | scalar scale parameter (positive) |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## 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 optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector (particularly the threshold), as often there are numerous local modes where multiple thresholds give similar fits. This is an inherent feature of such models. Options are provided by the arguments `pvector`, `useq` and `fixedu` to implement various commonly used likelihood inference approaches for such models:

1. (default) `pvector=NULL`, `useq=NULL` and `fixedu=FALSE` - to set initial value for threshold at 90% quantile along with usual defaults for other parameters as defined in Notes below. Standard likelihood optimisation is used;
2. `pvector=c(nmean,nsd,u,sigmau,xi)` - where initial values of all 5 parameters are manually set. Standard likelihood optimisation is used;
3. `useq` as vector - to specify a sequence of thresholds at which to evaluate profile likelihood and extract threshold which gives maximum profile likelihood; or
4. `useq` as scalar - to specify a single value for threshold to be considered.

In options (3) and (4) the threshold can be treated as:

- initial value for maximum likelihood estimation when `fixedu=FALSE`, using either profile likelihood estimate (3) or pre-chosen threshold (4); or
- a fixed threshold with MLE for other parameters when `fixedu=TRUE`, using either profile likelihood estimate (3) or pre-chosen threshold (4).

The latter approach can be used to implement the traditional fixed threshold modelling approach with threshold pre-chosen using, for example, graphical diagnostics. Further, in either such case (3) or (4) the `pvector` could be:

- `NULL` for usual defaults for other four parameters, defined in Notes below; or
- vector of initial values for remaining 4 parameters (`nmean`, `nsd`, `sigmau`, `xi`).

If the threshold is treated as fixed, then the likelihood is separable between the bulk and tail components. However, in practice we have found black-box optimisation of the combined likelihood works sufficiently well, so is used herein.

The following functions are provided:

- `fnormgpd` - maximum likelihood fitting with all the above options;
- `lnormgpd` - log-likelihood;
- `nlnormgpd` - negative log-likelihood;
- `proflunormgpd` - profile likelihood for given threshold; and

- nlunormgpd - negative log-likelihood (threshold specified separately).

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions.

Defaults values for the parameter vector pvector are given in the fitting fnormgpd and profile likelihood functions proflunormgpd. The parameter vector pvector must be specified in the negative log-likelihood functions nlnormgpd and nlunormgpd. The threshold u must also be specified in the profile likelihood function proflunormgpd and nlunormgpd.

Log-likelihood calculations are carried out in lnormgpd, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood functions nlnormgpd and nlunormgpd are wrappers for likelihood function lnormgpd designed towards optimisation, i.e. nlnormgpd has vector of all 5 parameters as first input and nlunormgpd has threshold as second input and vector of remaining 4 parameters as first input. The profile likelihood function proflunormgpd has threshold u as the first input, to permit use of sapply function to evaluate profile likelihood over vector of potential thresholds.

The tail fraction phiu is treated separately to the other parameters, to allow for all it's representations. In the fitting fnormgpd and profile likelihood function proflunormgpd it is logical:

- default value phiu=TRUE - tail fraction specified by normal survivor function phiu = 1 -pnorm(u,nmean,nsd) and standard error is output as NA; and
- phiu=FALSE - treated as extra parameter estimated using the MLE which is the sample proportion above the threshold and standard error is output.

In the likelihood functions lnormgpd, nlnormgpd and nlunormgpd it can be logical or numeric:

- logical - same as for fitting functions with default value phiu=TRUE.
- numeric - any value over range $(0, 1)$. Notice that the tail fraction probability cannot be 0 or 1 otherwise there would be no contribution from either tail or bulk components respectively.

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 function lnormgpd carries out the calculations for the log-likelihood directly, which can be exponentiated to give actual likelihood using (log=FALSE).

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 or for common indicators of lack of convergence (e.g. any estimated parameters same as initial values).

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

Log-likelihood is given by lnormgpd and it's wrappers for negative log-likelihood from nlnormgpd and nlunormgpd. Profile likelihood for single threshold given by proflunormgpd. Fitting function fnormgpd returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |

| | |
|---|---|
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of 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 increase usability.

## Acknowledgments

These functions are deliberately similar in syntax and functionality to the commonly used functions in the ismev and evd packages for which their author's contributions are gratefully acknowledged.

Anna MacDonald and Xin Zhao laid some of the groundwork with programs they wrote for MATLAB.

Clement Lee and Emma Eastoe suggested providing inbuilt profile likelihood estimation for threshold and fixed threshold approach.

## Note

Unlike most of the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and phiu.

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- threshold 90% quantile (not relevant for profile likelihood or fixed threshold approaches);
- MLE of GPD parameters above threshold.

Avoid setting the starting value for the shape parameter to xi=0 as depending on the optimisation method it may be get stuck.

A default value for the tail fraction phiu=TRUE is given. The lnormgpd also has the usual defaults for the other parameters, but nlnormgpd and nlunormgpd has no defaults.

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.

Invalid parameter ranges will give `0` for likelihood, `log(0)=-Inf` for log-likelihood and `-log(0)=Inf` for negative log-likelihood.

Due to symmetry, the lower tail can be described by GPD by negating the data/quantiles.

Infinite and missing sample values are dropped.

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

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

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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

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

dnorm, fgpd and gpd

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpdcon, gngcon, gng, hpdcon, hpd, normgpdcon, normgpd

Other gng: fgngcon, fgng, fitmgng, gngcon, gng, itmgng, normgpd

Other fnormgpd: normgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Bulk model based tail fraction
fit = fnormgpd(x)
```

```
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = fnormgpd(x, phiu = FALSE)
with(fit2, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fnormgpd(x, useq = seq(0, 3, length = 20))
fitfix = fnormgpd(x, useq = seq(0, 3, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topleft", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fnormgpdcon *MLE Fitting of Normal Bulk and GPD Tail Extreme Value Mixture Model with Single 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 continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

proflunormgpdcon(u, pvector, x, phiu = TRUE, method = "BFGS",
```

```
  control = list(maxit = 10000), finitelik = TRUE, ...)

nlunormgpdcon(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| nmean | scalar normal mean |
| nsd | scalar normal standard deviation (positive) |
| u | scalar threshold value |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

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

See help for fnormgpd for full details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dnormgpdcon for details, type help normgpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (nmean, nsd, u, xi) if threshold is also estimated and (nmean, nsd, xi) for profile likelihood or fixed threshold approach.

## Value

Log-likelihood is given by lnormgpdcon and it's wrappers for negative log-likelihood from nlnormgpdcon and nlunormgpdcon. Profile likelihood for single threshold given by proflunormgpdcon. Fitting function fnormgpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |

| | |
|---|---|
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| nmean: | MLE of normal mean |
| nsd: | MLE of normal standard deviation |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale (estimated from other parameters) |
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- MLE of normal parameters assuming entire population is normal; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

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

dnorm, fgpd and gpd

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, normgpdcon, normgpd

Other gngcon: fgngcon, fgng, gngcon, gng, normgpdcon

Other fnormgpdcon: normgpdcon

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Continuity constraint
fit = fnormgpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fnormgpd(x)
with(fit2, lines(xx, dnormgpd(xx, nmean, nsd, u, sigmau, xi), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fnormgpdcon(x, useq = seq(0, 3, length = 20))
fitfix = fnormgpdcon(x, useq = seq(0, 3, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topleft", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fpsden *MLE Fitting of P-splines Density Estimator*

## Description

Maximum likelihood estimation for P-splines density estimation. Histogram binning produces frequency counts, which are modelled by constrained B-splines in a Poisson regression. A penalty based on differences in the sequences B-spline coefficients is used to smooth/interpolate the counts. Iterated weighted least squares (IWLS) for a mixed model representation of the P-splines regression, conditional on a particular penalty coefficient, is used for estimating the B-spline coefficients. Leave-one-out cross-validation deviances are available for estimation of the penalty coefficient.

## Usage

```
fpsden(x, lambdaseq = NULL, breaks = NULL, xrange = NULL,
  nseg = 10, degree = 3, design.knots = NULL, ord = 2)

lpsden(x, beta = NULL, bsplines = NULL, nbinwidth = 1, log = TRUE)

nlpsden(pvector, x, bsplines = NULL, nbinwidth = 1,
  finitelik = FALSE)

cvpsden(lambda = 1, counts, bsplines, ord = 2)

iwlspsden(counts, bsplines, ord = 2, lambda = 10)
```

## Arguments

| | |
|---|---|
| x | quantiles |
| lambdaseq | vector of $\lambda$'s (or scalar) to be considered in profile likelihood. Required. |
| breaks | histogram breaks (as in hist function) |
| xrange | vector of minimum and maximum of B-spline (support of density) |
| nseg | number of segments between knots |
| degree | degree of B-splines (0 is constant, 1 is linear, etc.) |
| design.knots | spline knots for splineDesign function |
| ord | order of difference used in the penalty term |
| beta | vector of B-spline coefficients (required) |
| bsplines | matrix of B-splines |
| nbinwidth | scaling to convert count frequency into proper density |
| 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 |
| lambda | penalty coefficient |
| counts | counts from histogram binning |

## Details

The P-splines density estimator is fitted using maximum likelihood estimation, following the approach of Eilers and Marx (1996). Histogram binning produces frequency counts, which are modelled by constrained B-splines in a Poisson regression. A penalty based on differences in the sequences B-spline coefficients is used to smooth/interpolate the counts.

The B-splines are defined as in Eiler and Marx (1996), so that those are meet the boundary are simply shifted and truncated version of the internal B-splines. No renormalisation is carried out. They are not "natural" B-spline which are also commonly in use. Note that atural B-splines can be obtained by suitable linear combinations of these B-splines. Hence, in practice there is little difference in the fit obtained from either B-spline definition, even with the penalty constraining the coefficients. If the user desires they can force the use of natural B-splines, by prior specification of the `design.knots` with appropriate replication of the boundaries, see `dpsden`.

Iterated weighted least squares (IWLS) for a mixed model representation of the P-splines regression, conditional on a particular penalty coefficient, is used for estimating the B-spline coefficients which is equivalent to maximum likelihood estimation. Leave-one-out cross-validation deviances are available for estimation of the penalty coefficient.

The parameter vector is the B-spline coefficients `beta`, no matter whether the penalty coefficient is fixed or estimated. The penalty coefficient `lambda` is treated separately.

The log-likelihood functions `lpsden` and `nlpsden` evaluate the likelihood for the original dataset, using the fitted P-splines density estimator. The log-likelihood is output as `nllh` from the fitting function `fpsden`. They do not provide the likelihood for the Poisson regression of the histogram counts, which is usually evaluated using the deviance. The deviance (via CVMSE for Poisson counts) is also output as `cvlambda` from the fitting function `fpsden`.

The `iwlspsden` function performs the IWLS. The `cvpsden` function calculates the leave-one-out cross-validation sum of the squared errors. They are not designed to be used directly by users. No checks of the inputs are carried out.

## Value

Log-likelihood for original data is given by `lpsden` and it's wrappers for negative log-likelihood from `nlpsden`. Cross-validation sum of square of errors is provided by `cvpsden`. Poisson regression fitting by IWLS is carried out in `iwlspsden`. Fitting function `fpsden` returns a simple list with the following elements

| | |
|---|---|
| `call:` | `optim` call |
| `x:` | data vector `x` |
| `xrange:` | range of support of B-splines |
| `degree:` | degree of B-splines |
| `nseg:` | number of internal segments |
| `design.knots:` | knots used in `splineDesign` |
| `ord:` | order of penalty term |
| `binned:` | histogram results |
| `breaks:` | histogram breaks |
| `mids:` | histogram mid-bins |
| `counts:` | histogram counts |
| `nbinwidth:` | scaling factor to convert counts to density |
| `bsplines:` | B-splines matrix used for binned counts |
| `databsplines:` | B-splines matrix used for data |
| `counts:` | histogram counts |
| `lambdaseq:` | $\lambda$ vector for profile likelihood or scalar for fixed $\lambda$ |
| `cvlambda:` | CV MSE for each $\lambda$ |

| | |
|---|---|
| mle and beta: | vector of MLE of coefficients |
| nllh: | negative log-likelihood for original data |
| n: | total original sample size |
| lambda: | Estimated or fixed $\lambda$ |

## Acknowledgments

The Poisson regression and leave-one-out cross-validation functions are based on the code of Eilers and Marx (1996) available from Brian Marx's website http://statweb.lsu.edu/faculty/marx/, which is gratefully acknowledged.

## Note

The data are both vectors. Infinite and missing sample values are dropped.

No initial values for the coefficients are needed.

It is advised to specify the range of support `xrange`, using finite end-points. This is especially important when the support is bounded. By default `xrange` is simply the range of the input data `range(x)`.

Further, it is advised to always set the histogram bin `breaks`, expecially if the support is bounded. By default `10*ln(n)` equi-spaced bins are defined between `xrange`.

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

http://en.wikipedia.org/wiki/B-spline

http://statweb.lsu.edu/faculty/marx/

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science 11(2), 89-121.

## See Also

kden.

Other psden: fpsdengpd, psdengpd, psden

Other fpsden: psden

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)
```

```
# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 10 internal segments
# CV search for penalty coefficient.
fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
psdensity = exp(fit$bsplines %*% fit$mle)

hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

lines(fit$mids, psdensity/fit$nbinwidth, lwd = 2, col = "blue") # P-splines density

# check density against dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots),
                lwd = 2, col = "red", lty = 2))

# vertical lines for all knots
with(fit, abline(v = design.knots, col = "red"))

# internal knots
with(fit, abline(v = design.knots[(degree + 2):(length(design.knots) - degree - 1)], col = "blue"))

# boundary knots (support of B-splines)
with(fit, abline(v = design.knots[c(degree + 1, length(design.knots) - degree)], col = "green"))

legend("topright", c("True Density","P-spline density","Using dpsdens function"),
  col=c("black", "blue", "red"), lty = c(1, 1, 2))
legend("topleft", c("Internal Knots", "Boundaries", "Extra Knots"),
  col=c("blue", "green", "red"), lty = 1)

## End(Not run)
```

---

`fpsdengpd` *MLE Fitting of P-splines Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model*

---

## Description

Maximum likelihood estimation for fitting the extreme value mixture model with P-splines density estimate for bulk distribution upto the threshold and conditional GPD above threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

```
fpsdengpd(x, phiu = TRUE, useq = NULL, fixedu = FALSE,
  pvector = NULL, lambdaseq = NULL, breaks = NULL, xrange = NULL,
  nseg = 10, degree = 3, design.knots = NULL, ord = 2,
  std.err = TRUE, method = "BFGS", control = list(maxit = 10000),
  finitelik = TRUE, ...)

lpsdengpd(x, psdenx, u = NULL, sigmau = NULL, xi = 0, phiu = TRUE,
```

```
  bsplinefit = NULL, phib = NULL, log = TRUE)

nlpsdengpd(pvector, x, psdenx, phiu = TRUE, bsplinefit, phib = NULL,
  finitelik = FALSE)

proflupsdengpd(u, pvector, x, psdenx, phiu = TRUE, bsplinefit,
  method = "BFGS", control = list(maxit = 10000), finitelik = TRUE,
  ...)

nlupsdengpd(pvector, u, x, psdenx, phiu = TRUE,
  bsplinefit = bsplinefit, phib = NULL, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `lambdaseq` | vector of $\lambda$'s (or scalar) to be considered in profile likelihood. Required. |
| `breaks` | histogram breaks (as in `hist` function) |
| `xrange` | vector of minimum and maximum of B-spline (support of density) |
| `nseg` | number of segments between knots |
| `degree` | degree of B-splines (0 is constant, 1 is linear, etc.) |
| `design.knots` | spline knots for splineDesign function |
| `ord` | order of difference used in the penalty term |
| `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` |
| `psdenx` | P-splines based density estimate for each datapoint in x |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `bsplinefit` | list output from P-splines density fitting `fpsden` function |
| `phib` | renormalisation constant for bulk model density $(1 - \phi_u)/H(u)$, to make it integrate to `1-phiu` |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## Details

The extreme value mixture model with P-splines density estimate for bulk and GPD tail is fitted to the entire dataset. A two-stage maximum likelihood inference approach is taken. The first stage consists fitting of the P-spline density estimator, which is acheived by MLE using the `fpsden` function. The second stage, conditions on the B-spline coefficients, using MLE for the extreme value mixture model (GPD parameters and threshold, if requested). The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

See help for `fnormgpd` for details of extreme value mixture models, type `help fnormgpd`. Only the different features are outlined below for brevity.

As the second stage conditions on the Bs-pline coefficients, the full parameter vector is (`u`, `sigmau`, `xi`) if threshold is also estimated and (`sigmau`, `xi`) for profile likelihood or fixed threshold approach.

(Penalized) MLE estimation of the B-Spline coefficients is carried out using Poisson regression based on histogram bin counts. See help for `fpsden` for details, type `help fpsden`.

## Value

Log-likelihood is given by `lpsdengpd` and it's wrappers for negative log-likelihood from `nlpsdengpd` and `nlupsdengpd`. Profile likelihood for single threshold given by `proflupsdengpd`. Fitting function `fpsdengpd` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `bsplinefit`: | complete `fpsden` output |
| `psdenx`: | P-splines based density estimate for each datapoint in `x` |
| `xrange`: | range of support of B-splines |
| `degree`: | degree of B-splines |
| `nseg`: | number of internal segments |
| `design.knots`: | knots used in `splineDesign` |
| `nbinwidth`: | scaling factor to convert counts to density |
| `optim`: | complete `optim` output |
| `conv`: | indicator for "possible" convergence |
| `mle`: | vector of MLE of (GPD and threshold, if relevant) parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `beta`: | vector of MLE of B-spline coefficients |
| `lambda`: | Estimated or fixed $\lambda$ |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale |
| `xi`: | MLE of GPD shape |
| `phiu`: | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu`: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

The Poisson regression and leave-one-out cross-validation functions are based on the code of Eilers and Marx (1996) available from Brian Marx's website http://statweb.lsu.edu/faculty/marx/, which is gratefully acknowledged.

## Note

The data are both vectors. Infinite and missing sample values are dropped.

No initial values for the coefficients are needed.

It is advised to specify the range of support xrange, using finite end-points. This is especially important when the support is bounded. By default xrange is simply the range of the input data range(x).

Further, it is advised to always set the histogram bin breaks, expecially if the support is bounded. By default 10*ln(n) equi-spaced bins are defined between xrange.

When pvector=NULL then the initial values are:

- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

http://en.wikipedia.org/wiki/B-spline

http://statweb.lsu.edu/faculty/marx/

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science 11(2), 89-121.

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

fpsden, fnormgpd, fgpd and gpd

Other psden: fpsden, psdengpd, psden

Other psdengpd: psdengpd, psden

Other fpsdengpd: psdengpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-4, 4, 0.01)
y = dnorm(xx)

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 10 internal segments
# CV search for penalty coefficient.
fit = fpsdengpd(x, useq = seq(0, 3, 0.1), fixedu = TRUE,
             lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)

hist(x, freq = FALSE, breaks = breaks, xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

# P-splines+GPD
with(fit, lines(xx, dpsdengpd(xx, beta, nbinwidth,
                              u = u, sigmau = sigmau, xi = xi, design = design.knots),
                lwd = 2, col = "red"))
abline(v = fit$u, col = "red", lwd = 2, lty = 3)

# P-splines density estimate
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots),
                lwd = 2, col = "blue", lty = 2))

# vertical lines for all knots
with(fit, abline(v = design.knots, col = "red"))

# internal knots
with(fit, abline(v = design.knots[(degree + 2):(length(design.knots) - degree - 1)], col = "blue"))

# boundary knots (support of B-splines)
with(fit, abline(v = design.knots[c(degree + 1, length(design.knots) - degree)], col = "green"))

legend("topright", c("True Density","P-spline density","P-spline+GPD"),
  col=c("black", "blue", "red"), lty = c(1, 2, 1))
legend("topleft", c("Internal Knots", "Boundaries", "Extra Knots", "Threshold"),
  col=c("blue", "green", "red", "red"), lty = c(1, 1, 1, 2))

## 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. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

profluweibullgpd(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nluweibullgpd(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | vector of sample data |
| `phiu` | probability of being above threshold $(0, 1)$ or logical, see Details in help for `fnormgpd` |
| `useq` | vector of thresholds (or scalar) to be considered in profile likelihood or `NULL` for no profile likelihood |
| `fixedu` | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |
| `pvector` | vector of initial values of parameters or `NULL` for default values, see below |
| `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` |
| `wshape` | scalar Weibull shape (positive) |
| `wscale` | scalar Weibull scale (positive) |
| `u` | scalar threshold value |
| `sigmau` | scalar scale parameter (positive) |
| `xi` | scalar shape parameter |
| `log` | logical, if `TRUE` then log-likelihood rather than likelihood is output |

## 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.

See help for `fnormgpd` for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is (`wshape`, `wscale`, `u`, `sigmau`, `xi`) if threshold is also estimated and (`wshape`, `wscale`, `sigmau`, `xi`) for profile likelihood or fixed threshold approach.

Non-positive data are ignored (f(0) is infinite for wshape<1).

## Value

Log-likelihood is given by `lweibullgpd` and it's wrappers for negative log-likelihood from `nlweibullgpd` and `nluweibullgpd`. Profile likelihood for single threshold given by `profluweibullgpd`. Fitting function `fweibullgpd` returns a simple list with the following elements

| | |
|---|---|
| `call`: | `optim` call |
| `x`: | data vector `x` |
| `init`: | `pvector` |
| `fixedu`: | fixed threshold, logical |
| `useq`: | threshold vector for profile likelihood or scalar for fixed threshold |
| `nllhuseq`: | profile negative log-likelihood at each threshold in useq |
| `optim`: | complete `optim` output |
| `mle`: | vector of MLE of parameters |
| `cov`: | variance-covariance matrix of MLE of parameters |
| `se`: | vector of standard errors of MLE of parameters |
| `rate`: | `phiu` to be consistent with `evd` |
| `nllh`: | minimum negative log-likelihood |
| `n`: | total sample size |
| `wshape`: | MLE of Weibull shape |
| `wscale`: | MLE of Weibull scale |
| `u`: | threshold (fixed or MLE) |
| `sigmau`: | MLE of GPD scale |
| `xi`: | MLE of GPD shape |
| `phiu`: | MLE of tail fraction (bulk model or parameterised approach) |
| `se.phiu`: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in `fnormgpd`, type `help fnormgpd`.

## Note

When `pvector=NULL` then the initial values are:

- MLE of Weibull parameters assuming entire population is Weibull; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

### Author(s)

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

### References

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

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

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

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

dweibull, fgpd and gpd

Other weibullgpd: fitmweibullgpd, fweibullgpdcon, itmweibullgpd, weibullgpdcon, weibullgpd

Other weibullgpdcon: fweibullgpdcon, itmweibullgpd, weibullgpdcon, weibullgpd

Other itmweibullgpd: fitmweibullgpd, fweibullgpdcon, itmweibullgpd, weibullgpdcon, weibullgpd

Other fweibullgpd: weibullgpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rweibull(1000, shape = 2)
xx = seq(-0.1, 4, 0.01)
y = dweibull(xx, shape = 2)

# Bulk model based tail fraction
fit = fweibullgpd(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = fweibullgpd(x, phiu = FALSE)
with(fit2, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","Bulk Tail Fraction","Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fweibullgpd(x, useq = seq(0.5, 2, length = 20))
fitfix = fweibullgpd(x, useq = seq(0.5, 2, length = 20), fixedu = TRUE)
```

```
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

---

`fweibullgpdcon` *MLE Fitting of Weibull Bulk and GPD Tail Extreme Value Mixture Model with Single 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 continuity at threshold. With options for profile likelihood estimation for threshold and fixed threshold approach.

## Usage

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

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)

profluweibullgpdcon(u, pvector, x, phiu = TRUE, method = "BFGS",
  control = list(maxit = 10000), finitelik = TRUE, ...)

nluweibullgpdcon(pvector, u, x, phiu = TRUE, finitelik = FALSE)
```

## Arguments

| | |
|---|---|
| x | vector of sample data |
| phiu | probability of being above threshold $(0, 1)$ or logical, see Details in help for fnormgpd |
| useq | vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood |
| fixedu | logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq) |

| | |
|---|---|
| pvector | vector of initial values of parameters or NULL for default values, see below |
| 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 |
| wshape | scalar Weibull shape (positive) |
| wscale | scalar Weibull scale (positive) |
| u | scalar threshold value |
| xi | scalar shape parameter |
| log | logical, if TRUE then log-likelihood rather than likelihood is output |

## Details

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

See help for fnormgpd for details, type help fnormgpd. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for dweibullgpdcon for details, type help weibullgpdcon. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (wshape, wscale, u, xi) if threshold is also estimated and (wshape, wscale, xi) for profile likelihood or fixed threshold approach.

Negative data are ignored.

## Value

Log-likelihood is given by lweibullgpdcon and it's wrappers for negative log-likelihood from nlweibullgpdcon and nluweibullgpdcon. Profile likelihood for single threshold given by profluweibullgpdcon. Fitting function fweibullgpdcon returns a simple list with the following elements

| | |
|---|---|
| call: | optim call |
| x: | data vector x |
| init: | pvector |
| fixedu: | fixed threshold, logical |
| useq: | threshold vector for profile likelihood or scalar for fixed threshold |
| nllhuseq: | profile negative log-likelihood at each threshold in useq |
| optim: | complete optim output |
| mle: | vector of MLE of parameters |
| cov: | variance-covariance matrix of MLE of parameters |
| se: | vector of standard errors of MLE of parameters |
| rate: | phiu to be consistent with evd |
| nllh: | minimum negative log-likelihood |
| n: | total sample size |
| wshape: | MLE of Weibull shape |
| wscale: | MLE of Weibull scale |
| u: | threshold (fixed or MLE) |
| sigmau: | MLE of GPD scale (estimated from other parameters) |

| | |
|---|---|
| xi: | MLE of GPD shape |
| phiu: | MLE of tail fraction (bulk model or parameterised approach) |
| se.phiu: | standard error of MLE of tail fraction |

## Acknowledgments

See Acknowledgments in fnormgpd, type help fnormgpd.

## Note

When pvector=NULL then the initial values are:

- MLE of Weibull parameters assuming entire population is Weibull; and
- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD shape parameter above threshold.

## Author(s)

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

## References

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

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

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

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

dweibull, fgpd and gpd

Other weibullgpd: fitmweibullgpd, fweibullgpd, itmweibullgpd, weibullgpdcon, weibullgpd

Other weibullgpdcon: fweibullgpd, itmweibullgpd, weibullgpdcon, weibullgpd

Other itmweibullgpd: fitmweibullgpd, fweibullgpd, itmweibullgpd, weibullgpdcon, weibullgpd

Other fweibullgpdcon: weibullgpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

x = rweibull(1000, shape = 2)
xx = seq(-0.1, 4, 0.01)
y = dweibull(xx, shape = 2)

# Continuity constraint
fit = fweibullgpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpdcon(xx, wshape, wscale, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fweibullgpd(x, phiu = FALSE)
with(fit2, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density","No continuity constraint","With continuty constraint"),
  col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fweibullgpdcon(x, useq = seq(0.5, 2, length = 20))
fitfix = fweibullgpdcon(x, useq = seq(0.5, 2, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpdcon(xx, wshape, wscale, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dweibullgpdcon(xx, wshape, wscale, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dweibullgpdcon(xx, wshape, wscale, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density","Default initial value (90% quantile)",
 "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
 col=c("black", "red", "purple", "darkgreen"), 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

| | |
|---|---|
| `x` | quantiles |
| `gshape` | gamma shape (positive) |
| `gscale` | gamma scale (positive) |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive 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 $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the gamma bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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,1/gscale)` and `pgpd(x,u,sigmau,xi)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, 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 positive. Though behaviour at zero depends on the shape ($\alpha$):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where $\beta$ is the scale parameter.

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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: fgammagpdcon, fgammagpd, fmgammagpd, fmgamma, gammagpdcon, mgammagpd

Other gammagpdcon: fgammagpdcon, fgammagpd, fmgammagpdcon, gammagpdcon, mgammagpdcon

Other mgammagpd: fgammagpd, fmgammagpdcon, fmgammagpd, fmgamma, mgammagpdcon, mgammagpd, mgamma

Other fgammagpd: fgammagpd

## Examples

```
## Not run:
set.seed(1)
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("bottomright", 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 Single 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 continuity at threshold. The parameters are the gamma shape gshape and scale gscale, threshold u 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` | quantiles |
| `gshape` | gamma shape (positive) |
| `gscale` | gamma scale (positive) |
| `u` | threshold |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining gamma distribution for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the gamma bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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,1/gscale)` and `pgpd(x,u,sigmau,xi)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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. `dgammma(x,gshape,gscale)` and `dgpd(x,u,sigmau,xi)`) respectively. 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 positive. Though behaviour at zero depends on the shape ($\alpha$):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where $\beta$ is the scale parameter.

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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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: fgammagpdcon, fgammagpd, fmgammagpd, fmgamma, gammagpd, mgammagpd

Other gammagpdcon: fgammagpdcon, fgammagpd, fmgammagpdcon, gammagpd, mgammagpdcon

Other mgammagpdcon: fgammagpdcon, fmgammagpdcon, fmgammagpd, fmgamma, mgammagpdcon, mgammagpd, mgamma

Other fgammagpdcon: fgammagpdcon

## Examples

```
## Not run:
set.seed(1)
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("bottomright", 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 Estimate and GPD Both 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 kernel density estimate for bulk distribution between thresholds and conditional GPD beyond thresholds. 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, bw = NULL,
  kernel = "gaussian", 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, bw = NULL,
  kernel = "gaussian", 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, bw = NULL,
  kernel = "gaussian", 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, bw = NULL,
  kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `ul` | lower tail threshold |
| `sigmaul` | lower tail GPD scale parameter (positive) |
| `xil` | lower tail GPD shape parameter |

| | |
|---|---|
| phiul | probability of being below lower threshold $[0, 1]$ or TRUE |
| ur | upper tail threshold |
| sigmaur | upper tail GPD scale parameter (positive) |
| xir | upper tail GPD shape parameter |
| phiur | probability of being above upper threshold $[0, 1]$ or TRUE |
| bw | bandwidth for kernel (as standard deviations of kernel) or NULL |
| kernel | kernel name (default = "gaussian") |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

## Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk between thresholds and GPD beyond thresholds.

The user can pre-specify phiul and phiur permitting a parameterised value for the tail fractions $\phi_u l$ and $\phi_u r$. Alternatively, when phiul=TRUE and phiur=TRUE the tail fractions are estimated as the tail fractions from the KDE bulk model.

The alternate bandwidth definitions are discussed in the kernels, with the lambda as the default. The bw specification is the same as used in the density function.

The possible kernels are also defined in kernels with the "gaussian" as the default choice.

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 $\phi_{ul}$ 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,bw)) 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 $\phi_{ul}$ and $\phi_{ur}$ 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 $\phi_{ur} = 1 - H(u_r)$.

If no bandwidth is provided `lambda=NULL` and `bw=NULL` then the normal reference rule is used, using the `bw.nrd0` function, which is consistent with the `density` function. At least two kernel centres must be provided as the variance needs to be estimated.

See `gpd` for details of GPD upper tail component and `dkden` for details of KDE bulk component.

## Value

`dgkg` gives the density, `pgkg` gives the cumulative distribution function, `qgkg` gives the quantile function and `rgkg` gives a random sample.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of the other extreme value mixture model functions the `gkg` 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 `rgkg` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters or kernel centres.

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/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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd, kden

Other gkg: fgkgcon, fgkg, fkdengpd, gkgcon, kdengpd, kden

Other gkgcon: fgkgcon, fgkg, fkdengpdcon, gkgcon, kdengpdcon

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, kdengpd, kden

Other fgkg: fgkg

## Examples

```
## Not run:
set.seed(1)
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)

# asymmetric tail behaviours
x = rgkg(1000, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1)
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 = 0.1, xir = 0.3, phiur = 0.1))

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)
```

gkgcon *Kernel Density Estimate and GPD Both Upper and Lower Tails Extreme Value Mixture Model With Single Continuity Constraint at Both*

## Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with kernel density estimate for bulk distribution between thresholds and conditional GPD beyond thresholds and continuity at both of them. The parameters are the kernel bandwidth `lambda`, lower tail (threshold `ul`, GPD shape `xil` and tail fraction `phiul`) and upper tail (threshold `ur`, GPD shape `xiR` and tail fraction `phiur`).

## Usage

```
dgkgcon(x, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)), xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian", log = FALSE)

pgkgcon(q, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)), xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)

qgkgcon(p, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)), xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)

rgkgcon(n = 1, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)), xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `ul` | lower tail threshold |
| `xil` | lower tail GPD shape parameter |
| `phiul` | probability of being below lower threshold $[0, 1]$ or `TRUE` |
| `ur` | upper tail threshold |
| `xir` | upper tail GPD shape parameter |
| `phiur` | probability of being above upper threshold $[0, 1]$ or `TRUE` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `log` | logical, if TRUE then log density |

| | |
|---|---|
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

## Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk between thresholds and GPD beyond thresholds and continuity at both of them.

The user can pre-specify `phiul` and `phiur` permitting a parameterised value for the tail fractions $\phi_u l$ and $\phi_u r$. Alternatively, when `phiul=TRUE` and `phiur=TRUE` the tail fractions are estimated as the tail fractions from the KDE bulk model.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

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 $\phi_{ul}$ 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,bw))` 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 $u_l \le x \le u_r$ 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 $\phi_{ul}$ and $\phi_{ur}$ 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 $u_l \le x \le u_r$ 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 `ur` means that:

$$\phi_{ur}g_r(x) = (1 - \phi_{ul} - \phi_{ur})h(u_r)/(H(u_r) - H(u_l)).$$

By rearrangement, the GPD scale parameter `sigmaur` is then:

$$\sigma_u r = \phi_{ur}(H(u_r) - H(u_l))/h(u_r)(1 - \phi_{ul} - \phi_{ur}).$$

where $h(x)$, $g_l(x)$ and $g_r(x)$ are the KDE and conditional GPD density functions for lower and upper tail respectively. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u r = [1 - H(u_r)]/h(u_r)$$

.

The continuity constraint at 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

$$\sigma_u l = H(u_l)/h(u_l)$$

.

If no bandwidth is provided lambda=NULL and bw=NULL then the normal reference rule is used, using the bw.nrd0 function, which is consistent with the density function. At least two kernel centres must be provided as the variance needs to be estimated.

See gpd for details of GPD upper tail component and dkden for details of KDE bulk component.

## Value

dgkgcon gives the density, pgkgcon gives the cumulative distribution function, qgkgcon gives the quantile function and rgkgcon gives a random sample.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of the other extreme value mixture model functions the gkgcon 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 rgkgcon is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters or kernel centres.

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/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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fgkgcon, fkdengpdcon, fkdengpd, kdengpdcon, kdengpd

Other gkg: fgkgcon, fgkg, fkdengpd, gkg, kdengpd, kden

Other gkgcon: fgkgcon, fgkg, fkdengpdcon, gkg, kdengpdcon

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpdcon, kdengpdcon

Other fgkgcon: fgkgcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

kerncentres=rnorm(1000,0,1)
x = rgkgcon(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, dgkgcon(xx, kerncentres, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgkgcon(xx, kerncentres), type = "l")
lines(xx, pgkgcon(xx, kerncentres,xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgkgcon(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)

# asymmetric tail behaviours
x = rgkgcon(1000, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkgcon(xx, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1))

plot(xx, dgkgcon(xx, kerncentres, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2),
```

```
  type = "l", ylim = c(0, 0.4))
lines(xx, dgkgcon(xx, kerncentres, xil = -0.3, phiul = 0.3, xir = 0.3, phiur = 0.3),
  col = "red")
lines(xx, dgkgcon(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

| | |
|---|---|
| `x` | quantiles |
| `nmean` | normal mean |
| `nsd` | normal standard deviation (positive) |
| `ul` | lower tail threshold |
| `sigmaul` | lower tail GPD scale parameter (positive) |
| `xil` | lower tail GPD shape parameter |
| `phiul` | probability of being below lower threshold $[0, 1]$ or `TRUE` |

| | |
|---|---|
| `ur` | upper tail threshold |
| `sigmaur` | upper tail GPD scale parameter (positive) |
| `xir` | upper tail GPD shape parameter |
| `phiur` | probability of being above upper threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive 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 $\phi_{ul}$ 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 $u_l \le x \le u_r$ 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 $\phi_{ul}$ and $\phi_{ur}$ 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 $u_l \le x \le u_r$ 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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

gpd and dnorm

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpdcon, fnormgpd, gngcon, hpdcon, hpd, normgpdcon, normgpd

Other gng: fgngcon, fgng, fitmgng, fnormgpd, gngcon, itmgng, normgpd

Other gngcon: fgngcon, fgng, fnormgpdcon, gngcon, normgpdcon

Other fgng: fgng

## Examples

```
## Not run:
set.seed(1)
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 Single Continuity Constraint at Thresholds*

---

## 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 at the lower and upper thresholds. 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 | quantiles |
| nmean | normal mean |
| nsd | normal standard deviation (positive) |
| ul | lower tail threshold |
| xil | lower tail GPD shape parameter |
| phiul | probability of being below lower threshold $[0, 1]$ or TRUE |
| ur | upper tail threshold |
| xir | upper tail GPD shape parameter |
| phiur | probability of being above upper threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive 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 $\phi_{ul}$ 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 $u_l \le x \le u_r$ 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 $\phi_{ul}$ and $\phi_{ur}$ 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 $u_l \le x \le u_r$ 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 `ur` means that:

$$\phi_{ur} g_r(x) = (1 - \phi_{ul} - \phi_{ur}) h(u_r)/(H(u_r) - H(u_l)).$$

By rearrangement, the GPD scale parameter `sigmaur` is then:

$$\sigma_u r = \phi_{ur}(H(u_r) - H(u_l))/h(u_r)(1 - \phi_{ul} - \phi_{ur}).$$

where $h(x)$, $g_l(x)$ and $g_r(x)$ are the normal and conditional GPD density functions for lower and upper tail respectively. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u r = [1 - H(u_r)]/h(u_r)$$

.

The continuity constraint at `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

$$\sigma_u l = H(u_l)/h(u_l)$$

.

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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

gpd and dnorm

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpdcon, fnormgpd, gng, hpdcon, hpd, normgpdcon, normgpd

Other gng: fgngcon, fgng, fitmgng, fnormgpd, gng, itmgng, normgpd

Other gngcon: fgngcon, fgng, fnormgpdcon, gng, normgpdcon

Other fgngcon: fgngcon

## Examples

```
## Not run:
set.seed(1)
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 generalised Pareto distribution, either as a conditional on being above the threshold u or unconditional.

## 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` | quantiles |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

The GPD with parameters scale $\sigma_u$ and shape $\xi$ has conditional density of being above the threshold u given by

$$f(x|X > u) = 1/\sigma_u[1 + \xi(x - u)/\sigma_u]^{-1/\xi - 1}$$

for non-zero $\xi$, $x > u$ 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 $\xi = 0$ considered in the limit $\xi \rightarrow 0$, which is treated here 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 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.

## Acknowledgments

Based on the gpd functions in the evd package for which their author's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

## 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 essentially assumes the user provide excesses above u by default, rather than exceedances. The default sample size for rgpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

Coles, S.G. (2001). An Introduction to Statistical Modelling of Extreme Values. Springer Series in Statistics. Springer-Verlag: London.

## See Also

evd package and fpot

Other gpd: fgpd

Other fgpd: fgpd

## Examples

```
set.seed(1)
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)
```

hillplot *Hill Plot*

## Description

Plots the Hill plot and some its variants.

## Usage

```
hillplot(data, orderlim = NULL, tlim = NULL, hill.type = "Hill",
  r = 2, x.theta = FALSE, y.alpha = FALSE, alpha = 0.05,
  ylim = NULL, legend.loc = "topright",
  try.thresh = quantile(data[data > 0], 0.9, na.rm = TRUE),
  main = paste(ifelse(x.theta, "Alt", ""), hill.type, " Plot", sep = ""),
  xlab = ifelse(x.theta, "theta", "order"),
  ylab = paste(ifelse(x.theta, "Alt", ""), hill.type, ifelse(y.alpha,
  " alpha", " xi"), ">0", sep = ""), ...)
```

## Arguments

| | |
|---|---|
| `data` | vector of sample data |
| `orderlim` | vector of (lower, upper) limits of order statistics to plot estimator, or `NULL` to use default values |
| `tlim` | vector of (lower, upper) limits of range of threshold to plot estimator, or `NULL` to use default values |
| `hill.type` | "Hill" or "SmooHill" |
| `r` | smoothing factor for "SmooHill" (integer > 1) |
| `x.theta` | logical, should order (`FALSE`) or theta (`TRUE`) be given on x-axis |
| `y.alpha` | logical, should shape xi (`FALSE`) or tail index alpha (`TRUE`) be given on y-axis |
| `alpha` | significance level over range (0, 1), or `NULL` for no CI |
| `ylim` | y-axis limits or `NULL` |
| `legend.loc` | location of legend (see `legend`) or `NULL` for no legend |
| `try.thresh` | vector of thresholds to consider |
| `main` | title of plot |
| `xlab` | x-axis label |
| `ylab` | y-axis label |
| `...` | further arguments to be passed to the plotting functions |

## Details

Produces the Hill, AltHill, SmooHill and AltSmooHill plots, including confidence intervals.

For an ordered iid sequence $X_{(1)} \geq X_{(2)} \geq \cdots \geq X_{(n)} > 0$ the Hill (1975) estimator using $k$ order statistics is given by

$$H_{k,n} = \frac{1}{k} \sum_{i=1}^{k} \log(\frac{X_{(i)}}{X_{(k+1)}})$$

which is the pseudo-likelihood estimator of reciprocal of the tail index $\xi = /\alpha > 0$ for regularly varying tails (e.g. Pareto distribution). The Hill estimator is defined on orders $k > 2$, as when$k = 1$ the

$$H_{1,n} = 0$$

. The function will calculate the Hill estimator for $k \geq 1$. The simple Hill plot is shown for `hill.type="Hill"`.

Once a sufficiently low order statistic is reached the Hill estimator will be constant, upto sample uncertainty, for regularly varying tails. The Hill plot is a plot of

$$H_{k,n}$$

against the $k$. Symmetric asymptotic normal confidence intervals assuming Pareto tails are provided.

These so called Hill's horror plots can be difficult to interpret. A smooth form of the Hill estimator was suggested by Resnick and Starica (1997):

$$smooH_{k,n} = \frac{1}{(r-1)k} \sum_{j=k+1}^{rk} H_{j,n}$$

giving the smooHill plot which is shown for `hill.type="SmooHill"`. The smoothing factor is `r=2` by default.

It has also been suggested to plot the order on a log scale, by plotting the points $(\theta, H_{\lceil n^\theta \rceil, n})$ for $0 \leq \theta \leq 1$. This gives the so called AltHill and AltSmooHill plots. The alternative x-axis scale is chosen by `x.theta=TRUE`.

The Hill estimator is for the GPD shape $\xi > 0$, or the reciprocal of the tail index $\alpha = 1/\xi > 0$. The shape is plotted by default using `y.alpha=FALSE` and the tail index is plotted when `y.alpha=TRUE`.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The estimated parameter ($\xi$ or $\alpha$) at each threshold are plot by a horizontal solid line for all higher thresholds. The threshold should be set as low as possible, so a dashed line is shown below the pre-chosen threshold. If the Hill estimator is similar to the dashed line then a lower threshold may be chosen.

If no order statistic (or threshold) limits are provided `orderlim = tlim = NULL` then the lowest order statistic is set to $X_{(3)}$ and highest possible value $X_{(n-1)}$. However, the Hill estimator is always output for all $k = 1, \ldots, n-1$ and $k = 1, \ldots, floor(n/k)$ for smooHill estimator.

The missing (`NA` and `NaN`) and non-finite values are ignored. Non-positive data are ignored.

The lower x-axis is the order $k$ or $\theta$, chosen by the option `x.theta=FALSE` and `x.theta=TRUE` respectively. The upper axis is for the corresponding threshold.

## Value

`hillplot` gives the Hill plot. It also returns a dataframe containing columns of the order statistics, order, Hill estimator, it's standard devation and $100(1-\alpha)\%$ confidence interval (when requested). When the SmooHill plot is selected, then the corresponding SmooHill estimates are appended.

## Acknowledgments

Thanks to Younes Mouatasim, Risk Dynamics, Brussels for reporting various bugs in these functions.

## Note

Warning: Hill plots are not location invariant.

Asymptotic Wald type CI's are estimated for non-NULL signficance level alpha for the shape parameter, assuming exactly Pareto tails. When plotting on the tail index scale, then a simple reciprocal transform of the CI is applied which may be sub-optimal.

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

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

Hill, B.M. (1975). A simple general approach to inference about the tail of a distribution. Annals of Statistics 13, 331-341.

Resnick, S. and Starica, C. (1997). Smoothing the Hill estimator. Advances in Applied Probability 29, 271-293.

Resnick, S. (1997). Discussion of the Danish Data of Large Fire Insurance Losses. Astin Bulletin 27, 139-151.

## See Also

hill

## Examples

```
## Not run:
# Reproduce graphs from Figure 2.4 of Resnick (1997)
data(danish, package="evir")
par(mfrow = c(2, 2))

# Hill plot
hillplot(danish, y.alpha=TRUE, ylim=c(1.1, 2))

# AltHill plot
hillplot(danish, y.alpha=TRUE, x.theta=TRUE, ylim=c(1.1, 2))

# AltSmooHill plot
hillplot(danish, hill.type="SmooHill", r=3, y.alpha=TRUE, x.theta=TRUE, ylim=c(1.35, 1.85))

# AltHill and AltSmooHill plot (no CI's or legend)
hillout = hillplot(danish, hill.type="SmooHill", r=3, y.alpha=TRUE,
 x.theta=TRUE, try.thresh = c(), alpha=NULL, ylim=c(1.1, 2), legend.loc=NULL, lty=2)
n = length(danish)
with(hillout[3:n,], lines(log(ks)/log(n), 1/H, type="s"))

## End(Not run)
```

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` | quantiles |
| `nmean` | normal mean |
| `nsd` | normal standard deviation (positive) |
| `xi` | shape parameter |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive 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 it has 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 cumulative distribution function defined upto the threshold $x \le u$, 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. 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 two continuity constraints leads to the threshold u and GPD scale sigmau being replaced by a function of the normal mean, standard deviation and GPD shape parameters. Determined from setting $h(u) = g(u)$ 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)). 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

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 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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 and dnorm.

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: fhpdcon, fhpd, hpdcon

Other hpdcon: fhpdcon, fhpd, hpdcon

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, itmnormgpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, normgpdcon, normgpd

Other fhpd: fhpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(-5, 20, 0.01)
f1 = dhpd(xx, nmean = 0, nsd = 1, xi = 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, nmean = 0, nsd = 1.5, xi = 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, but only continuity at threshold and not necessarily continuous in first derivative. The parameters are the normal mean nmean and standard deviation nsd 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` | quantiles |
| `nmean` | normal mean |
| `nsd` | normal standard deviation (positive) |
| `u` | threshold |
| `xi` | shape parameter |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail which is continuous at threshold and not necessarily continuous in first derivative.

But it has 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 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. 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 leads to the GPD scale `sigmau` being replaced by a function of the normal mean, standard deviation, threshold and GPD shape parameters. Determined from setting $h(u) = g(u)$ 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

`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 `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 `rhpdcon` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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` and `dnorm`.

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: `fhpdcon`, `fhpd`, `hpd`

Other hpdcon: `fhpdcon`, `fhpd`, `hpd`

Other normgpd: `fgng`, `fhpd`, `fitmnormgpd`, `flognormgpd`, `fnormgpdcon`, `fnormgpd`, `gngcon`, `gng`, `hpd`, `itmnormgpd`, `lognormgpdcon`, `lognormgpd`, `normgpdcon`, `normgpd`

Other normgpdcon: `fgngcon`, `fhpdcon`, `flognormgpdcon`, `fnormgpdcon`, `fnormgpd`, `gngcon`, `gng`, `hpd`, `normgpdcon`, `normgpd`

Other fhpdcon: `fhpdcon`

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(-5, 20, 0.01)
f1 = dhpdcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0.4)
plot(xx, f1, type = "l")
abline(v = 4)

# 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(10000, nmean = 0, nsd = 1.5, u = 1, 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 = 1, xi = 0.2), col = "blue")

plot(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0), type = "l")
lines(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0.2), col = "red")
lines(xx, dhpdcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = -0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "u = 1, 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, kernel = "gaussian")

pkdenx(x, kerncentres, lambda, kernel = "gaussian")

bckdenxsimple(x, kerncentres, lambda, kernel = "gaussian")

pbckdenxsimple(x, kerncentres, lambda, kernel = "gaussian")

bckdenxcutnorm(x, kerncentres, lambda, kernel = "gaussian")

pbckdenxcutnorm(x, kerncentres, lambda, kernel = "gaussian")

bckdenxrenorm(x, kerncentres, lambda, kernel = "gaussian")
```

```
pbckdenxrenorm(x, kerncentres, lambda, kernel = "gaussian")

bckdenxreflect(x, kerncentres, lambda, kernel = "gaussian")

pbckdenxreflect(x, kerncentres, lambda, kernel = "gaussian")

pxb(x, lambda)

bckdenxbeta1(x, kerncentres, lambda, xmax)

pbckdenxbeta1(x, kerncentres, lambda, xmax)

bckdenxbeta2(x, kerncentres, lambda, xmax)

pbckdenxbeta2(x, kerncentres, lambda, xmax)

bckdenxgamma1(x, kerncentres, lambda)

pbckdenxgamma1(x, kerncentres, lambda)

bckdenxgamma2(x, kerncentres, lambda)

pbckdenxgamma2(x, kerncentres, lambda)

bckdenxcopula(x, kerncentres, lambda, xmax)

pbckdenxcopula(x, kerncentres, lambda, xmax)

pbckdenxlog(x, kerncentres, lambda, offset, kernel = "gaussian")

pbckdenxnn(x, kerncentres, lambda, kernel = "gaussian", nn)

qmix(x, u, epsilon)

qmixprime(x, u, epsilon)

qgbgmix(x, ul, ur, epsilon)

qgbgmixprime(x, ul, ur, epsilon)

pscounts(x, beta, design.knots, degree)
```

## Arguments

| | |
|---|---|
| x | quantiles |
| kerncentres | kernel centres (typically sample data vector or scalar) |
| lambda | bandwidth for kernel (as half-width of kernel) or NULL |
| kernel | kernel name (default = "gaussian") |
| xmax | upper bound on support (copula and beta kernels only) or NULL |
| offset | offset added to kernel centres (logtrans only) or NULL |

| | |
|---|---|
| nn | non-negativity correction method (simple boundary correction only) |
| u | threshold |
| epsilon | interval half-width |
| ul | lower tail threshold |
| ur | upper tail threshold |
| beta | vector of B-spline coefficients (required) |
| design.knots | spline knots for splineDesign function |
| degree | degree of B-splines (0 is constant, 1 is linear, etc.) |

## 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.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Author(s)

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

## See Also

density, kden and bckden.

---

itmgng *Normal Bulk with GPD Upper and Lower Tails Interval Transition 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 and interval transition. The parameters are the normal mean nmean and standard deviation nsd, interval half-width espilon, 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

```
ditmgng(x, nmean = 0, nsd = 1, epsilon = nsd, ul = qnorm(0.1,
  nmean, nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, log = FALSE)

pitmgng(q, nmean = 0, nsd = 1, epsilon = nsd, ul = qnorm(0.1,
  nmean, nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, lower.tail = TRUE)
```

```
qitmgng(p, nmean = 0, nsd = 1, epsilon, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, lower.tail = TRUE)

ritmgng(n = 1, nmean = 0, nsd = 1, epsilon = sd, ul = qnorm(0.1,
  nmean, nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0)
```

## Arguments

| | |
|---|---|
| x | quantiles |
| nmean | normal mean |
| nsd | normal standard deviation (positive) |
| epsilon | interval half-width |
| ul | lower tail threshold |
| sigmaul | lower tail GPD scale parameter (positive) |
| xil | lower tail GPD shape parameter |
| ur | upper tail threshold |
| sigmaur | upper tail GPD scale parameter (positive) |
| xir | upper tail GPD shape parameter |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

## Details

The interval transition extreme value mixture model combines a normal distribution for the bulk between the lower and upper thresholds and GPD for upper and lower tails, with a smooth transition over the interval $(u - epsilon, u + epsilon)$ (where $u$ can be exchanged for the lower and upper thresholds). The mixing function warps the normal to map from $(u - epsilon, u)$ to $(u - epsilon, u + epsilon)$ and warps the GPD from $(u, u + epsilon)$ to $(u - epsilon, u + epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(G_l(q(x)) + H_t(r(x)) + G_u(p(x)))$$

where $H_t(x)$ is the truncated normal cdf, i.e. `pnorm(x,nmean,nsd)`. The conditional GPD for the upper tail has cdf $G_u(x)$, i.e. `pgpd(x,ur,sigmaur,xir)` and lower tail cdf $G_l(x)$ is for the negated support, i.e. `1 -pgpd(-x,-ul,sigmaul,xil)`. The truncated normal is not renormalised to be proper, so $H_t(x)$ contributes `pnorm(ur,nmean,nsd) -pnorm(ul,nmean,nsd)` to the cdf for all $x \geq (u_r + \epsilon)$ and zero below $x \leq (u_l - \epsilon)$. The normalisation constant $\kappa$ ensures a proper density, given by `1/(2 + pnorm(ur,nmean,nsd) -pnorm(ul,nmean,nsd)` where the 2 is from two GPD components and latter is contribution from normal component.

The mixing functions $q(x)$, $r(x)$ and $p(x)$ are reformulated from the $q_i(x)$ suggested by Holden and Haug (2013). These are symmetric about each threshold, which for convenience will be referred to a simply $u$. So for computational convenience only a single $q(x; u)$ has been implemented for the lower and upper GPD components called qmix for a given $u$, with the complementary mixing function then defined as $p(x; u) = -q(-x; -u)$. The bulk model mixing function $r(x)$ utilises the equivalent of the $q(x)$ for the lower threshold and $p(x)$ for the upper threshold, so these are reused in the bulk mixing function qgbgmix.

A minor adaptation of the mixing function has been applied following a similar approach to that explained in ditmnormgpd. For the bulk model mixing function $r(x)$, we need $r(x) <= ul$ for all $x \leq ul - epsilon$ and $r(x) >= ur$ for all $x \geq ur + epsilon$, as then the bulk model will contribute zero below the lower interval and the constant $H_t(ur) = H(ur) - H(ul)$ for all $x$ above the upper interval. Holden and Haug (2013) define $r(x) = x - \epsilon$ for all $x \geq ur$ and $r(x) = x + \epsilon$ for all $x \leq ul$. For more straightforward and interpretable computational implementation the mixing function has been set to the lower threshold $r(x) = u_l$ for all $x \leq u_l - \epsilon$ and to the upper threshold $r(x) = u_r$ for all $x \leq u_r + \epsilon$, so the cdf/pdf of the normal model can be used directly. We do not have to define cdf/pdf for the non-proper truncated normal seperately. As such $r'(x) = 0$ for all $x \leq u_l - \epsilon$ and $x \geq u_r + \epsilon$ in qmixxprime, which also makes it clearer that normal does not contribute to either tails beyond the intervals and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the normal and GPD components directly.

## Value

ditmgng gives the density, pitmgng gives the cumulative distribution function, qitmgng gives the quantile function and ritmgng 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 ritmgng 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 ritmgng is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

## Author(s)

Alfadino Akbar 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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

### See Also

gng, normgpd, gpd and dnorm

Other itmgng: fitmgng

Other gng: fgngcon, fgng, fitmgng, fnormgpd, gngcon, gng, normgpd

Other itmnormgpd: fitmgng, fitmnormgpd, itmnormgpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(-5, 5, 0.01)
ul = -1.5;ur = 2
epsilon = 0.8
kappa = 1/(2 + pnorm(ur, 0, 1) - pnorm(ul, 0, 1))

f = ditmgng(xx, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5, ur, sigmaur = 1, xir = 0.5)
plot(xx, f, ylim = c(0, 0.5), xlim = c(-5, 5), type = 'l', lwd = 2, xlab = "x", ylab = "density")
lines(xx, kappa * dgpd(-xx, -ul, sigmau = 1, xi = 0.5), col = "blue", lty = 2, lwd = 2)
lines(xx, kappa * dnorm(xx, 0, 1), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dgpd(xx, ur, sigmau = 1, xi = 0.5), col = "green", lty = 2, lwd = 2)
abline(v = ul + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "blue")
abline(v = ur + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "green")
legend('topright', c('Normal-GPD ITM', 'kappa*GPD Lower', 'kappa*Normal', 'kappa*GPD Upper'),
      col = c("black", "blue", "red", "green"), lty = c(1, 2, 2, 2), lwd = 2)

# cdf contributions
F = pitmgng(xx, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5, ur, sigmaur = 1, xir = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(-5, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx < ul], kappa * (1 - pgpd(-xx[xx < ul], -ul, 1, 0.5)), col = "blue", lty = 2, lwd = 2)
lines(xx[(xx >= ul) & (xx <= ur)], kappa * (1 + pnorm(xx[(xx >= ul) & (xx <= ur)], 0, 1) -
      pnorm(ul, 0, 1)), col = "red", lty = 2, lwd = 2)
lines(xx[xx > ur], kappa * (1 + (pnorm(ur, 0, 1) - pnorm(ul, 0, 1)) +
      pgpd(xx[xx > ur], ur, sigmau = 1, xi = 0.5)), col = "green", lty = 2, lwd = 2)
abline(v = ul + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "blue")
abline(v = ur + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "green")
legend('topleft', c('Normal-GPD ITM', 'kappa*GPD Lower', 'kappa*Normal', 'kappa*GPD Upper'),
      col = c("black", "blue", "red", "green"), lty = c(1, 2, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmgng(10000, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5,
                                                ur, sigmaur = 1, xir = 0.5)
hist(x, freq = FALSE, breaks = seq(-1000, 1000, 0.1), xlim = c(-5, 5))
lines(xx, ditmgng(xx, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5,
  ur, sigmaur = 1, xir = 0.5), lwd = 2, col = 'black')

## End(Not run)
```

---

`itmnormgpd` *Normal Bulk and GPD Tail Interval Transition Mixture Model*

---

## Description

Density, cumulative distribution function, quantile function and random number generation for the normal bulk and GPD tail interval transition mixture model. The parameters are the normal mean `nmean` and standard deviation `nsd`, threshold `u`, interval half-width `epsilon`, GPD scale `sigmau` and shape `xi`.

## Usage

```
ditmnormgpd(x, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9,
  nmean, nsd), sigmau = nsd, xi = 0, log = FALSE)

pitmnormgpd(q, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9,
  nmean, nsd), sigmau = nsd, xi = 0, lower.tail = TRUE)

qitmnormgpd(p, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9,
  nmean, nsd), sigmau = nsd, xi = 0, lower.tail = TRUE)

ritmnormgpd(n = 1, nmean = 0, nsd = 1, epsilon = nsd,
  u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `nmean` | normal mean |
| `nsd` | normal standard deviation (positive) |
| `epsilon` | interval half-width |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

The interval transition mixture model combines a normal for the bulk model with GPD for the tail model, with a smooth transition over the interval $(u - epsilon, u + epsilon)$. The mixing function warps the normal to map from $(u - epsilon, u)$ to $(u - epsilon, u + epsilon)$ and warps the GPD from $(u, u + epsilon)$ to $(u - epsilon, u + epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(H_t(q(x)) + G(p(x)))$$

where $H_t(x)$ and $G(x)$ are the truncated normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x,nmean,nsd)` and `pgpd(x,u,sigmau,xi)`) respectively. The truncated normal is not renormalised to be proper, so $H_t(x)$ contrubutes `pnorm(u,nmean,nsd)` to the cdf for all $x \geq (u+\epsilon)$. The normalisation constant $\kappa$ ensures a proper density, given by `1/(1+pnorm(u,nmean,nsd))` where 1 is from GPD component and latter is contribution from normal component.

The mixing functions $q(x)$ and $p(x)$ suggested by Holden and Haug (2013) have been implemented. These are symmetric about the threshold $u$. So for computational convenience only $q(x;u)$ has been implemented as qmix for a given $u$, with the complementary mixing function is then defined as $p(x;u) = -q(-x;-u)$.

A minor adaptation of the mixing function has been applied. For the mixture model to function correctly $q(x) >= u$ for all $x \geq u + \epsilon$, as then the bulk model will contribute the constant $H_t(u) = H(u)$ for all $x$ above the interval. Holden and Haug (2013) define $q(x) = x - \epsilon$ for all $x \geq u$. For more straightforward and interpretable computational implementation the mixing function has been set to the threshold $q(x) = u$ for all $x \geq u$, so the cdf/pdf of the normal model can be used directly. We do not have to define cdf/pdf for the non-proper truncated normal seperately. As such $q'(x) = 0$ for all $x \geq u$ in qmixxprime, which also makes it clearer that normal does not contribute to the tail above the interval and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the normal and GPD components directly.

## Value

ditmnormgpd gives the density, pitmnormgpd gives the cumulative distribution function, qitmnormgpd gives the quantile function and ritmnormgpd 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 ritmnormgpd 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 ritmnormgpd is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

## Author(s)

Alfadino Akbar 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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

### See Also

normgpd, gpd and dnorm

Other itmnormgpd: fitmgng, fitmnormgpd, itmgng

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, lognormgpdcon, lognormgpd, normgpdcon, normgpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(-4, 5, 0.01)
u = 1.5
epsilon = 0.4
kappa = 1/(1 + pnorm(u, 0, 1))

f = ditmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(-4, 5), type = 'l', lwd = 2, xlab = "x", ylab = "density")
lines(xx, kappa * dgpd(xx, u, sigmau = 1, xi = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dnorm(xx, 0, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Normal-GPD ITM', 'kappa*Normal', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# cdf contributions
F = pitmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(-4, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx > u], kappa * (pnorm(u, 0, 1) + pgpd(xx[xx > u], u, sigmau = 1, xi = 0.5)),
     col = "red", lty = 2, lwd = 2)
lines(xx[xx <= u], kappa * pnorm(xx[xx <= u], 0, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topleft', c('Normal-GPD ITM', 'kappa*Normal', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmnormgpd(10000, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
hist(x, freq = FALSE, breaks = seq(-4, 1000, 0.1), xlim = c(-4, 5))
lines(xx, ditmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5),
  lwd = 2, col = 'black')

## End(Not run)
```

---

itmweibullgpd *Weibull Bulk and GPD Tail Interval Transition Mixture Model*

---

### Description

Density, cumulative distribution function, quantile function and random number generation for the Weibull bulk and GPD tail interval transition mixture model. The parameters are the Weibull shape wshape and scale wscale, threshold u, interval half-width epsilon, GPD scale sigmau and shape xi.

### Usage

```
ditmweibullgpd(x, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  u = qweibull(0.9, wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 +
  2/wshape) - (wscale * gamma(1 + 1/wshape))^2), xi = 0, log = FALSE)

pitmweibullgpd(q, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  u = qweibull(0.9, wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 +
  2/wshape) - (wscale * gamma(1 + 1/wshape))^2), xi = 0,
  lower.tail = TRUE)

qitmweibullgpd(p, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2),
  u = qweibull(0.9, wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 +
  2/wshape) - (wscale * gamma(1 + 1/wshape))^2), xi = 0,
  lower.tail = TRUE)

ritmweibullgpd(n = 1, wshape = 1, wscale = 1,
  epsilon = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
  1/wshape))^2), u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
  1/wshape))^2), xi = 0)
```

### Arguments

| | |
|---|---|
| x | quantiles |
| wshape | Weibull shape (positive) |
| wscale | Weibull scale (positive) |
| epsilon | interval half-width |
| u | threshold |
| sigmau | scale parameter (positive) |
| xi | shape parameter |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

### Details

The interval transition mixture model combines a Weibull for the bulk model with GPD for the tail model, with a smooth transition over the interval $(u - epsilon, u + epsilon)$. The mixing function warps the Weibull to map from $(u - epsilon, u)$ to $(u - epsilon, u + epsilon)$ and warps the GPD from $(u, u + epsilon)$ to $(u - epsilon, u + epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(H_t(q(x)) + G(p(x)))$$

where $H_t(x)$ and $G(X)$ are the truncated Weibull and conditional GPD cumulative distribution functions (i.e. pweibull(x,wshape,wscale) and pgpd(x,u,sigmau,xi)) respectively. The truncated Weibull is not renormalised to be proper, so $H_t(x)$ contrubutes pweibull(u,wshape,wscale) to the cdf for all $x \geq (u + \epsilon)$. The normalisation constant $\kappa$ ensures a proper density, given by 1/(1+pweibull(u,wshape,wscale)) where 1 is from GPD component and latter is contribution from Weibull component.

The mixing functions $q(x)$ and $p(x)$ suggested by Holden and Haug (2013) have been implemented. These are symmetric about the threshold $u$. So for computational convenience only $q(x; u)$ has been implemented as qmix for a given $u$, with the complementary mixing function is then defined as $p(x; u) = -q(-x; -u)$.

A minor adaptation of the mixing function has been applied. For the mixture model to function correctly $q(x) >= u$ for all $x \geq u + \epsilon$, as then the bulk model will contribute the constant $H_t(u) = H(u)$ for all $x$ above the interval. Holden and Haug (2013) define $q(x) = x - \epsilon$ for all $x \geq u$. For more straightforward and interpretable computational implementation the mixing function has been set to the threshold $q(x) = u$ for all $x \geq u$, so the cdf/pdf of the Weibull model can be used directly. We do not have to define cdf/pdf for the non-proper truncated Weibull seperately. As such $q'(x) = 0$ for all $x \geq u$ in qmixxprime, which also it makes clearer that Weibull does not contribute to the tail above the interval and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the Weibull and GPD components directly.

## Value

ditmweibullgpd gives the density, pitmweibullgpd gives the cumulative distribution function, qitmweibullgpd gives the quantile function and ritmweibullgpd 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 ritmweibullgpd 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 ritmweibullgpd is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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

## Author(s)

Alfadino Akbar 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

Holden, L. and Haug, O. (2013). A mixture model for unsupervised tail estimation. arxiv:0902.4137

## See Also

weibullgpd, gpd and dweibull

Other itmweibullgpd: fitmweibullgpd, fweibullgpdcon, fweibullgpd, weibullgpdcon, weibullgpd

Other weibullgpd: fitmweibullgpd, fweibullgpdcon, fweibullgpd, weibullgpdcon, weibullgpd

Other weibullgpdcon: fweibullgpdcon, fweibullgpd, weibullgpdcon, weibullgpd

Other fitmweibullgpd: fitmweibullgpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(0.001, 5, 0.01)
u = 1.5
epsilon = 0.4
kappa = 1/(1 + pweibull(u, 2, 1))

f = ditmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2, xlab = "x", ylab = "density")
lines(xx, kappa * dgpd(xx, u, sigmau = 1, xi = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dweibull(xx, 2, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Weibull-GPD ITM', 'kappa*Weibull', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# cdf contributions
F = pitmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx > u], kappa * (pweibull(u, 2, 1) + pgpd(xx[xx > u], u, sigmau = 1, xi = 0.5)),
     col = "red", lty = 2, lwd = 2)
lines(xx[xx <= u], kappa * pweibull(xx[xx <= u], 2, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Weibull-GPD ITM', 'kappa*Weibull', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmweibullgpd(10000, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
hist(x, freq = FALSE, breaks = seq(0, 1000, 0.1), xlim = c(0, 5))
lines(xx, ditmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5),
  lwd = 2, col = 'black')

## End(Not run)
```

---

kden *Kernel Density Estimation, With Variety of Kernels*

---

### Description

Density, cumulative distribution function, quantile function and random number generation for the kernel density estimation using the kernel specified by `kernel`, with a constant bandwidth specified by either `lambda` or `bw`.

### Usage

```
dkden(x, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
  log = FALSE)

pkden(q, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
  lower.tail = TRUE)

qkden(p, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
  lower.tail = TRUE)

rkden(n = 1, kerncentres, lambda = NULL, bw = NULL,
  kernel = "gaussian")
```

### Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

### Details

Kernel density estimation using one of many possible kernels with a constant bandwidth.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` help documentation with the `"gaussian"` as the default choice.

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 (using the FFT), 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 or large number of evaluation points.

However, this KDE function applies the less efficient convolution using the standard 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 kernel. Thus are no restriction on the values x can take. For example, in the "gaussian" kernel case for a particular x the density is evaluated as mean(dnorm(x,kerncentres,lambda)) for the density and mean(pnorm(x,kerncentres,lambda)) for cumulative distribution function which is slower than the FFT but is more adaptable.

An inversion sampler is used for random number generation which also rather inefficient, as it can be carried out more efficiently using a mixture representation.

The quantile function is rather complicated as there is no closed form solution, so is obtained by numerical approximation of the inverse cumulative distribution function $P(X \leq q) = p$ to find $q$. The quantile function qkden evaluates the KDE cumulative distribution function over the range from c(max(kerncentre) -lambda,max(kerncentre) + lambda), or c(max(kerncentre) -5*lambda,max(kerncentre) + 5*lambda) for normal kernel. Outside of this 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 (with minimum of 1000) 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.

If no bandwidth is provided lambda=NULL and bw=NULL then the normal reference rule is used, using the bw.nrd0 function, 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.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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/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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kdengpd

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, gkg, kdengpdcon, kdengpd

Other gkg: fgkgcon, fgkg, fkdengpd, gkgcon, gkg, kdengpd

Other bckden: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, gkg, kdengpd

Other fkden: fkden

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

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
x = rnorm(nk)
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 5), ylim = c(0, 0.6))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$bw)*0.05)
lines(xx,dnorm(xx), col = "black")
```

```
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
lines(density(x, bw = fit$bw), 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)
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(10000, x, lambda = fit$lambda)
hist(xnew, breaks = 100, 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 Estimate 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 kernel density estimate for 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, bw = NULL,
  kernel = "gaussian", log = FALSE)

pkdengpd(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), sigmau = sqrt(6 *
  var(kerncentres))/pi, xi = 0, phiu = TRUE, bw = NULL,
  kernel = "gaussian", 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, bw = NULL,
  kernel = "gaussian", 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, bw = NULL,
  kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default = "gaussian"`) |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining kernel density estimate (KDE) 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 $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the KDE bulk model.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

The cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the kernel density estimate (`phiu=TRUE`), upto the threshold $x \le u$, 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 KDE and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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 $\phi_u = 1 - H(u)$.

If no bandwidth is provided `lambda=NULL` and `bw=NULL` then the normal reference rule is used, using the `bw.nrd0` function, which is consistent with the `density` function. At least two kernel centres must be provided as the variance needs to be estimated.

See `gpd` for details of GPD upper tail component and `dkden` for details of KDE bulk component.

## Value

`dkdengpd` gives the density, `pkdengpd` gives the cumulative distribution function, `qkdengpd` gives the quantile function and `rkdengpd` gives a random sample.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of 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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters or kernel centres.

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/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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpdcon, kden

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, gkg, kdengpdcon, kden

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fgkgcon, fkdengpdcon, fkdengpd, gkgcon, kdengpdcon

Other gkg: fgkgcon, fgkg, fkdengpd, gkgcon, gkg, kden

Other bckdengpd: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpd, gkg, kden

Other fkdengpd: fkdengpd

## Examples

```
## Not run:
set.seed(1)
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)

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, u = 1.2, sigmau = 0.56, xi = 0.1))

plot(xx, dkdengpd(xx, kerncentres, xi=0, phiu = 0.1), type = "l")
lines(xx, dkdengpd(xx, kerncentres, xi=0.2, phiu = 0.1), col = "red")
lines(xx, dkdengpd(xx, kerncentres, 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)
```

kdengpdcon *Kernel Density Estimate and GPD Tail Extreme Value Mixture Model With Single Continuity Constraint*

## Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the bandwidth `lambda`, threshold `u` GPD shape `xi` and tail fraction `phiu`.

## Usage

```
dkdengpdcon(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", log = FALSE)

pkdengpdcon(q, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)

qkdengpdcon(p, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)

rkdengpdcon(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `u` | threshold |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kernel` | kernel name (`default` = `"gaussian"`) |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the KDE bulk model.

The alternate bandwidth definitions are discussed in the `kernels`, with the `lambda` as the default. The `bw` specification is the same as used in the `density` function.

The possible kernels are also defined in `kernels` with the `"gaussian"` as the default choice.

The cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the kernel density estimate (`phiu=TRUE`), upto the threshold $x \le u$, 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 KDE and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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 KDE and conditional GPD density functions respectively. 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

$$\sigma_u = [1 - H(u)]/h(u)$$

.

If no bandwidth is provided `lambda=NULL` and `bw=NULL` then the normal reference rule is used, using the `bw.nrd0` function, which is consistent with the `density` function. At least two kernel centres must be provided as the variance needs to be estimated.

See `gpd` for details of GPD upper tail component and `dkden` for details of KDE bulk component.

## Value

`dkdengpdcon` gives the density, `pkdengpdcon` gives the cumulative distribution function, `qkdengpdcon` gives the quantile function and `rkdengpdcon` gives a random sample.

## Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

## Note

Unlike most of 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters or kernel centres.

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/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.

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

kernels, kfun, density, bw.nrd0 and dkde in ks package.

Other kden: bckden, fbckden, fgkgcon, fgkg, fkdengpdcon, fkdengpd, fkden, kdengpd, kden

Other kdengpd: bckdengpd, fbckdengpd, fgkg, fkdengpdcon, fkdengpd, fkden, gkg, kdengpd, kden

Other kdengpdcon: bckdengpdcon, fbckdengpdcon, fgkgcon, fkdengpdcon, fkdengpd, gkgcon, kdengpd

Other gkgcon: fgkgcon, fgkg, fkdengpdcon, gkgcon, gkg

Other bckdengpdcon: bckdengpdcon, bckdengpd, bckden, fbckdengpdcon, fbckdengpd, fbckden, fkdengpdcon, gkgcon

Other fkdengpdcon: fkdengpdcon

### Examples

```
## Not run:
set.seed(1)
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)

x = rkdengpdcon(1000, kerncentres, phiu = 0.2, u = 1, xi = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dkdengpdcon(xx, kerncentres, phiu = 0.2, u = 1, xi = -0.1))

plot(xx, dkdengpdcon(xx, kerncentres, xi=0, u = 1, phiu = 0.2), type = "l")
lines(xx, dkdengpdcon(xx, kerncentres, xi=0.2, u = 1, phiu = 0.2), col = "red")
lines(xx, dkdengpdcon(xx, kerncentres, xi=-0.2, u = 1, 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)
```

---

kernels *Kernel functions*

---

### Description

Functions for commonly used kernels for kernel density estimation. The density and cumulative distribution functions are provided.

### Usage

```
kdgaussian(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kduniform(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdtriangular(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdepanechnikov(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdbiweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdtriweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdtricube(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
```

```
kdparzen(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdoptcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpgaussian(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpuniform(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kptriangular(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpepanechnikov(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpbiweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kptriweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kptricube(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpparzen(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpoptcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdz(z, kernel = "gaussian")

kpz(z, kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| x | location to evaluate KDE (single scalar or vector) |
| lambda | bandwidth for kernel (as half-width of kernel) or NULL |
| bw | bandwidth for kernel (as standard deviations of kernel) or NULL |
| kerncentres | kernel centres (typically sample data vector or scalar) |
| z | standardised location put into kernel z = (x-kerncentres)/lambda |
| kernel | kernel name (default = "gaussian") |

## Details

Functions for the commonly used kernels for kernel density estimation. The density and cumulative distribution functions are provided. Each function can accept the bandwidth specified as either:

1. bw - in terms of number of standard deviations of the kernel, consistent with the defined values in the density function in the R base libraries
2. lambda - in terms of half-width of kernel

If both bandwidths are given as NULL then the default bandwidth is lambda=1. If either one is specified then this will be used. If both are specified then lambda will be used.

All the kernels have bounded support $[-\lambda, \lambda]$, except the normal ("gaussian") which is unbounded. In the latter, both bandwidths are the same bw=lambda and equal to the standard deviation.

Typically,a single location x at which to evaluate kernel is given along with vector of kernel centres. As such, they are designed to be used with sapply to loop over vector of locations at which to evaluate KDE. Alternatively, a vector of locations x can be given with a single scalar kernel centre kerncentres, which is commonly used when locations are pre-standardised by (x-kerncentres)/lambda and kerncentre=0. A warnings is given if both the evaluation locations and kernel centres are vectors as this is not often needed so is likely to be a user error.

If no kernel centres are provided then by default it is set to zero (i.e. x is at middle of kernel).

The following kernels are implemented, with relevant ones having definitions consistent with those of the density function, except where specified:

- gaussian or normal
- uniform or rectangular - same as "rectangular" in density function
- triangular
- epanechnikov
- biweight
- triweight
- tricube
- parzen
- cosine
- optcosine

The kernel densities are all normalised to unity. See Wikipedia reference below for their definitions.

Each kernel's functions can be called individually, or the global functions kdz and kpz for the density and cumulative distribution function can apply any particular kernel which is specified by the kernel input. These global functions take the standardised locations z = (x -kerncentres)/lambda.

## Value

codekd* and kp* give the density and cumulative distribution functions for each kernel respectively, where * is the kernel name. kdz and kpz are the equivalent global functions for all of the kernels.

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

## References

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

http://en.wikipedia.org/wiki/Kernel_(statistics)

Wand, M. and Jones, M.C. (1995). Kernel Smoothing. Chapman && Hall.

## See Also

density, kden and bckden.

Other kernels: kfun

## Examples

```
xx = seq(-2, 2, 0.01)
plot(xx, kdgaussian(xx), type = "l", col = "black",ylim = c(0, 1.2))
lines(xx, kduniform(xx), col = "grey")
lines(xx, kdtriangular(xx), col = "blue")
lines(xx, kdepanechnikov(xx), col = "darkgreen")
lines(xx, kdbiweight(xx), col = "red")
lines(xx, kdtriweight(xx), col = "purple")
lines(xx, kdtricube(xx), col = "orange")
lines(xx, kdparzen(xx), col = "salmon")
lines(xx, kdcosine(xx), col = "cyan")
lines(xx, kdoptcosine(xx), col = "goldenrod")
legend("topright", c("Gaussian", "uniform", "triangular", "Epanechnikov",
"biweight", "triweight", "tricube", "Parzen", "cosine", "optcosine"), lty = 1,
col = c("black", "grey", "blue", "darkgreen", "red", "purple", "orange",
  "salmon", "cyan", "goldenrod"))
```

---

kfun *Various subsidiary kernel function, conversion of bandwidths and evaluating certain kernel integrals.*

---

## Description

Functions for checking the inputs to the kernel functions, evaluating integrals $\int u^l K*(u)du$ for $l = 0, 1, 2$ and conversion between the two bandwidth definitions.

## Usage

```
check.kinputs(x, lambda, bw, kerncentres, allownull = FALSE)

check.kernel(kernel)

check.kbw(lambda, bw, allownull = FALSE)

klambda(bw = NULL, kernel = "gaussian", lambda = NULL)

kbw(lambda = NULL, kernel = "gaussian", bw = NULL)

ka0(truncpoint, kernel = "gaussian")

ka1(truncpoint, kernel = "gaussian")

ka2(truncpoint, kernel = "gaussian")
```

## Arguments

| | |
|---|---|
| `x` | location to evaluate KDE (single scalar or vector) |
| `lambda` | bandwidth for kernel (as half-width of kernel) or `NULL` |
| `bw` | bandwidth for kernel (as standard deviations of kernel) or `NULL` |
| `kerncentres` | kernel centres (typically sample data vector or scalar) |

| | |
|---|---|
| allownull | logical, where TRUE permits NULL values |
| kernel | kernel name (default = "gaussian") |
| truncpoint | upper endpoint as standardised location x/lambda |

## Details

Various boundary correction methods require integral of (partial moments of) kernel within the range of support, over the range $[-1, p]$ where $p$ is the truncpoint determined by the standardised distance of location $x$ where KDE is being evaluated to the lower bound of zero, i.e. truncpoint = x/lambda. The exception is the normal kernel which has unbounded support so the $[-5 * \lambda, p]$ where lambda is the standard deviation bandwidth. There is a function for each partial moment of degree (0, 1, 2):

- ka0 - $\int_{-1}^{p} K * (z) dz$
- ka1 - $\int_{-1}^{p} uK * (z) dz$
- ka2 - $\int_{-1}^{p} u^2 K * (z) dz$

Notice that when evaluated at the upper endpoint on the support $p = 1$ (or $p = \infty$ for normal) these are the zeroth, first and second moments. In the normal distribution case the lower bound on the region of integration is $\infty$ but implemented here as $-5 * \lambda$. These integrals are all specified in closed form, there is no need for numerical integration (except normal which uses the pnorm function).

See kpu for list of kernels and discussion of bandwidth definitions (and their default values):

1. bw - in terms of number of standard deviations of the kernel, consistent with the defined values in the density function in the R base libraries
2. lambda - in terms of half-width of kernel

The klambda function converts the bw to the lambda equivalent, and kbw applies converse. These conversions are kernel specific as they depend on the kernel standard deviations. If both bw and lambda are provided then the latter is used by default. If neither are provided (bw=NULL and lambda=NULL) then default is lambda=1.

check.kinputs checks all the kernel function inputs, check.klambda checks the pair of inputted bandwidths and check.kernel checks the kernel names.

## Value

klambda and kbw return the lambda and bw bandwidths respectively.

The checking functions check.kinputs, check.klambda and check.kernel will stop on errors and return no value.

ka0, ka1 and ka2 return the partial moment integrals specified above.

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

## References

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

http://en.wikipedia.org/wiki/Kernel_(statistics)

Wand and Jones (1995). Kernel Smoothing. Chapman & Hall.

## See Also

kernels, density, kden and bckden.

Other kernels: kernels

## Examples

```
xx = seq(-2, 2, 0.01)
plot(xx, kdgaussian(xx), type = "l", col = "black",ylim = c(0, 1.2))
lines(xx, kduniform(xx), col = "grey")
lines(xx, kdtriangular(xx), col = "blue")
lines(xx, kdepanechnikov(xx), col = "darkgreen")
lines(xx, kdbiweight(xx), col = "red")
lines(xx, kdtriweight(xx), col = "purple")
lines(xx, kdtricube(xx), col = "orange")
lines(xx, kdparzen(xx), col = "salmon")
lines(xx, kdcosine(xx), col = "cyan")
lines(xx, kdoptcosine(xx), col = "goldenrod")
legend("topright", c("Gaussian", "uniform", "triangular", "Epanechnikov",
"biweight", "triweight", "tricube", "Parzen", "cosine", "optcosine"), lty = 1,
col = c("black", "grey", "blue", "darkgreen", "red", "purple",
  "salmon", "orange", "cyan", "goldenrod"))
```

---

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 log-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

| | |
|---|---|
| x | quantiles |
| lnmean | mean on log scale |

| | |
|---|---|
| `lnsd` | standard deviation on log scale (positive) |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive 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 $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the log-normal bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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,lnmean,lnsd)` and `pgpd(x,u,sigmau,xi)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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 log-normal is defined on the positive reals, so the threshold must be positive.

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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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: `flognormgpdcon`, `flognormgpd`, `lognormgpdcon`

Other lognormgpdcon: `flognormgpdcon`, `flognormgpd`, `lognormgpdcon`

Other normgpd: `fgng`, `fhpd`, `fitmnormgpd`, `flognormgpd`, `fnormgpdcon`, `fnormgpd`, `gngcon`, `gng`, `hpdcon`, `hpd`, `itmnormgpd`, `lognormgpdcon`, `normgpdcon`, `normgpd`

Other flognormgpd: `flognormgpd`

## Examples

```
## Not run:
set.seed(1)
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 Single 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 continuity at threshold. The parameters are the log-normal mean lnmean and standard deviation lnsd, threshold u GPD 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 | quantiles |
| lnmean | mean on log scale |
| lnsd | standard deviation on log scale (positive) |
| u | threshold |
| xi | shape parameter |
| phiu | probability of being above threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |

| | |
|---|---|
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

**Details**

Extreme value mixture model combining log-normal distribution for the bulk below the threshold and GPD for upper tailwith continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the log-normal bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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,lnmean,lnsd)` and `pgpd(x,u,sigmau,xi)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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 log-normal is defined on the positive reals, so the threshold must be positive.

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,lnmean,lnsd)` and `dgpd(x,u,sigmau,xi)`) respectively. 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 `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 rlognormgpdcon 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: flognormgpdcon, flognormgpd, lognormgpd

Other lognormgpdcon: flognormgpdcon, flognormgpd, lognormgpd

Other normgpd: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpd, normgpdcon, normgpd

Other flognormgpdcon: flognormgpdcon

## Examples

```
## Not run:
set.seed(1)
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)
```

---

mgamma *Mixture of Gammas Distribution*

---

## Description

Density, cumulative distribution function, quantile function and random number generation for the mixture of gammas distribution. The parameters are the multiple gamma shapes `mgshape` scales `mgscale` and weights `mgweights`.

## Usage

```
dmgamma(x, mgshape = 1, mgscale = 1, mgweight = NULL, log = FALSE)

pmgamma(q, mgshape = 1, mgscale = 1, mgweight = NULL,
  lower.tail = TRUE)

qmgamma(p, mgshape = 1, mgscale = 1, mgweight = NULL,
  lower.tail = TRUE)

rmgamma(n = 1, mgshape = 1, mgscale = 1, mgweight = NULL)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `mgshape` | mgamma shape (positive) as list or vector |
| `mgscale` | mgamma scale (positive) as list or vector |
| `mgweight` | mgamma weights (positive) as list or vector (`NULL` for equi-weighted) |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Distribution functions for weighted mixture of gammas.

Suppose there are $M >= 1$ gamma components in the mixture model. If you wish to have a single (scalar) value for each parameter within each of the $M$ components then these can be input as a vector of length $M$. If you wish to input a vector of values for each parameter within each of the $M$ components, then they are input as a list with each entry the parameter object for each component (which can either be a scalar or vector as usual). No matter whether they are input as a vector or list there must be $M$ elements in `mgshape` and `mgscale`, one for each gamma mixture component. Further, any vectors in the list of parameters must of the same length of the `x,q,p` or equal to the sample size `n`, where relevant.

If `mgweight=NULL` then equal weights for each component are assumed. Otherwise, `mgweight` 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 gamma is defined on the non-negative reals. Though behaviour at zero depends on the shape ($\alpha$):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where $\beta$ is the scale parameter.

## Value

`dmgamma` gives the density, `pmgamma` gives the cumulative distribution function, `qmgamma` gives the quantile function and `rmgamma` gives a random sample.

## Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

## Note

All inputs are vectorised except `log` and `lower.tail`, and the gamma mixture 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 `rmgamma` any input vector must be of length `n`. The only exception is when the parameters are single scalar values, input as vector of length $M$.

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

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

## See Also

gammagpd, gpd and dgamma

Other mgamma: fmgammagpdcon, fmgammagpd, fmgamma, mgammagpdcon, mgammagpd

Other mgammagpd: fgammagpd, fmgammagpdcon, fmgammagpd, fmgamma, gammagpd, mgammagpdcon, mgammagpd

Other mgammagpdcon: fgammagpdcon, fmgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, mgammagpdcon, mgammagpd

Other fmgamma: fmgamma

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 1))

n = 1000
x = rmgamma(n, mgshape = c(1, 6), mgscale = c(1,2), mgweight = c(1, 2))
xx = seq(-1, 40, 0.01)

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgamma(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2)))

# By direct simulation
n1 = rbinom(1, n, 1/3) # sample size from population 1
x = c(rgamma(n1, shape = 1, scale = 1), rgamma(n - n1, shape = 6, scale = 2))

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgamma(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2)))

## End(Not run)
```

---

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 multiple gamma shapes mgshape, scales mgscale and mgweights, threshold u GPD scale sigmau and shape xi and tail fraction phiu.

## Usage

```
dmgammagpd(x, mgshape = 1, mgscale = 1, mgweight = 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 = 1, mgscale = 1, mgweight = 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 = 1, mgscale = 1, mgweight = 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 = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]),
  sigmau = sqrt(mgshape[[1]]) * mgscale[[1]], xi = 0, phiu = TRUE)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `mgshape` | mgamma shape (positive) as list or vector |
| `mgscale` | mgamma scale (positive) as list or vector |
| `mgweight` | mgamma weights (positive) as list or vector (`NULL` for equi-weighted) |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining mixture of gammas 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 $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the mixture of gammas bulk model.

Suppose there are $M >= 1$ gamma components in the mixture model. If you wish to have a single (scalar) value for each parameter within each of the $M$ components then these can be input as a vector of length $M$. If you wish to input a vector of values for each parameter within each of the $M$ components, then they are input as a list with each entry the parameter object for each component (which can either be a scalar or vector as usual). No matter whether they are input as a vector or list there must be $M$ elements in `mgshape` and `mgscale`, one for each gamma mixture component. Further, any vectors in the list of parameters must of the same length of the `x,q,p` or equal to the sample size `n`, where relevant.

If `mgweight=NULL` then equal weights for each component are assumed. Otherwise, `mgweight` 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 cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the mixture of gammas bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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 positive. Though behaviour at zero depends on the shape ($\alpha$):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where $\beta$ is the scale parameter.

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.

## Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

## Note

All inputs are vectorised except `log` and `lower.tail`, and the gamma mixture 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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

## Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

## References

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

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

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

gpd and dgamma

Other gammagpd: fgammagpdcon, fgammagpd, fmgammagpd, fmgamma, gammagpdcon, gammagpd

Other mgamma: fmgammagpdcon, fmgammagpd, fmgamma, mgammagpdcon, mgamma

Other mgammagpd: fgammagpd, fmgammagpdcon, fmgammagpd, fmgamma, gammagpd, mgammagpdcon, mgamma

Other mgammagpdcon: fgammagpdcon, fmgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, mgammagpdcon, mgamma

Other fmgammagpd: fmgammagpd

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rmgammagpd(1000, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2),
  u = 15, sigmau = 4, xi = 0)
xx = seq(-1, 40, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgammagpd(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2),
  u = 15, sigmau = 4, xi = 0))
abline(v = 15)

## End(Not run)
```

`mgammagpdcon` *Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint*

## 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 for upper tail with continuity at threshold. The parameters are the multiple gamma shapes `mgshape`, scales `mgscale` and `mgweights`, threshold `u` GPD shape `xi` and tail fraction `phiu`.

## Usage

```
dmgammagpdcon(x, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0, phiu = TRUE,
  log = FALSE)

pmgammagpdcon(q, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qmgammagpdcon(p, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rmgammagpdcon(n = 1, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0, phiu = TRUE)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `mgshape` | mgamma shape (positive) as list or vector |
| `mgscale` | mgamma scale (positive) as list or vector |
| `mgweight` | mgamma weights (positive) as list or vector (`NULL` for equi-weighted) |
| `u` | threshold |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

**Details**

Extreme value mixture model combining mixture of gammas for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the mixture of gammas bulk model.

Suppose there are $M >= 1$ gamma components in the mixture model. If you wish to have a single (scalar) value for each parameter within each of the $M$ components then these can be input as a vector of length $M$. If you wish to input a vector of values for each parameter within each of the $M$ components, then they are input as a list with each entry the parameter object for each component (which can either be a scalar or vector as usual). No matter whether they are input as a vector or list there must be $M$ elements in `mgshape` and `mgscale`, one for each gamma mixture component. Further, any vectors in the list of parameters must of the same length of the `x,q,p` or equal to the sample size `n`, where relevant.

If `mgweight=NULL` then equal weights for each component are assumed. Otherwise, `mgweight` 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 cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the mixture of gammas bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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 mixture of gammas and conditional GPD density functions respectively. 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 positive. Though behaviour at zero depends on the shape ($\alpha$):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where $\beta$ is the scale parameter.

See `gammagpd` for details of simpler parametric mixture model with single gamma for bulk component and GPD for upper tail.

### Value

dmgammagpdcon gives the density, pmgammagpdcon gives the cumulative distribution function, qmgammagpdcon gives the quantile function and rmgammagpdcon gives a random sample.

### Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

### Note

All inputs are vectorised except log and lower.tail, and the gamma mixture 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 rmgammagpdcon 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 rmgammagpdcon is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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

### Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

### References

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

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

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

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

McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

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

gpd and dgamma

Other gammagpdcon: fgammagpdcon, fgammagpd, fmgammagpdcon, gammagpdcon, gammagpd

Other mgamma: fmgammagpdcon, fmgammagpd, fmgamma, mgammagpd, mgamma

Other mgammagpd: fgammagpd, fmgammagpdcon, fmgammagpd, fmgamma, gammagpd, mgammagpd, mgamma

Other mgammagpdcon: fgammagpdcon, fmgammagpdcon, fmgammagpd, fmgamma, gammagpdcon, mgammagpd, mgamma

Other fmgammagpdcon: fmgammagpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rmgammagpdcon(1000, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2), u = 15, xi = 0)
xx = seq(-1, 40, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgammagpdcon(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2),
 u = 15, xi = 0))
abline(v = 15)

## End(Not run)
```

---

`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 |
| `p.or.n` | logical, should tail fraction (`FALSE`) or number of exceedances (`TRUE`) be given on upper x-axis |
| `alpha` | significance level over range (0, 1), or `NULL` for no CI |
| `ylim` | y-axis limits or `NULL` |
| `legend.loc` | location of legend (see `legend`) or `NULL` for no legend |
| `try.thresh` | vector of thresholds to consider |
| `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.

If the generalised Pareto distribution (GPD) is an appropriate model for the excesses $X - u$ above $u$ then their expected value is:

$$E(X - u|X > u) = \sigma_u/(1 - \xi).$$

For any higher threshold $v > u$ the expected value is

$$E(X - v|X > v) = [\sigma_u + \xi * (v - u)]/(1 - \xi)$$

which is linear in higher thresholds $v$ with intercept given by $[\sigma_u - \xi * u]/(1 - \xi)$ and gradient $\xi/(1-\xi)$. The estimated mean residual life above a threshold $v$ is given by the sample mean excess `mean(x[x > v]) -v`.

Symmetric CLT based confidence intervals are provided, 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 the MRL will be evaluated at each unique datapoint.

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 if requested. The standard deviation and confidence interval are `NA` for less than 5 exceedances.

## Acknowledgments

Based on the `mrlplot` function in the `evd` package for which Stuart Coles' and Alec Stephenson's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

## 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

| | |
|---|---|
| x | quantiles |
| nmean | normal mean |
| nsd | normal standard deviation (positive) |
| u | threshold |
| sigmau | scale parameter (positive) |
| xi | shape parameter |
| phiu | probability of being above threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive 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 $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the normal bulk model (phiu=TRUE), upto the threshold $x \le u$, 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)) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

Hu Y. and Scarrott, C.J. (2018). evmix: An R Package for Extreme Value Mixture Modeling, Threshold Estimation and Boundary Corrected Kernel Density Estimation. Journal of Statistical Software 84(5), 1-27. doi: 10.18637/jss.v084.i05.

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: `fgng`, `fhpd`, `fitmnormgpd`, `flognormgpd`, `fnormgpdcon`, `fnormgpd`, `gngcon`, `gng`, `hpdcon`, `hpd`, `itmnormgpd`, `lognormgpdcon`, `lognormgpd`, `normgpdcon`

Other normgpdcon: `fgngcon`, `fhpdcon`, `flognormgpdcon`, `fnormgpdcon`, `fnormgpd`, `gngcon`, `gng`, `hpdcon`, `hpd`, `normgpdcon`

Other gng: `fgngcon`, `fgng`, `fitmgng`, `fnormgpd`, `gngcon`, `gng`, `itmgng`

Other fnormgpd: `fnormgpd`

## Examples

```
## Not run:
set.seed(1)
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 Single 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 continuity at threshold. 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` | quantiles |
| `nmean` | normal mean |

| | |
|---|---|
| nsd | normal standard deviation (positive) |
| u | threshold |
| xi | shape parameter |
| phiu | probability of being above threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive integer) |

## Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the normal bulk model (`phiu=TRUE`), upto the threshold $x \le u$, 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)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $x \le u$, 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)`) respectively. 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: fgng, fhpd, fitmnormgpd, flognormgpd, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, itmnormgpd, lognormgpdcon, lognormgpd, normgpd

Other normgpdcon: fgngcon, fhpdcon, flognormgpdcon, fnormgpdcon, fnormgpd, gngcon, gng, hpdcon, hpd, normgpd

Other gngcon: fgngcon, fgng, fnormgpdcon, gngcon, gng

Other fnormgpdcon: fnormgpdcon

## Examples

```
## Not run:
set.seed(1)
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)
```

---

`pickandsplot` *Pickands Plot*

---

## Description

Produces the Pickand's plot.

## Usage

```
pickandsplot(data, orderlim = NULL, tlim = NULL, y.alpha = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "topright",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Pickand's Plot", xlab = "order", ylab = ifelse(y.alpha,
  " tail index - alpha", "shape  - xi"), ...)
```

## Arguments

| | |
|---|---|
| `data` | vector of sample data |
| `orderlim` | vector of (lower, upper) limits of order statistics to plot estimator, or `NULL` to use default values |
| `tlim` | vector of (lower, upper) limits of range of threshold to plot estimator, or `NULL` to use default values |
| `y.alpha` | logical, should shape xi (`FALSE`) or tail index alpha (`TRUE`) be given on y-axis |
| `alpha` | significance level over range (0, 1), or `NULL` for no CI |
| `ylim` | y-axis limits or `NULL` |
| `legend.loc` | location of legend (see `legend`) or `NULL` for no legend |

| | |
|---|---|
| `try.thresh` | vector of thresholds to consider |
| `main` | title of plot |
| `xlab` | x-axis label |
| `ylab` | y-axis label |
| `...` | further arguments to be passed to the plotting functions |

## Details

Produces the Pickand's plot including confidence intervals.

For an ordered iid sequence $X_{(1)} \geq X_{(2)} \geq \cdots \geq X_{(n)}$ the Pickand's estimator of the reciprocal of the shape parameter $\xi$ at the $k$th order statistic is given by

$$\hat{\xi}_{k,n} = \frac{1}{\log(2)} \log\left(\frac{X_{(k)} - X_{(2k)}}{X_{(2k)} - X_{(4k)}}\right).$$

Unlike the Hill estimator it does not assume positive data, is valid for any $\xi$ and is location and scale invariant. The Pickands estimator is defined on orders $k = 1, \ldots, \lfloor n/4 \rfloor$.

Once a sufficiently low order statistic is reached the Pickand's estimator will be constant, upto sample uncertainty, for regularly varying tails. Pickand's plot is a plot of

$$\hat{\xi}_{k,n}$$

against the $k$. Symmetric asymptotic normal confidence intervals assuming Pareto tails are provided.

The Pickand's estimator is for the GPD shape $\xi$, or the reciprocal of the tail index $\alpha = 1/\xi$. The shape is plotted by default using `y.alpha=FALSE` and the tail index is plotted when `y.alpha=TRUE`.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The estimated parameter ($\xi$ or $\alpha$) at each threshold are plot by a horizontal solid line for all higher thresholds. The threshold should be set as low as possible, so a dashed line is shown below the pre-chosen threshold. If Pickand's estimator is similar to the dashed line then a lower threshold may be chosen.

If no order statistic (or threshold) limits are provided `orderlim = tlim = NULL` then the lowest order statistic is set to $X_{(1)}$ and highest possible value $X_{\lfloor n/4 \rfloor}$. However, Pickand's estimator is always output for all $k = 1, \ldots, \lfloor n/4 \rfloor$.

The missing (`NA` and `NaN`) and non-finite values are ignored.

The lower x-axis is the order $k$. The upper axis is for the corresponding threshold.

## Value

`pickandsplot` gives Pickand's plot. It also returns a dataframe containing columns of the order statistics, order, Pickand's estimator, it's standard devation and $100(1-\alpha)\%$ confidence interval (when requested).

## Acknowledgments

Thanks to Younes Mouatasim, Risk Dynamics, Brussels for reporting various bugs in these functions.

### Note

Asymptotic Wald type CI's are estimated for non-NULL signficance level alpha for the shape parameter, assuming exactly GPD tails. When plotting on the tail index scale, then a simple reciprocal transform of the CI is applied which may well be sub-optimal.

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

### Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

### References

Pickands III, J.. (1975). Statistical inference using extreme order statistics. Annal of Statistics 3(1), 119-131.

Dekkers A. and de Haan, S. (1989). On the estimation of the extreme-value index and large quantile estimation. Annals of Statistics 17(4), 1795-1832.

Resnick, S. (2007). Heavy-Tail Phenomena - Probabilistic and Statistical Modeling. Springer.

### See Also

pickands

### Examples

```
## Not run:
par(mfrow = c(2, 1))

# Reproduce graphs from Figure 4.7 of Resnick (2007)
data(danish, package="evir")

# Pickand's plot
pickandsplot(danish, orderlim=c(1, 150), ylim=c(-0.1, 2.2),
 try.thresh=c(), alpha=NULL, legend.loc=NULL)

# Using default settings
pickandsplot(danish)

## End(Not run)
```

---

psden *P-Splines probability density function*

---

### Description

Density, cumulative distribution function, quantile function and random number generation for the P-splines density estimate. B-spline coefficients can be result from Poisson regression with log or identity link.

### Usage

```
dpsden(x, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, log = FALSE)

ppsden(q, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, lower.tail = TRUE)

qpsden(p, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, lower.tail = TRUE)

rpsden(n = 1, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, design.knots = NULL)
```

### Arguments

| | |
|---|---|
| `x` | quantiles |
| `beta` | vector of B-spline coefficients (required) |
| `nbinwidth` | scaling to convert count frequency into proper density |
| `xrange` | vector of minimum and maximum of B-spline (support of density) |
| `nseg` | number of segments between knots |
| `degree` | degree of B-splines (0 is constant, 1 is linear, etc.) |
| `design.knots` | spline knots for splineDesign function |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

### Details

P-spline density estimate using B-splines with given coefficients. B-splines knots can be specified using `design.knots` or regularly spaced knots can be specified using `xrange`, `nseg` and `deg`. No default knots are provided.

If regularly spaced knots are specified using `xrange`, `nseg` and `deg`, then B-splines which are shifted/spliced versions of each other are defined (i.e. not natural B-splines) which is consistent with definition of Eilers and Marx, the masters of P-splines.

The `splineDesign` function is used to calculate the B-splines, which intakes knot locations as `design.knots`. As such the `design.knots` are not the knots in their usual sense (e.g. to cover [0, 100] with 10 segments the usual knots would be $0, 10, \ldots, 100$). The `design.knots` must be extended by the degree, so for `degree = 2` the `design.knots = seq(-20,120,10)`.

Further, if the user wants natural B-splines then these can be specified using the `design.knots`, with replicated knots at each bounday according to the degree. To continue the above example, for `degree = 2` the `design.knots = c(rep(0,2),seq(0,100,10),rep(100,2))`.

If both the `design.knots` and other knot specification are provided, then the former are used by default. Default values for only the `degree` and `nseg` are provided, all the other P-spline inputs must be provided. Notice that the `order` and `lambda` penalty are not needed as these are encapsulated in the inference for the B-spline coefficients.

Poisson regression is typically used for estimating the B-spline coefficients, using maximum likelihood estimation (via iterative re-weighted least squares). A log-link function is usually used and as such the `beta` coefficients are on a log-scale, and the density needs to be exponentiated. However, an identity link may be (carefully) used and then these coefficients are on the usual scale.

The `beta` coefficients are estimated using a particular sample (size) and histogram bin-width, using Poisson regression. Thus to convert the predicted counts into a proper density it needs to be rescaled by dividing by $n * binwidth$. If `nbinwidth=NULL` is not provided then a crude approximate scaling is used by normalising the density to be proper. The renormalisation requires numerical integration, which is computationally intensive and so best avoided wherever possible.

Checks of the consistency of the `xrange`, `degree` and `nseg` and `design.knots` are made, with the values implied by the `design.knots` used by default to replace any incorrect values. These replacements are made for notational efficiency for users.

An inversion sampler is used for random number generation which also rather inefficient, as it could be carried out more efficiently using a mixture representation.

The quantile function is rather complicated as there is no closed form solution, so is obtained by numerical approximation of the inverse cumulative distribution function $P(X \leq q) = p$ to find $q$. The quantile function `qpsden` evaluates the P-splines cumulative distribution function over the `xrange`. A sequence of values of length fifty times the number of knots (with a minimum of 1000) 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.

## Value

`dpsden` gives the density, `ppsden` gives the cumulative distribution function, `qpsden` gives the quantile function and `rpsden` gives a random sample.

## Note

Unlike most of the other extreme value mixture model functions the `psden` 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.

Default values are provided for P-spline inputs of `degree` and `nseg` only, but all others must be provided by the user. The default sample size for `rpsden` is 1.

Missing (`NA`) and Not-a-Number (`NaN`) values in `x`, `p` and `q` are passed through as is and infinite values are set to `NA`. None of these are not permitted for the parameters.

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

## Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

## References

http://en.wikipedia.org/wiki/B-spline

http://statweb.lsu.edu/faculty/marx/

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science 11(2), 89-121.

## See Also

splineDesign.

Other psden: fpsdengpd, fpsden, psdengpd

Other psdengpd: fpsdengpd, psdengpd

Other fpsden: fpsden

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 8 internal segments
# CV search for penalty coefficient.
fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
psdensity = exp(fit$bsplines %*% fit$mle)

hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

# P-splines density from dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))

legend("topright", c("True Density","P-spline density"), col=c("black", "blue"), lty = 1)

# plot B-splines
par(mfrow = c(2, 1))
with(fit, matplot(mids, as.matrix(bsplines), type = "l", lty = 1))

# Natural B-splines
knots = with(fit, seq(xrange[1], xrange[2], length.out = nseg + 1))
natural.knots = with(fit, c(rep(xrange[1], degree), knots, rep(xrange[2], degree)))
naturalb = splineDesign(natural.knots, fit$mids, ord = fit$degree + 1, outer.ok = TRUE)
with(fit, matplot(mids, naturalb, type = "l", lty = 1))

# Compare knot specifications
rbind(fit$design.knots, natural.knots)

# User can use natural B-splines if design.knots are specified manually
natural.fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             design.knots = natural.knots, nseg = 10, degree = 3, ord = 2)
psdensity = with(natural.fit, exp(bsplines %*% mle))

par(mfrow = c(1, 1))
hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density
```

```
# check density against dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))
with(natural.fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots),
                        lwd = 2, col = "red", lty = 2))

legend("topright", c("True Density", "Eilers and Marx B-splines", "Natural B-splines"),
   col=c("black", "blue", "red"), lty = c(1, 1, 2))

## End(Not run)
```

---

psdengpd — *P-Splines Density Estimate 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 P-splines density estimate for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the B-spline coefficients `beta` (and associated features), threshold `u` GPD scale `sigmau` and shape `xi` and tail fraction `phiu`.

## Usage

```
dpsdengpd(x, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, u = NULL, sigmau = NULL, xi = 0,
  phiu = TRUE, design.knots = NULL, log = FALSE)

ppsdengpd(q, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, u = NULL, sigmau = NULL, xi = 0,
  phiu = TRUE, design.knots = NULL, lower.tail = TRUE)

qpsdengpd(p, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, u = NULL, sigmau = NULL, xi = 0,
  phiu = TRUE, design.knots = NULL, lower.tail = TRUE)

rpsdengpd(n = 1, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, u = NULL, sigmau = NULL, xi = 0,
  phiu = TRUE, design.knots = NULL)
```

## Arguments

| | |
|---|---|
| `x` | quantiles |
| `beta` | vector of B-spline coefficients (required) |
| `nbinwidth` | scaling to convert count frequency into proper density |
| `xrange` | vector of minimum and maximum of B-spline (support of density) |
| `nseg` | number of segments between knots |
| `degree` | degree of B-splines (0 is constant, 1 is linear, etc.) |
| `u` | threshold |
| `sigmau` | scale parameter (positive) |

| | |
|---|---|
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `design.knots` | spline knots for splineDesign function |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

## Details

Extreme value mixture model combining P-splines density estimate 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 $\phi_u$. Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the KDE bulk model.

The cumulative distribution function with tail fraction $\phi_u$ defined by the upper tail fraction of the P-splines density estimate (`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 P-splines density estimate and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified $\phi_u$, 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. The specification of the underlying B-splines and the P-splines density estimator are discussed in the `psden` function help.

## Value

`dpsdengpd` gives the density, `ppsdengpd` gives the cumulative distribution function, `qpsdengpd` gives the quantile function and `rpsdengpd` gives a random sample.

## Note

Unlike most of the other extreme value mixture model functions the `psdengpd` 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 B-splines coefficients `beta` and knots `design.knots` are vectors.

Default values are provided for P-spline inputs of `degree` and `nseg` only, but all others must be provided by the user. The default sample size for `rpsdengpd` is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are permitted for the parameters/B-spline criteria.

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)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

### References

http://en.wikipedia.org/wiki/B-spline

http://statweb.lsu.edu/faculty/marx/

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

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science 11(2), 89-121.

### See Also

psden and fpsden.

Other psden: fpsdengpd, fpsden, psden

Other psdengpd: fpsdengpd, psden

Other fpsdengpd: fpsdengpd

### Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 8 internal segments
# CV search for penalty coefficient.
fit = fpsdengpd(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))

# P-splines only
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))

# P-splines+GPD
with(fit, lines(xx, dpsdengpd(xx, beta, nbinwidth, design = design.knots,
```

```
    u = u, sigmau = sigmau, xi = xi, phiu = phiu), lwd = 2, col = "red"))
abline(v = fit$u, col = "red")

legend("topleft", c("True Density","P-spline density", "P-spline+GPD"),
 col=c("black", "blue", "red"), lty = 1)

## End(Not run)
```

---

tcplot *Parameter Threshold Stability Plots*

---

## Description

Plots the MLE of the GPD parameters against threshold

## 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 = "bottomright", 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 = "bottomright", 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 = "bottomright", try.thresh = quantile(data, 0.9, na.rm =
  TRUE), main = "Modified Scale Threshold Stability Plot",
  xlab = "Threshold u", ylab = "Modified Scale Parameter", ...)
```

## 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 |
| `p.or.n` | logical, should tail fraction (`FALSE`) or number of exceedances (`TRUE`) be given on upper x-axis |
| `alpha` | significance level over range (0, 1), or `NULL` for no CI |
| `ylim.xi` | y-axis limits for shape parameter or `NULL` |
| `ylim.sigmau` | y-axis limits for scale parameter or `NULL` |
| `legend.loc` | location of legend (see `legend`) or `NULL` for no legend |
| `try.thresh` | vector of thresholds to consider |
| `...` | 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 a set of possible thresholds. If the GPD is a suitable model for a threshold $u$ then for all higher thresholds $v > u$ it will also be suitable, with the shape and modified scale being constant. Known as the threshold stability plots (Coles, 2001). The modified scale parameter is $\sigma_u - u\xi$.

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 uncertainty. 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 shown as a horizontal line which is solid above this threshold, for which they 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 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 if requested. 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.

## Acknowledgments

Based on the threshold stability plot function `tcplot` in the `evd` package for which Stuart Coles' and Alec Stephenson's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

## 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.

By default, no legend is included when using `tcplot` to get both threshold stability plots.

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

| | |
|---|---|
| x | quantiles |
| wshape | Weibull shape (positive) |
| wscale | Weibull scale (positive) |
| u | threshold |
| sigmau | scale parameter (positive) |
| xi | shape parameter |
| phiu | probability of being above threshold $[0, 1]$ or TRUE |
| log | logical, if TRUE then log density |
| q | quantiles |
| lower.tail | logical, if FALSE then upper tail probabilities |
| p | cumulative probabilities |
| n | sample size (positive 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 $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the Weibull 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 Weibull and conditional GPD cumulative distribution functions (i.e. `pweibull(x,wshape,wscale)` and `pgpd(x,u,sigmau,xi)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, 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 Weibull is defined on the non-negative 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: fitmweibullgpd, fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpdcon

Other weibullgpdcon: fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpdcon

Other itmweibullgpd: fitmweibullgpd, fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpdcon

Other fweibullgpd: fweibullgpd

## Examples

```
## Not run:
set.seed(1)
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 Single 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 continuity at threshold. The parameters are the weibull shape `wshape` and scale `wscale`, threshold `u` 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` | quantiles |
| `wshape` | Weibull shape (positive) |
| `wscale` | Weibull scale (positive) |
| `u` | threshold |
| `xi` | shape parameter |
| `phiu` | probability of being above threshold $[0, 1]$ or `TRUE` |
| `log` | logical, if TRUE then log density |
| `q` | quantiles |
| `lower.tail` | logical, if FALSE then upper tail probabilities |
| `p` | cumulative probabilities |
| `n` | sample size (positive integer) |

### Details

Extreme value mixture model combining Weibull distribution for the bulk below the threshold and GPD for upper tail with continuity at threshold.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction $\phi_u$. 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 $\phi_u$ defined by the upper tail fraction of the Weibull bulk model (`phiu=TRUE`), upto the threshold $0 < x \le u$, 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)`) respectively.

The cumulative distribution function for pre-specified $\phi_u$, upto the threshold $0 < x \le u$, 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)`) respectively. 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 Weibull is defined on the non-negative 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.

### Acknowledgments

Thanks to Ben Youngman, Exeter University, UK for reporting a bug in the rweibullgpdcon function.

### 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, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

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: fitmweibullgpd, fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpd

Other weibullgpdcon: fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpd

Other itmweibullgpd: fitmweibullgpd, fweibullgpdcon, fweibullgpd, itmweibullgpd, weibullgpd

Other fweibullgpdcon: fweibullgpdcon

## Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

x = rweibullgpdcon(1000)
xx = seq(-0.1, 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("bottomright", 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("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
  col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

# Index