

R documentation

of all in ‘man’

November 1, 2013

R topics documented:

evmix-package	2
bckden	3
bckdengpd	8
bckdengpdcon	12
betagpd	16
betagpdcon	19
checking	22
dwm	24
evmix.diag	26
fbckden	29
fbckdengpd	34
fbckdengpdcon	38
fbetagpd	43
fbetagpdcon	46
fdwm	49
fgammagpd	52
fgammagpdcon	55
fgkg	59
fgkgcon	63
fgng	68
fgngcon	72
fgpd	76
fhpd	79
fhpdcon	82
fkden	85
fkdengpd	90
fkdengpdcon	94
flognormgpd	98
flognormgpdcon	102
fingamma	105
fingammagpd	108
fingammagpdcon	113
fnormgpd	118
fnormgpdcon	123
fweibullgpd	127

fweibullgpdcon	130
gammagpd	133
gammagpdcon	136
gkg	140
gkgcon	144
gng	148
gngcon	151
gpd	155
hpd	157
hpdcon	160
internal	163
kden	164
kdengpd	168
kdengpdcon	171
kernels	174
kfun	177
lognormgpd	180
lognormgpdcon	182
mgamma	185
mgammagpd	188
mgammagpdcon	191
mrlplot	194
normgpd	196
normgpdcon	199
tcplot	202
weibullgpd	205
weibullgpdcon	207

Index **211**

evmix-package	<i>Extreme Value Mixture Modelling, Threshold Estimation and Boundary Corrected Kernel Density Estimation</i>
---------------	---

Description

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

Details

Package:	evmix
Type:	Package
Version:	0.2-0
Date:	2013-10-20
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)

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

References

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

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

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

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

See Also

[evd](#), [ismev](#) and [condmixt](#)

bckden

Boundary Corrected Kernel Density Estimation 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 λ .

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

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)
xmax	upper bound on support (copula and beta kernels only) or NULL
offset	offset added to kernel centres (logtrans only) or NULL
x	location to evaluate KDE (single scalar or vector)
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

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 $c(0, \max(\text{kerncentre}) + \text{lambda})$, or $c(0, \max(\text{kerncentre}) + 5*\text{lambda})$ 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 occasionally give an extra boundary bias for certain populations (e.g. $\text{Gamma}(2, 1)$), 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 kernel 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.

Note

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

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

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

The `xmax` option is only relevant for the beta and copula methods, so a warning is produced if this is 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>. Based on code by Anna MacDonald produced for MATLAB.

References

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

[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](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 Statistical 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 [kdengpd](#) [kdengpdcon](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fkdengpd](#) [fkdengpdcon](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [bckdengpd](#), [bckdengpdcon](#), [dbckdengpd](#), [dbckdengpdcon](#), [dkden](#), [dkdengpd](#), [dkdengpdcon](#), [fbckden](#), [fkden](#), [kden](#), [kdengpd](#), [kdengpdcon](#), [lbckden](#), [lkden](#), [nlbckden](#), [nlkden](#), [pbckdengpd](#), [pbckdengpdcon](#), [pkden](#), [pkdengpd](#), [pkdengpdcon](#), [qbckdengpd](#), [qbckdengpdcon](#), [qkden](#), [qkdengpd](#), [qkdengpdcon](#), [rbckdengpd](#), [rbckdengpdcon](#), [rkden](#), [rkdengpd](#), [rkdengpdcon](#)

Examples

```
## Not run:
n=100
x = rgamma(n, shape = 1, scale = 2)
xx = seq(-0.5, 12, 0.01)
plot(xx, dgamma(xx, shape = 1, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 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 renormalisation",
  "Simple BC without renormalisation"),
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 λ , threshold u GPD scale σ and shape ξ and tail fraction ϕ .

Usage

```

dbckdengpd(x, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)),
  sigma = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phi = 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)),
  sigma = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phi = 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)),
  sigma = sqrt(6 * var(kerncentres))/pi, xi = 0,
  phi = 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	location to evaluate KDE (single scalar or vector)
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)
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter
phiu	probability of being above threshold $[0, 1]$

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 ϕ_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 ϕ_u defined by the upper tail fraction of the BCKDE (`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 BCKDE and conditional GPD cumulative distribution functions respectively.

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

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

and above the threshold $x > u$:

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

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

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.

Note

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

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

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

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>. Based on code by Anna MacDonald produced for MATLAB.

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 kden [kdengpd](#) [kdengpdcon](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fkdengpd](#) [fkdengpdcon](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [bckden](#), [bckdengpdcon](#), [dbckden](#), [dbckdengpdcon](#), [dkden](#), [dkdengpd](#), [dkdengpdcon](#), [fbckden](#), [fkden](#), [kden](#), [kdengpd](#), [kdengpdcon](#), [lbckden](#), [lkden](#), [nlbckden](#), [nlkden](#), [pbckden](#), [pbckdengpdcon](#), [pkden](#), [pkdengpd](#), [pkdengpdcon](#), [qbckden](#), [qbckdengpdcon](#), [qkden](#), [qkdengpd](#), [qkdengpdcon](#), [rbckden](#), [rbckdengpdcon](#), [rkden](#), [rkdengpd](#), [rkdengpdcon](#)

Examples

```
## Not run:
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 λ , threshold u GPD shape ξ and tail fraction ϕ .

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	location to evaluate KDE (single scalar or vector)
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]
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 ϕ_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 ϕ_u defined by the upper tail fraction of the BCKDE (`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 BCKDE and conditional GPD cumulative distribution functions respectively.

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

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

and above the threshold $x > u$:

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

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

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the 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.

Note

Unlike all 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>. Based on code by Anna MacDonald produced for MATLAB.

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 kden `kdengpd` `kdengpdcon` `bckden` `bckdengpd` `bckdengpdcon` `fkden` `fkdengpd` `fkdengpdcon` `fbckden` `fbckdengpd` `fbckdengpdcon`: `bckden`, `bckdengpd`, `dbckden`, `dbckdengpd`, `dkden`, `dkdengpd`, `dkdengpdcon`, `fbckden`, `fkden`, `kden`, `kdengpd`, `kdengpdcon`, `lbckden`, `lkden`, `n1bckden`, `n1kden`, `pbckden`, `pbckdengpd`, `pkden`, `pkdengpd`, `pkdengpdcon`, `qbckden`, `qbckdengpd`, `qkden`, `qkdengpd`, `qkdengpdcon`, `rbckden`, `rbckdengpd`, `rkden`, `rkdengpd`, `rkdengpdcon`

Examples

```

## Not run:
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, bmethod = "reflect"),
xlab = "x", ylab = "f(x)")
abline(v = quantile(kerncentres, 0.9))

plot(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, bmethod = "reflect"),
xlab = "x", ylab = "F(x)", type = "l")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = 0.3, bmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "red")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = -0.3, bmethod = "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, bmethod = "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, bmethod = "reflect"),
xlab = "x", ylab = "f(x)")

lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, xi=-0.2, phiu = 0.1, bmethod = "reflect"),
xlab = "x", ylab = "f(x)", col = "red")
lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, xi=0.2, phiu = 0.1, bmethod = "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 b_{shape1} and shape 2 b_{shape2} , threshold u GPD scale σ and shape ξ and tail fraction ϕ .

Usage

```

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

pbetagpd(q, bshape1 = 1, bshape2 = 1,

```

```

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

```

```

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

```

```

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

```

Arguments

bshape1	beta shape 1 (positive)
bshape2	beta shape 2 (positive)
u	threshold over (0, 1)
phiu	probability of being above threshold [0, 1] or TRUE
x	quantiles
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

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

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

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

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the beta and conditional GPD cumulative distribution functions (i.e. `pbeta(x, bshape1, bshape2)` and `pgpd(x, u, sigmau, xi)`).

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

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

and above the threshold $x > u$:

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

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

See [gpd](#) for details of GPD upper tail component and [dbeta](#) for details of beta bulk component.

Value

[dbetagpd](#) gives the density, [pbetagpd](#) gives the cumulative distribution function, [qbetagpd](#) gives the quantile function and [rbetagpd](#) gives a random sample.

Note

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

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for [rbetagpd](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in `x`, `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` `fbetagpd` `fbetagpdcon`: [betagpdcon](#), [dbetagpdcon](#), [pbetagpdcon](#), [qbetagpdcon](#), [rbetagpdcon](#)

Examples

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

# three tail behaviours
plot(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2), type = "l")
lines(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2, xi = 0.3), col = "red")
lines(xx, pbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2, xi = -0.3), col = "blue")
legend("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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the beta bulk model.

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the beta and conditional GPD cumulative distribution functions (i.e. `pbeta(x, bshape1, bshape2)` and `pgpd(x, u, sigmau, xi)`).

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

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

and above the threshold $x > u$:

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

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

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 `dgpdc(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 betagpdcon fbetagpd fbetagpdcon: [betagpd](#), [dbetagpd](#), [pbetagpd](#), [qbetagpd](#), [rbetagpd](#)

Examples

```
## Not run:
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

Functions for checking function input argument

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.

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.quant(x, allowvec = TRUE, allownull = FALSE,
  allowmiss = FALSE, allowinf = FALSE)

check.prob(prob, allowvec = TRUE, allownull = FALSE,
  allowmiss = FALSE)

check.n(n)

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

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

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

check.nparam(param, nparam = 1, allownull = FALSE)

check.text(textarg, allowvec = FALSE)

check.inputn(inputn, allowvec = TRUE)

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)
```

Arguments

x	scalar or vector of quantiles
prob	scalar or vector of probability
n	scalar sample size
param	scalar or vector of parameters
phiu	scalar or vector of phiu (logical, NULL or 0-1 exclusive)
nparam	vector of lengths of parameter vectors
logicarg	logical input argument
textarg	character input argument
inputn	vector of input lengths
allowvec	logical, where TRUE permits vector
allownull	logical, where TRUE permits NULL values
allowmiss	logical, where TRUE permits NA and NaN values
allowna	logical, where TRUE permits NA values
allowinf	logical, where TRUE permits Inf values
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

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

<code>cmu</code>	Cauchy location
<code>ctau</code>	Cauchy scale
<code>qinit</code>	scalar or vector of initial values for the quantile estimate
<code>wshape</code>	Weibull shape (positive)
<code>wscale</code>	Weibull scale (positive)
<code>x</code>	quantiles
<code>sigmau</code>	scale parameter (positive)
<code>xi</code>	shape parameter
<code>log</code>	logical, if TRUE then log density
<code>q</code>	quantiles
<code>lower.tail</code>	logical, if FALSE then upper tail probabilities
<code>p</code>	cumulative probabilities
<code>n</code>	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, sigma, 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](#)

Examples

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

Description

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

Usage

```

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

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

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

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

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

```

Arguments

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

Details

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

Consistent with [plot](#) function in the [evd](#) library the [ppoints](#) to estimate the empirical cumulative probabilities. The 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 transformation 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 ϕ_{iu} are shown. For the two tailed models both thresholds u_l and u_r and corresponding tail fractions ϕ_{iul} and ϕ_{iur} 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.

Note

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

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

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

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

Author(s)

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

References

Based on GPD/POT diagnostic functions in the `evd` function from the `evd` package.

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

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

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

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

See Also

`ppoints`, `plot.uvevd` and `gpd.diag`.

Examples

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

# repeat without focussing on upper tail
par(mfrow=c(2,2))
r1plot(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

linit	initial value for bandwidth (as kernel half-width) or NULL
bwinit	initial value for bandwidth (as kernel standard deviations) or NULL
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
add.jitter	logical, whether jitter is needed for rounded kernel centres

factor	see jitter
amount	see jitter
lambda	scalar bandwidth for kernel (as half-width of kernel)
bw	scalar bandwidth for kernel (as standard deviations of kernel)
x	location to evaluate KDE (single scalar or vector)
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
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The 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

$$\hat{f}_{-i}(x_i) = \frac{1}{(n-1)\lambda} \sum_{j=1:j \neq i}^n K\left(\frac{x_i - x_j}{\lambda}\right)$$

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

<code>call:</code>	<code>optim</code> call
<code>x:</code>	(jittered) data vector <code>x</code>
<code>kerncentres:</code>	actual kernel centres used <code>x</code>
<code>init:</code>	linit for lambda
<code>optim:</code>	complete <code>optim</code> output
<code>mle:</code>	vector of MLE of bandwidth
<code>cov:</code>	variance of MLE of bandwidth
<code>se:</code>	standard error of MLE of bandwidth
<code>nllh:</code>	minimum negative cross-validation log-likelihood
<code>n:</code>	total sample size
<code>lambda:</code>	MLE of lambda (kernel half-width)
<code>bw:</code>	MLE of bw (kernel standard deviations)
<code>kernel:</code>	kernel name
<code>bcmethod:</code>	boundary correction method
<code>proper:</code>	logical, whether renormalisation is requested
<code>nn:</code>	non-negative correction method
<code>offset:</code>	offset for log transformation method
<code>xmax:</code>	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 $\hat{\lambda} \rightarrow 0$ 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 occasionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (x only) has been included in the fitting inputs, using the `jitter` function, to remove the ties. The default options red in the `jitter` are specified above, but the user can override these. Notice the default scaling factor=0.1, which is a tenth of the default value in the `jitter` function itself.

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

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

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, $\log(0)=-\text{Inf}$ for log-likelihood and $-\log(0)=\text{Inf}$ 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>. Based on code by Anna MacDonald produced for MATLAB.

References

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

[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](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 kdengpd kdengpdcon bckden bckdengpd bckdengpdcon fkden fkdengpd fkdengpdcon fbckden fbckdengpd fbckdengpdcon: [bckden](#), [bckdengpd](#), [bckdengpdcon](#), [dbckden](#), [dbckdengpd](#), [dbckdengpdcon](#), [dkden](#), [dkdengpd](#), [dkdengpdcon](#), [fkden](#), [kden](#), [kdengpd](#), [kdengpdcon](#), [lkden](#), [nlkden](#), [pbckden](#), [pbckdengpd](#), [pbckdengpdcon](#), [pkden](#), [pkdengpd](#), [pkdengpdcon](#), [qbckden](#), [qbckdengpd](#), [qbckdengpdcon](#), [qkden](#), [qkdengpd](#), [qkdengpdcon](#), [rbckden](#), [rbckdengpd](#), [rbckdengpdcon](#), [rkden](#), [rkdengpd](#), [rkdengpdcon](#)

Examples

```
## Not run:
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 = FALSE, ...)

```

```

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

```

Arguments

add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter

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
u	scalar threshold value
x	location to evaluate KDE (single scalar or vector)
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
lambda	bandwidth for kernel (as half-width of kernel) or NULL
sigmau	scale parameter (positive)
xi	shape parameter
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
log	logical, if TRUE then log density
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The extreme value mixture model with 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 [prof1ubckdengpd](#). [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
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
nlh:	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.phi:	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 MATLAB code written by Anna MacDonald.

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/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](#) [kdengpdcon](#) [fkdengpd](#) [fkdengpdcon](#) [normgpd](#) [fnormgpd](#) [kden](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [fbckdengpdcon](#), [fkdengpd](#), [fkdengpdcon](#), [lbckdengpdcon](#), [lkdengpd](#), [lkdengpdcon](#), [nlbckdengpdcon](#), [nlkdengpd](#), [nlkdengpdcon](#), [nlubckdengpdcon](#), [nlukdengpd](#), [nlukdengpdcon](#), [proflubckdengpdcon](#), [proflukdengpd](#), [proflukdengpdcon](#)

Examples

```
## Not run:
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 threshold with 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 = FALSE, ...)
```

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

Arguments

x	location to evaluate KDE (single scalar or vector)
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	shape parameter
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
log	logical, if TRUE then log density

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 (λ, u, ξ) if threshold is also estimated and (λ, ξ) 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
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.phi:    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 MATLAB code written by Anna MacDonald.

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/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](#) [kdengpdcon](#) [fkdengpd](#) [fkdengpdcon](#) [normgpd](#) [fnormgpd](#) [kden](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [fbckdengpd](#), [fkdengpd](#), [fkdengpdcon](#), [lbckdengpd](#), [lkdengpd](#), [lkdengpdcon](#), [nlbckdengpd](#), [nlkdengpd](#), [nlkdengpdcon](#), [nlubckdengpd](#), [nlukdengpd](#), [nlukdengpdcon](#), [proflubckdengpd](#), [proflukdengpd](#), [proflukdengpdcon](#)

Examples

```
## Not run:
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 continuity 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 = FALSE, ...)

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

Arguments

bshape1	scalar beta shape 1 (positive)
bshape2	scalar beta shape 2 (positive)
u	scalar threshold over (0, 1)
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
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters

...	optional inputs passed to optim
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
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.phi:	standard error of MLE of tail fraction

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`. Based on MATLAB code written by Anna MacDonald.

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 fbetagpd fbetagpdcon normgpd fnormgpd: [fbetagpdcon](#), [lbetagpdcon](#), [nlbetagpdcon](#), [nlubetagpdcon](#), [proflubetagpdcon](#)

Examples

```
## Not run:
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, sigma, 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, sigma, 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(0.5, 0.95, length = 20))
fitfix = fbetagpd(x, useq = seq(0.5, 0.95
, 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, sigma, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigma, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigma, 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 = FALSE, ...)

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 sigma parameter is now specified as function of other parameters, see help for [dbetagpdcon](#) for details, type `help betagpdcon`. Therefore, sigma 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
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 MATLAB code written by Anna MacDonald.

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](#) [fbetagpd](#) [fbetagpdcon](#) [normgpd](#) [fnormgpd](#): [fbetagpd](#), [lbetagpd](#), [nlbetagpd](#), [nlbetagpd](#), [proflubetagpd](#)

Examples

```

## Not run:
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 continuity constraint"),
      col=c("black", "blue", "red"), lty = 1)

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fbetagpdcon(x, useq = seq(0.5, 0.95, length = 20))
fitfix = fbetagpdcon(x, useq = seq(0.5, 0.95, 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

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

Details

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

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.

Note

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

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

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](#)

Examples

```
## Not run:
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)
lines(xx, ddwm(xx, wshape = fit$wshape, wscale = fit$wscale, cmu = fit$cmu, ctau = fit$ctau,
  sigmau = fit$sigmau, xi = fit$xi), col="red")

## End(Not run)
```

fgammagpd

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

Description

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

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

Arguments

gshape	scalar gamma shape (positive)
gscale	scalar gamma scale (positive)
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
u	scalar threshold value
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
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
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 `mgamma` `fmgamma` `gammagpd` `gammagpdcon` `fgammagpd` `fgammagpdcon` `normgpd` `fnormgpd` `mgammagpd` `mgammagpdcon` `fmgammagpd` `fmgammagpdcon`: [dammagpd](#), [dammagpdcon](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMgamma](#), [nlEMgammagpd](#), [nlEMgammagpdcon](#), [nlgammagpdcon](#), [nlmgamma](#),

nlgammagpd, nlgammagpdcon, nluEMgammagpd, nluEMgammagpdcon, nlugammagpdcon, nlugammagpd, nlugammagpdcon, pgammagpd, pgammagpdcon, pmgamma, pmgammagpd, pmgammagpdcon, proflugammagpdcon, proflumgammagpd, proflumgammagpdcon, qgammagpd, qgammagpdcon, qmgamma, qmgammagpd, qmgammagpdcon, rgammagpd, rgammagpdcon, rmgamma, rmgammagpd, rmgammagpdcon

Examples

```
## Not run:
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 = FALSE, ...)
```

```
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 sigma parameter is now specified as function of other parameters, see help for [dgammagpdcon](#) for details, type `help gammagpdcon`. Therefore, sigma 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 its 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
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.phi:	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 mgamma fmgamma gammagpd gammagpdcon fgammagpd fgammagpdcon normgpd fnormgpd mgammagpd mgammagpdcon fmgammagpd fmgammagpdcon: [dgammagpd](#), [dgammagpdcon](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMmgamma](#), [nlEMmgammagpd](#), [nlEMmgammagpdcon](#), [nlgammagpd](#), [nlmgamma](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMmgammagpd](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlumgammagpd](#), [nlumgammagpdcon](#), [pgammagpd](#), [pgammagpdcon](#), [pmgamma](#), [pmgammagpd](#), [pmgammagpdcon](#), [proflugammagpd](#), [proflumgammagpd](#), [proflumgammagpdcon](#), [qgammagpd](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpd](#), [rgammagpdcon](#), [rmgamma](#), [rmgammagpd](#), [rmgammagpdcon](#)

Examples

```
## Not run:
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 continuity 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$u, col = "purple")
with(fitfix, lines(xx, dgammapdcon(xx, gshape, gscale, u, xi), col="darkgreen"))
abline(v = fitfix$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 = FALSE, ...)

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

```

Arguments

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 useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
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
ulr	vector of length 2 giving lower and upper tail thresholds or NULL for default values
lambda	scalar bandwidth for kernel (as half-width of kernel)
bw	scalar bandwidth for kernel (as standard deviations of kernel)
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter
x	quantiles
kernel	kernel name (default = "gaussian")
log	logical, if TRUE then log density
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

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 $(\lambda, u_l, \sigma_{ul}, x_{il}, u_r, \sigma_{ur}, x_{ir})$ if threshold is also estimated and $(\lambda, \sigma_{ul}, x_{il}, \sigma_{ur}, x_{ir})$ 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 λ 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 ϕ_{iul} and ϕ_{iur} 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 survivor 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`, `nlkgk` 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, $\text{phiul} + \text{phiur} < 1$ as bulk must contribute.

Value

Log-likelihood is given by `lgkg` and it's wrappers for negative log-likelihood from `nlkgk` and `nlugkg`. Profile likelihood for single pair of upper and lower thresholds given by `proflugkg`. Fitting function `fgkg` returns a simple list with the following elements

<code>call:</code>	optim call
<code>x:</code>	data vector x
<code>init:</code>	pvector
<code>fixedu:</code>	fixed threshold, logical
<code>useq:</code>	threshold vector for profile likelihood or scalar for fixed threshold
<code>optim:</code>	complete optim output
<code>mle:</code>	vector of MLE of parameters
<code>cov:</code>	variance-covariance matrix of MLE of parameters
<code>se:</code>	vector of standard errors of MLE of parameters
<code>rate:</code>	phiu to be consistent with <code>evd</code>
<code>nllh:</code>	minimum negative log-likelihood
<code>n:</code>	total sample size
<code>lambda:</code>	MLE of lambda (kernel half-width)
<code>ul:</code>	lower threshold (fixed or MLE)
<code>sigmaul:</code>	MLE of lower tail GPD scale
<code>xil:</code>	MLE of lower tail GPD shape
<code>phiul:</code>	MLE of lower tail fraction (bulk model or parameterised approach)
<code>se.phiul:</code>	standard error of MLE of lower tail fraction
<code>ur:</code>	upper threshold (fixed or MLE)
<code>sigmaur:</code>	MLE of upper tail GPD scale
<code>xir:</code>	MLE of upper tail GPD shape
<code>phiur:</code>	MLE of upper tail fraction (bulk model or parameterised approach)
<code>se.phiur:</code>	standard error of MLE of upper tail fraction
<code>bw:</code>	MLE of bw (kernel standard deviations)
<code>kernel:</code>	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 MATLAB code written by Anna MacDonald.

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/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 `kdengpd` `kdengpdcon` `fkdenpd` `fkdenpdcon` `normgpd` `fnormgpd` `gkg` `gkgcon` `fgkg` `fgkgcon` `kden` `bckden` `bckdenpd` `bckdenpdcon` `fkden` `fbckden` `fbckdenpd` `fbckdenpdcon`: [fgkgcon](#), [lgkgcon](#), [nlkgkgcon](#), [nlugkgcon](#), [proflugkgcon](#)

Examples

```

## Not run:
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 = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 10))
fitfix = fgkg(x, ulseq = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 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)
```

```
nlkgkgcon(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 = FALSE, ...)
```

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

Arguments

x	quantiles
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 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)
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 density
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 threshold is 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 survivor 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](#), [nlkgkgcon](#) 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.

Value

Log-likelihood is given by [lgkgcon](#) and it's wrappers for negative log-likelihood from [nlkgkgcon](#) and [nlugkgcon](#). Profile likelihood for single pair of upper and lower 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 threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
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 MATLAB code written by Anna MacDonald.

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/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 [kdengpd](#) [kdengpdcon](#) [fkdengpd](#) [fkdengpdcon](#) [normgpd](#) [fnormgpd](#) [gkg](#) [gkgcon](#) [fgkg](#) [fgkgcon](#) [kden](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [fgkg](#), [lgkg](#), [nlgkg](#), [nlugkg](#), [proflugkg](#)

Examples

```
## Not run:
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 continuity constraint"),
  col=c("black", "blue", "red"), lty = 1)

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

```

fitu = fgkgcon(x, ulseq = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 10))
fitfix = fgkgcon(x, ulseq = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 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)

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

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

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

```

Arguments

ul	scalar lower tail threshold
sigmaul	scalar lower tail GPD scale parameter (positive)
xil	scalar lower tail GPD shape parameter
phiul	probability of being below lower threshold (0, 1) or logical, see Details in help for fgng
ur	scalar upper tail threshold
sigmaur	scalar upper tail GPD scale parameter (positive)
xir	scalar upper tail GPD shape parameter
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 useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
ulr	vector of length 2 giving lower and upper tail thresholds or NULL for default values
pvector	vector of initial values of parameters or NULL for default values, see below
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
log	logical, if TRUE then log-likelihood rather than likelihood is output

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 threshold is 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 survivor 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, $\text{phiul} + \text{phiur} < 1$ as bulk must contribute.

Value

Log-likelihood is given by `lgng` and it's wrappers for negative log-likelihood from `nlgng` and `nlugng`. Profile likelihood for single threshold 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 threshold, logical
useq:      threshold vector for profile likelihood or scalar for fixed threshold
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 MATLAB code written by Xin Zhao.

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](#) [normgpdcon](#) [gng](#) [gngcon](#) [fnormgpd](#) [fnormgpdcon](#) [fgng](#) [fgngcon](#): [dgng](#), [dgngcon](#), [dnormgpd](#), [dnormgpdcon](#), [fgngcon](#), [fnormgpd](#), [fnormgpdcon](#), [gng](#), [gngcon](#), [lgngcon](#), [lnormgpd](#), [lnormgpdcon](#), [nlngngcon](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugngcon](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpd](#), [normgpdcon](#), [pgng](#), [pgngcon](#), [pnormgpd](#), [pnormgpdcon](#), [proflugngcon](#), [proflunormgpd](#), [proflunormgpdcon](#), [qgng](#), [qgngcon](#), [qnormgpd](#), [qnormgpdcon](#), [rgng](#), [rgngcon](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
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 = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 10))
fitfix = fgng(x, ulseq = rep(seq(-2, -0.2, length = 10), times = 10),
  urseq = rep(seq(0.2, 2, length = 10), each = 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)

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

proflungcon(ulr, pvector, x, phiul = TRUE, phiur = TRUE,

```

```
method = "BFGS", control = list(maxit = 10000),
finitelik = FALSE, ...)
```

```
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 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)
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 [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 [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 threshold is also estimated and `(nmean, nsd, xil, xir)` for profile likelihood or fixed threshold approach.

Value

Log-likelihood is given by [lgngcon](#) and its wrappers for negative log-likelihood from [nlngngcon](#) and [nlugngcon](#). Profile likelihood for single threshold 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 threshold, logical
useq:      threshold vector for profile likelihood or scalar for fixed threshold
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 MATLAB code written by Xin Zhao.

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 normgpd normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dgng](#), [dgngcon](#), [dnormgpd](#), [dnormgpdcon](#), [fgng](#), [fnormgpd](#), [fnormgpdcon](#), [gng](#), [gngcon](#), [lgng](#), [lnormgpd](#), [lnormgpdcon](#), [nlngng](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugng](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpd](#), [normgpdcon](#), [pgng](#), [pgngcon](#), [pnormgpd](#), [pnormgpdcon](#), [proflugng](#), [proflunormgpd](#), [proflunormgpdcon](#), [qgng](#), [qgngcon](#), [qnormgpd](#), [qnormgpdcon](#), [rgng](#), [rgngcon](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
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 continuity constraint"),
  col=c("black", "blue", "red"), lty = 1)

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

```

urseq = rep(seq(0.2, 2, length = 10), each = 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 σ and shape ξ to the threshold exceedances, conditional on being above a threshold u . Unconditional likelihood fitting also provided when the probability ϕ 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

<code>x</code>	vector of sample data
<code>u</code>	scalar threshold
<code>sigmau</code>	scalar scale parameter (positive)
<code>xi</code>	scalar shape parameter
<code>pvector</code>	vector of initial values of GPD parameters (<code>sigmau</code> , <code>xi</code>) or NULL
<code>phiu</code>	probability of being above threshold $[0, 1]$ or NULL, see Details
<code>std.err</code>	logical, should standard errors be calculated
<code>method</code>	optimisation method (see optim)
<code>control</code>	optimisation control list (see optim)
<code>finitelik</code>	logical, should log-likelihood return finite value for invalid parameters
<code>log</code>	logical, if TRUE then log-likelihood rather than likelihood is output
<code>...</code>	optional inputs passed to optim

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.phi`, 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:

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

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

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

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

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

Value

`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.phi: 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.

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 [fgpd](#), 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 [nlpd](#).

The [lpgd](#) also has the usual defaults for the other parameters, but [nlpd](#) has no defaults.

Infinite sample values are dropped in fitting function [fgpd](#), but missing values are used to estimate phiu as described above. But in likelihood functions [lpgd](#) and [nlpd](#) 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

Based on GPD fitting function in the [evd](#) package.

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

See Also

[dgp](#), [fpot](#) and [fitdistr](#)

Other gpd fgpd: [dgp](#), [gpd](#), [pgpd](#), [qgpd](#), [rgpd](#)

Examples

```

par(mfrow=c(2,1))
x = rgpd(1000, u = 10, sigmau = 5, xi = 0.2)
xx = seq(0, 100, 0.1)
hist(x, breaks = 100, freq = FALSE, xlim = c(0, 100))

```

```

lines(xx, dgpd(xx, u = 10, sigmau = 5, xi = 0.2))
fit = fgpd(x, u = 10, phiu = NULL, std.err = FALSE)
lines(xx, dgpd(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi), col="red")

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

```

fhpd

MLE Fitting of Hybrid Pareto Extreme Value Mixture Model

Description

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

Usage

```

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

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

nlhpd(pvector, x, finitelik = FALSE)

```

Arguments

pvector	vector of initial values of parameters (nmean, nsd, xi) or NULL
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
x	quantiles
xi	shape parameter
log	logical, if TRUE then log density
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

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

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
sigmau:    MLE of GPD scale
xi:        MLE of GPD shape

```

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

Note

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

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

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, $\log(0)=-\text{Inf}$ for log-likelihood and $-\log(0)=\text{Inf}$ 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 `hpdcon` `normgpd` `normgpdcon` `gng` `gngcon` `fhpd` `fhpdcon` `fnormgpd` `fnormgpdcon` `fgng` `fgngcon`: [fhpdcon](#), [lhpdcon](#), [nlhpdcon](#), [nluhpdcon](#), [profluahpdcon](#)

Examples

```
## Not run:
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)
```

```

lines(xx, dhpd(xx, nmean = fit$nmean, nsd = fit$nsd, xi = fit$xi), col="red")
abline(v = fit$u)

# Notice that if tail fraction is included a better fit is obtained
fit2 = fnormgpdcon(x, std.err = FALSE)
lines(xx, dnormgpdcon(xx, nmean = fit2$nmean, nsd = fit2$nsd, u = fit2$u, xi = fit2$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 = FALSE, ...)

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

```

Arguments

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
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
u	scalar threshold value
x	quantiles
xi	shape parameter

log	logical, if TRUE then log density
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The hybrid Pareto model is fitted to the entire dataset using maximum likelihood estimation, 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 its 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 [prof1uhpdcon](#). `fhpdcn` returns a simple list with the following elements

call:	<code>optim</code> call
x:	data vector <code>x</code>
init:	<code>pvector</code>
fixedu:	fixed threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
optim:	complete <code>optim</code> 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:	<code>phiu</code> 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 $1/(1+pnorm(u, nmean, nsd))$

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 hpdcon normgpd normgpdcon gng gngcon fhpd fhpdcon fnormgpd fnormgpdcon fgng fgngcon: [fhpd](#), [lhpd](#), [nlhpd](#)

Examples

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

```
# Hybrid Pareto provides reasonable fit for 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, dhpdcn(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fhpd(x)
with(fit2, lines(xx, dhpdcn(xx, nmean, nsd, xi), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density", "No continuity constraint", "With continuity 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, dhpdcn(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dhpdcn(xx, nmean, nsd, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dhpdcn(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)

hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 4))
lines(xx, y)
with(fit, lines(xx, dhpdcn(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)

```

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

linit	initial value for bandwidth (as kernel half-width) or NULL
bwinit	initial value for bandwidth (as kernel standard deviations) or NULL
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter
x	quantiles
kernel	kernel name (default = "gaussian")
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 density
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

Details

The kernel density estimator (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

$$\hat{f}_{-i}(x_i) = \frac{1}{(n-1)\lambda} \sum_{j=1:j \neq i}^n K\left(\frac{x_i - x_j}{\lambda}\right)$$

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 $i = 1, \dots, nb, nb + 1, \dots, nb + nu$. The cross-validation likelihood using the extra kernel centres is then:

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

where

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

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;
- `nlnornlkdenmgpd` - 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).

Default 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 `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 `finiteolik=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 `finiteolik` will be overridden and set to `finiteolik=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

<code>call:</code>	<code>optim</code> call
<code>x:</code>	(jittered) data vector <code>x</code>
<code>kerncentres:</code>	actual kernel centres used <code>x</code>
<code>init:</code>	<code>linit</code> for <code>lambda</code>
<code>optim:</code>	complete <code>optim</code> output
<code>mle:</code>	vector of MLE of bandwidth
<code>cov:</code>	variance of MLE of bandwidth
<code>se:</code>	standard error of MLE of bandwidth
<code>nllh:</code>	minimum negative cross-validation log-likelihood
<code>n:</code>	total sample size
<code>lambda:</code>	MLE of <code>lambda</code> (kernel half-width)
<code>bw:</code>	MLE of <code>bw</code> (kernel standard deviations)
<code>kernel:</code>	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 occasionally rears its ugly head for data which has been heavily rounded, as even when using cross-validation the density can be non-zero even if the bandwidth is zero. To overcome this issue an option to add a small jitter should be added to the data (`x` only) has been included in the fitting inputs, using the `jitter` function, to remove the ties. The default options red in the `jitter` are specified above, but the user can override these. Notice the default scaling `factor=0.1`, which is a tenth of the default value in the `jitter` function itself.

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

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

Note

When `limit=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, $\log(0)=-\text{Inf}$ for log-likelihood and $-\log(0)=\text{Inf}$ 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_\(statistics\)](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 [kdengpd](#) [kdengpdcon](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fkdengpd](#) [fkdengpdcon](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [bckden](#), [bckdengpd](#), [bckdengpdcon](#), [dbckden](#), [dbckdengpd](#), [dbckdengpdcon](#), [dkden](#), [dkdengpd](#), [dkdengpdcon](#), [fbckden](#), [kden](#), [kdengpd](#), [kdengpdcon](#), [lbckden](#), [nlbckden](#), [pbckden](#), [pbckdengpd](#), [pbckdengpdcon](#), [pkden](#), [pkdengpd](#), [pkdengpdcon](#), [qbckden](#), [qbckdengpd](#), [qbckdengpdcon](#), [qkden](#), [qkdengpd](#), [qkdengpdcon](#), [rbckden](#), [rbckdengpd](#), [rbckdengpdcon](#), [rkden](#), [rkdengpd](#), [rkdengpdcon](#)

Examples

```

## Not run:
nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
fit = fkdengpd(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"))

# bandwidth is biased towards oversmoothing for heavy tails
nk=100
x = rt(nk, df = 2)
xx = seq(-8, 8, 0.01)
fit = fkdengpd(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 = fkdengpd(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)

```

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 = FALSE, ...)
```

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

Arguments

bw	scalar bandwidth for kernel (as standard deviations of kernel)
lambda	scalar bandwidth for kernel (as half-width of kernel)
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
u	scalar threshold value
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

kernel	kernel name (default = "gaussian")
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter

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 $(\lambda, u, \sigma, \xi)$ if threshold is also estimated and (λ, σ, ξ) 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 λ 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
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:	ϕ to be consistent with evd
nllh:	minimum negative log-likelihood
n:	total sample size
lambda:	MLE of λ (kernel half-width)
u:	threshold (fixed or MLE)
sigma:	MLE of GPD scale
xi:	MLE of GPD shape
phi:	MLE of tail fraction (bulk model or parameterised approach)
se.phi:	standard error of MLE of tail fraction
bw:	MLE of <code>bw</code> (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 MATLAB code written by Anna MacDonald.

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/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 `kdengpd` `kdengpdcon` `fkdengpd` `fkdengpdcon` `normgpd` `fnormgpd` `kden` `bckden` `bckdengpd` `bckdengpdcon` `fkden` `fbckden` `fbckdengpd` `fbckdengpdcon`: [fbckdengpd](#), [fbckdengpdcon](#), [fkdengpdcon](#), [lbckdengpd](#), [lbckdengpdcon](#), [lkdengpdcon](#), [nlbckdengpd](#), [nlbckdengpdcon](#), [nlkdengpdcon](#), [nlubckdengpd](#), [nlubckdengpdcon](#), [nlukdengpdcon](#), [proflubckdengpd](#), [proflubckdengpdcon](#), [proflukdengpdcon](#)

Examples

```

## Not run:
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)

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

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 (λ, u, ξ) if threshold is also estimated and (λ, ξ) 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
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.phi:    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 MATLAB code written by Anna MacDonald.

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/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 `kdengpd` `kdengpdcon` `fkdengpd` `fkdengpdcon` `normgpd` `fnormgpd` `kden` `bckden` `bckdengpd` `bckdengpdcon` `fkden` `fbckden` `fbckdengpd` `fbckdengpdcon`: [fbckdengpd](#), [fbckdengpdcon](#), [fkdengpd](#), [lbckdengpd](#), [lbckdengpdcon](#), [lkdengpd](#), [nlbckdengpd](#), [nlbckdengpdcon](#), [nlkdengpd](#), [nlubckdengpd](#), [nlubckdengpdcon](#), [nlukdengpd](#), [proflubckdengpd](#), [proflubckdengpdcon](#), [proflukdengpd](#)

Examples

```

## Not run:
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 continuity 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, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd),
  sigmau = sqrt(lnmean) * lnstd, 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 = FALSE, ...)

nluognormgpd(pvector, u, x, phiu = TRUE,
  finitelik = FALSE)

```

Arguments

lnmean	scalar mean on log scale
lnstd	scalar standard deviation on log scale (positive)
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
u	scalar threshold value
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
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, lnstd, u, sigmau, xi) if threshold is also estimated and (lnmean, lnstd, sigmau, xi) for profile likelihood or fixed threshold approach.

Non-positive data are ignored.

Value

Log-likelihood is given by `llognormgpd` and its 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
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 lognormgpdcon flognormgpd flognormgpdcon normgpd fnormgpd: [flognormgpdcon](#), [llognormgpdcon](#), [nllognormgpdcon](#), [nlulognormgpdcon](#), [proflulognormgpdcon](#)

Examples

```
## Not run:
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, lnstd, u, sigma, xi), col="red"))
abline(v = fit$u, col = "red")

# Parameterised tail fraction
fit2 = flognormgpd(x, phi = FALSE)
with(fit2, lines(xx, dlognormgpd(xx, lnmean, lnstd, u, sigma, xi, phi), 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, lnstd, u, sigma, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dlognormgpd(xx, lnmean, lnstd, u, sigma, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dlognormgpd(xx, lnmean, lnstd, u, sigma, 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	<i>MLE Fitting of log-normal Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint</i>
----------------	--

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, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), xi = 0, phiu = TRUE,
  log = TRUE)

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

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

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
lnstd	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 sigma parameter is now specified as function of other parameters, see help for [dlognormgpdcon](#) for details, type `help lognormgpdcon`. Therefore, sigma 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
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.phi:	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 lognormgpdcon flognormgpd flognormgpdcon normgpd fnormgpd: [flognormgpd](#), [llognormgpd](#), [nllognormgpd](#), [nlulognormgpd](#), [proflulognormgpd](#)

Examples

```
## Not run:
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, lnstd, 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, lnstd, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density", "No continuity constraint", "With continuity 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

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
M	number of gamma components in mixture
tau	matrix of posterior probability of being in each component (nxM where n is length(x))
x	vector of sample data
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
log	logical, if TRUE then log-likelihood rather than likelihood is output

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 [nlEMgamma](#).

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 [nlEMgamma](#), 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 $2 \times M$ 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 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 `n1mgamma`. The conditional negative log-likelihood using the posterior probabilities is given by `n1EMmgamma`. 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 probabilities
```

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`, `n1mgamma` and `n1EMmgamma` 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, $\log(0)=-\text{Inf}$ for log-likelihood and $-\log(0)=\text{Inf}$ 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 `mgamma` `fmgamma` `gammagpd` `gammagpdcon` `fgammagpd` `fgammagpdcon` `normgpd` `fnormgpd` `mgammagpd` `mgammagpdcon` `fmgammagpd` `fmgammagpdcon`: [dgamma](#), [dgamma](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMmgammagpd](#), [nlEMmgammagpdcon](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMmgammagpd](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpd](#), [nlumgammagpdcon](#), [pgammagpd](#), [pgammagpdcon](#), [pmgamma](#), [pmgammagpd](#), [pmgammagpdcon](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpd](#), [proflumgammagpdcon](#), [qgammagpd](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpd](#), [rgammagpdcon](#), [rmgamma](#), [rmgammagpd](#), [rmgammagpdcon](#)

Examples

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

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 = FALSE, ...)

nluEMmgammagpd(pvector, u, tau, mgweight, x, M,
  phiu = TRUE, finitelik = FALSE)
```

Arguments

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
M	number of gamma components in mixture
tau	matrix of posterior probability of being in each component (nxM where n is length(x))
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
u	scalar threshold value
x	vector of sample data
std.err	logical, should standard errors be calculated

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

<code>call:</code>	optim call
<code>x:</code>	data vector x
<code>init:</code>	pvector
<code>fixedu:</code>	fixed threshold, logical
<code>useq:</code>	threshold vector for profile likelihood or scalar for fixed threshold
<code>optim:</code>	complete optim output
<code>mle:</code>	vector of MLE of parameters
<code>cov:</code>	variance-covariance matrix of MLE of parameters
<code>se:</code>	vector of standard errors of MLE of parameters
<code>rate:</code>	phiu to be consistent with <code>evd</code>
<code>nllh:</code>	minimum negative log-likelihood
<code>n:</code>	total sample size
<code>M:</code>	number of gamma components
<code>mgshape:</code>	MLE of gamma shapes
<code>mgscale:</code>	MLE of gamma scales
<code>mgweight:</code>	MLE of gamma weights
<code>u:</code>	threshold (fixed or MLE)
<code>sigmau:</code>	MLE of GPD scale
<code>xi:</code>	MLE of GPD shape
<code>phiu:</code>	MLE of tail fraction (bulk model or parameterised approach)
<code>se.phi:</code>	standard error of MLE of tail fraction
<code>EMresults:</code>	EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation s
<code>posterior:</code>	posterior probabilities

Acknowledgments

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`, `nlgammagpd`, `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 `mgamma` `fmgamma` `gammagpd` `gammagpdcon` `fgammagpd` `fgammagpdcon` `normgpd` `fnormgpd` `mgammagpd` `mgammagpdcon` `fmgammagpd` `fmgammagpdcon`: [dggammagpd](#), [dggammagpdcon](#), [dmgamma](#), [dmggammagpd](#), [dmggammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nLEMmgamma](#), [nLEMmgammagpdcon](#), [nlgammagpd](#), [nlgammagpdcon](#), [nlmgamma](#), [nlmgammagpdcon](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpdcon](#), [pgammagpd](#), [pgammagpdcon](#), [pmgamma](#), [pmggammagpd](#), [pmggammagpdcon](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpdcon](#), [qgammagpd](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpd](#), [rgammagpdcon](#), [rmgamma](#), [rmggammagpd](#), [rmggammagpdcon](#)

Examples

```

## Not run:
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, dmammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi),
  col="red"))
abline(v = fit.noinit$u, col = "red")
with(fit.withinit, lines(xx, dmammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi),
  col="green"))
abline(v = fit.withinit$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, dmammagpd(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, dmammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi), col="red"))
abline(v = fit.withinit$u, col = "red")
with(fitfix, lines(xx, dmammagpd(xx, mgshape, mgscale, mgweight, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$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 = FALSE, ...)

nluEMmgammagpdcon(pvector, u, tau, mgweight, x, M,
  phiu = TRUE, finitelik = FALSE)
```

Arguments

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
M	number of gamma components in mixture
tau	matrix of posterior probability of being in each component (nxM where n is length(x))
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
u	scalar threshold value
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

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
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.phi:    standard error of MLE of tail fraction
EMresults: EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation st
posterior: posterior probabilities

```

Acknowledgments

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 mgamma fmgamma gammagpd gammagpdcon fgammagpd fgammagpdcon normgpd fnormgpd mgammagpd mgammagpdcon fmgammagpd fmgammagpdcon: [dgammagpd](#), [dgammagpdcon](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMgamma](#), [nlEMmgammagpd](#), [nlgammagpd](#), [nlgammagpdcon](#), [nlmgamma](#), [nlmgammagpd](#), [nluEMmgammagpd](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpd](#), [pgammagpd](#), [pgammagpdcon](#), [pmgamma](#), [pmgammagpd](#), [pmgammagpdcon](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpd](#), [qgammagpd](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpd](#), [rgammagpdcon](#), [rmgamma](#), [rmgammagpd](#), [rmgammagpdcon](#)

Examples

```
## Not run:
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, dmammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="red"))
abline(v = fit.noinit$u, col = "red")
with(fit.withinit, lines(xx, dmammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="green"))
abline(v = fit.withinit$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, dmammagpdcon(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, dmammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="red"))
abline(v = fit.withinit$u, col = "red")
with(fitfix, lines(xx, dmammagpdcon(xx, mgshape, mgscale, mgweight, u, xi), col="darkgreen"))
abline(v = fitfix$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, x, phiu = TRUE,
```

```
method = "BFGS", control = list(maxit = 10000),
finitelik = FALSE, ...)
```

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

Arguments

nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
u	scalar threshold value
pvector	vector of initial values of parameters or NULL for default values, see below
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)
x	vector of sample data
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
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 its representations. In the fitting `fnormgpd` and profile likelihood function `proflunormgpd` it is logical:

- default value `phiu=TRUE` - tail fraction specified by normal survivor function $\text{phiu} = 1 - \text{pnorm}(u, \text{nmean}, \text{nsd})$ 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

<code>call:</code>	<code>optim</code> call
<code>x:</code>	data vector <code>x</code>
<code>init:</code>	<code>pvector</code>
<code>fixedu:</code>	fixed threshold, logical
<code>useq:</code>	threshold vector for profile likelihood or scalar for fixed threshold
<code>optim:</code>	complete <code>optim</code> output
<code>mle:</code>	vector of MLE of parameters
<code>cov:</code>	variance-covariance matrix of MLE of parameters
<code>se:</code>	vector of standard errors of MLE of parameters
<code>rate:</code>	<code>phiu</code> to be consistent with <code>evd</code>
<code>nllh:</code>	minimum negative log-likelihood
<code>n:</code>	total sample size
<code>nmean:</code>	MLE of normal mean
<code>nsd:</code>	MLE of normal standard deviation
<code>u:</code>	threshold (fixed or MLE)
<code>sigmau:</code>	MLE of GPD scale
<code>xi:</code>	MLE of GPD shape
<code>phiu:</code>	MLE of tail fraction (bulk model or parameterised approach)
<code>se.phi:</code>	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

Aspects of these functions are deliberately similar in syntax and function with 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 all 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 $\xi_i=0$ 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, $\log(0)=-\text{Inf}$ for log-likelihood and $-\log(0)=\text{Inf}$ 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>

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 normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dgng](#), [dgngcon](#), [dnormgpd](#), [dnormgpdcon](#), [fgng](#), [fgngcon](#), [fnormgpdcon](#), [gng](#), [gngcon](#), [lgng](#), [lgngcon](#), [lnormgpdcon](#), [nlngng](#), [nlngngcon](#), [nlnormgpdcon](#), [nlugng](#), [nlugngcon](#), [nlunormgpdcon](#), [normgpd](#), [normgpdcon](#), [pgng](#), [pgngcon](#), [pnormgpd](#), [pnormgpdcon](#), [proflugng](#), [proflugngcon](#), [proflunormgpdcon](#), [qgng](#), [qgngcon](#), [qnormgpd](#), [qnormgpdcon](#), [rgng](#), [rgngcon](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
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)
```

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 = FALSE, ...)
```

```
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 nnormgpdcon`. 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
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.phi:    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 normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dgng](#), [dgngcon](#), [dnormgpd](#), [dnormgpdcon](#), [fgng](#), [fgngcon](#), [fnormgpd](#), [gng](#), [gngcon](#), [lgng](#), [lgngcon](#), [lnormgpd](#), [nlngng](#), [nlngngcon](#), [nlnormgpd](#), [nlugng](#), [nlugngcon](#), [nlunormgpd](#), [normgpd](#), [normgpdcon](#), [pgng](#), [pgngcon](#), [pnormgpd](#), [pnormgpdcon](#), [proflugng](#), [proflugngcon](#), [proflunormgpd](#), [qgng](#), [qgngcon](#), [qnormgpd](#), [qnormgpdcon](#), [rgng](#), [rgngcon](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
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, sigma, xi), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topleft", c("True Density", "No continuity constraint", "With continuity 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)

```

fweibullgpd	<i>MLE Fitting of Weibull Bulk and GPD Tail Extreme Value Mixture Model</i>
-------------	---

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 = FALSE, ...)

nluweibullgpd(pvector, u, x, phiu = TRUE,
  finitelik = FALSE)

```

Arguments

wshape	scalar Weibull shape (positive)
wscale	scalar Weibull scale (positive)
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
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.

Negative data are ignored.

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
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

phi: MLE of tail fraction (bulk model or parameterised approach)
 se.phi: 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` `weibullgpdcon` `fweibullgpd` `fweibullgpdcon` `normgpd` `fnormgpd`: [fweibullgpdcon](#), [lweibullgpdcon](#), [nlweibullgpdcon](#), [nlweibullgpdcon](#), [profluweibullgpdcon](#)

Examples

```
## Not run:
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 = FALSE, ...)
```

```
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 σ parameter is now specified as function of other parameters, see help for [dweibullgpdcon](#) for details, type `help weibullgpdcon`. Therefore, σ 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
 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.phi: 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 weibullgpdcon fweibullgpd fweibullgpdcon normgpd fnormgpd: [fweibullgpd](#), [lweibullgpd](#), [nluweibullgpd](#), [nlweibullgpd](#), [profluweibullgpd](#)

Examples

```
## Not run:
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 continuity 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)
```

Description

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

Usage

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

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

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

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

Arguments

gshape	gamma shape (positive)
gscale	gamma scale (positive)
phiu	probability of being above threshold $[0, 1]$ or TRUE
x	quantiles
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

Extreme value mixture model combining gamma distribution for the bulk below the threshold and GPD for upper tail.

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

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the gamma and conditional GPD cumulative distribution functions (i.e. `pgamma(x, gshape, 1/gscale)` and `pgpd(x, u, sigmau, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The gamma is defined on the non-negative reals, so the threshold must be positive. Though behaviour at zero depends on the shape (α):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where β 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 mgamma fmgamma gammagpd gammagpdcon fgammagpd fgammagpdcon normgpd fnormgpd mgammagpd mgammagpdcon fmgammagpd fmgammagpdcon: [dgammagpdcon](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMgamma](#), [nlEMgammagpd](#), [nlEMgammagpdcon](#), [nlgammagpd](#), [nlgammagpdcon](#), [nlmgamma](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMgammagpd](#), [nluEMgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpd](#), [nlumgammagpdcon](#), [pgammagpdcon](#), [pmgamma](#), [pmgammagpd](#), [pmgammagpdcon](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpd](#), [proflumgammagpdcon](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpdcon](#), [rmgamma](#), [rmgammagpd](#), [rmgammagpdcon](#)

Examples

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

# three tail behaviours
plot(xx, pgammagpd(xx, gshape = 2), type = "l")
lines(xx, pgammagpd(xx, gshape = 2, xi = 0.3), col = "red")
lines(xx, pgammagpd(xx, gshape = 2, xi = -0.3), col = "blue")
legend("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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the gamma bulk model.

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the gamma and conditional GPD cumulative distribution functions (i.e. pgamma(x, gshape, 1/gscale) and pgpd(x, u, sigmau, xi)) respectively.

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

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

and above the threshold $x > u$:

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

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

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the gamma and conditional GPD density functions (i.e. `dgamma(x, gshape, gscale)` and `dgpd(x, u, sigma_u, 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 (α):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where β 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 mgamma fmgamma gammagpd gammagpdcon fgammagpd fgammagpdcon normgpd fnormgpd mgammagpd mgammagpdcon fmgammagpd fmgammagpdcon: [dgamma](#), [dmgamma](#), [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMgamma](#), [nlEMmgammagpd](#), [nlEMmgammagpdcon](#), [nlgammagpd](#), [nlgammagpdcon](#), [nlmgamma](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMmgammagpd](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpd](#), [nlumgammagpdcon](#), [pgammagpd](#), [pmgamma](#), [pmgammagpd](#), [pmgammagpdcon](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpd](#), [proflumgammagpdcon](#), [qgammagpd](#), [qmgamma](#), [qmgammagpd](#), [qmgammagpdcon](#), [rgammagpd](#), [rmgamma](#), [rmgammagpd](#), [rmgammagpdcon](#)

Examples

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

# three tail behaviours
plot(xx, pgammagpdcon(xx, gshape = 2), type = "l")
lines(xx, pgammagpdcon(xx, gshape = 2, xi = 0.3), col = "red")
lines(xx, pgammagpdcon(xx, gshape = 2, xi = -0.3), col = "blue")
legend("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 λ , lower tail (threshold u_l , GPD scale σ_{aul} and shape ξ_l and tail fraction ϕ_{il}) and upper tail (threshold u_r , GPD scale σ_{aur} and shape ξ_r and tail fraction ϕ_{ir}).

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

u_l 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
x	location to evaluate KDE (single scalar or vector)
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

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 ϕ_{ul} and ϕ_{ur} . 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 ϕ_{ul} defined by the KDE bulk model (phiul=TRUE) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)[1 - G_l(x)].$$

where $H(x)$ is the kernel density estimator cumulative distribution function (i.e. mean(pnorm(x, kerncentres, 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 ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}[1 - G_l(x)].$$

The KDE bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

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.

Note

Unlike all 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>. Based on code by Anna MacDonald produced for MATLAB.

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 kdengpd kdengpdcon gkg gkgcon bckden bckdengpd bckdengpdcon fkden fkdengpd fkdengpdcon fgkg fgkgcon fbckden fbckdengpd fbckdengpdcon: [dgkgcon](#), [gkgcon](#), [pgkgcon](#), [qgkgcon](#), [rgkgcon](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(1000,0,1)
x = rgkg(1000, kerncentres, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkg(xx, kerncentres, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgkg(xx, kerncentres), type = "l")
lines(xx, pgkg(xx, kerncentres, xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgkg(xx, kerncentres, xil = -0.3, xir = -0.3), col = "blue")
legend("topleft", paste("Symmetric xil=xir=", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

# 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

```
dkggcon(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)
```

```
qkggcon(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

<code>x</code>	location to evaluate KDE (single scalar or vector)
<code>kerncentres</code>	kernel centres (typically sample data vector or scalar)
<code>lambda</code>	bandwidth for kernel (as half-width of kernel) or NULL
<code>ul</code>	lower tail threshold
<code>xil</code>	lower tail GPD shape parameter
<code>phiul</code>	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 ϕ_{ul} and ϕ_{ur} . 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 ϕ_{ul} defined by the KDE bulk model (`phiul=TRUE`) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)[1 - G_l(x)].$$

where $H(x)$ is the kernel density estimator cumulative distribution function (i.e. `mean(pnorm(x, kerncentres, 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 ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}[1 - G_l(x)].$$

The KDE bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

The continuity constraint at ur means that:

$$\phi_{ur}g_r(x) = (1 - \phi_{ul} - \phi_{ur})h(u_l)/(H(u_r) - H(u_l)).$$

By rearrangement, the GPD scale parameter σ_{ur} is then:

$$\sigma_{ur} = \phi_{ur}(H(u_r) - H(u_l))/h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

where $h(x)$, $g_l(x)$ and $g_r(x)$ are the 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

$$\sigma_{ur} = [1 - H(u_r)]/h(u_r)$$

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 σ_{ul} is replaced by:

$$\sigma_{ul} = \phi_{ul}(H(u_r) - H(u_l))/h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_{ul} = H(u_l)/h(u_l)$$

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.

Note

Unlike all 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>. Based on code by Anna MacDonald produced for MATLAB.

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 kdengpd kdengpdcon gkg gkgcon bckden bckdengpd bckdengpdcon fkden fkdengpd fkdengpdcon fgkg fgkgcon fbckden fbckdengpd fbckdengpdcon: [dgkg](#), [gkg](#), [pgkg](#), [qgkg](#), [rgkg](#)

Examples

```
## Not run:
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	<i>Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model</i>
-----	---

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

<code>ul</code>	lower tail threshold
<code>sigmaul</code>	lower tail GPD scale parameter (positive)
<code>xil</code>	lower tail GPD shape parameter
<code>phiul</code>	probability of being below lower threshold $[0, 1]$ or TRUE
<code>ur</code>	upper tail threshold
<code>sigmaur</code>	upper tail GPD scale parameter (positive)
<code>xir</code>	upper tail GPD shape parameter

phiur	probability of being above upper threshold [0, 1] or TRUE
nmean	normal mean
nsd	normal standard deviation (positive)
x	quantiles
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 ϕ_{ul} defined by the normal bulk model (phiul=TRUE) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)G_l(x).$$

where $H(x)$ is the normal cumulative distribution function (i.e. pnorm(ur, nmean, nsd)). The $G_l(X)$ is the conditional GPD cumulative distribution function with negated data and threshold, i.e. dgpd(-x, -ul, sigmaul, xil, phiul). The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

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

Above the threshold $x > u_r$ the usual conditional GPD:

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

where $G(X)$.

The cumulative distribution function for the pre-specified tail fractions ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}G_l(x).$$

The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

See [gpd](#) for details of GPD upper tail component, [dnorm](#) for details of normal bulk component and [dnormgpd](#) for normal with GPD extreme value mixture model.

Value

[dgng](#) gives the density, [pgng](#) gives the cumulative distribution function, [qgng](#) gives the quantile function and [rgng](#) gives a random sample.

Note

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

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for [rgng](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in `x`, `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](#) [normgpdcon](#) [gng](#) [gngcon](#) [fnormgpd](#) [fnormgpdcon](#) [fgng](#) [fgngcon](#): [dgngcon](#), [dnormgpd](#), [dnormgpdcon](#), [fgng](#), [fgngcon](#), [fnormgpd](#), [fnormgpdcon](#), [gngcon](#), [lgng](#), [lgngcon](#), [lnormgpd](#), [lnormgpdcon](#), [nlngng](#), [nlngngcon](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugng](#), [nlugngcon](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpd](#), [normgpdcon](#), [pgngcon](#), [pnormgpd](#), [pnormgpdcon](#), [proflugng](#), [proflugngcon](#), [proflunormgpd](#), [proflunormgpdcon](#), [qgngcon](#), [qnormgpd](#), [qnormgpdcon](#), [rgngcon](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgng(1000, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgng(xx, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgng(xx), type = "l")
lines(xx, pgng(xx, xil = 0.3, xir = 0.3), col = "red")
```

```

lines(xx, pngng(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 = rngng(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)

```

```

pngngcon(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)

```

```

rngngcon(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 ϕ_{ul} defined by the normal bulk model (phiul=TRUE) upto the lower threshold $x < u_l$:

$$F(x) = H(u_l)G_l(x).$$

where $H(x)$ is the normal cumulative distribution function (i.e. pnorm(ur, nmean, nsd)). The $G_l(X)$ is the conditional GPD cumulative distribution function with negated data and threshold, i.e. dgp(-x, -ul, sigmaul, xil, phiul). The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

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

Above the threshold $x > u_r$ the usual conditional GPD:

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

where $G(X)$.

The cumulative distribution function for the pre-specified tail fractions ϕ_{ul} and ϕ_{ur} is more complicated. The unconditional GPD is used for the lower tail $x < u_l$:

$$F(x) = \phi_{ul}G_l(x).$$

The normal bulk model between the thresholds $u_l \leq x \leq u_r$ given by:

$$F(x) = \phi_{ul} + (1 - \phi_{ul} - \phi_{ur})(H(x) - H(u_l))/(H(u_r) - H(u_l)).$$

Above the threshold $x > u_r$ the usual conditional GPD:

$$F(x) = (1 - \phi_{ur}) + \phi_{ur}G(x)$$

Notice that these definitions are equivalent when $\phi_{ul} = H(u_l)$ and $\phi_{ur} = 1 - H(u_r)$.

The continuity constraint at u_r means that:

$$\phi_{ur}g_r(x) = (1 - \phi_{ul} - \phi_{ur})h(u_l)/(H(u_r) - H(u_l)).$$

By rearrangement, the GPD scale parameter σ_{ur} is then:

$$\sigma_{ur} = \phi_{ur}(H(u_r) - H(u_l))/h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

where $h(x)$, $g_l(x)$ and $g_r(x)$ are the normal and conditional GPD density functions for lower and upper tail respectively. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_{ur} = [1 - H(u_r)]/h(u_r)$$

.

The continuity constraint at u_l 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 σ_{ul} is replaced by:

$$\sigma_{ul} = \phi_{ul}(H(u_r) - H(u_l))/h(u_l)(1 - \phi_{ul} - \phi_{ur}).$$

In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_{ul} = H(u_l)/h(u_l)$$

.

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 normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dngng](#), [dnormgpd](#), [dnormgpdcon](#), [fgng](#), [fgngcon](#), [fnormgpd](#), [fnormgpdcon](#), [gng](#), [lgng](#), [lgngcon](#), [lnormgpd](#), [lnormgpdcon](#), [nlngng](#), [nlngngcon](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugng](#), [nlugngcon](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpd](#), [normgpdcon](#), [pgng](#), [pnormgpd](#), [pnormgpdcon](#), [proflugng](#), [proflugngcon](#), [proflunormgpd](#), [proflunormgpdcon](#), [qngng](#), [qnormgpd](#), [qnormgpdcon](#), [rgng](#), [rnormgpd](#), [rnormgpdcon](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rgngcon(1000, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgngcon(xx, phiul = 0.15, phiur = 0.15))

# three tail behaviours
plot(xx, pgngcon(xx), type = "l")
lines(xx, pgngcon(xx, xil = 0.3, xir = 0.3), col = "red")
lines(xx, pgngcon(xx, xil = -0.3, xir = -0.3), col = "blue")
legend("topleft", paste("Symmetric xil=xir=", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

x = rgngcon(1000, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgngcon(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2))

plot(xx, dgngcon(xx, xil = -0.3, phiul = 0.2, xir = 0.3, phiur = 0.2), type = "l", ylim = c(0, 0.4))
lines(xx, dgngcon(xx, xil = -0.3, phiul = 0.3, xir = 0.3, phiur = 0.3), col = "red")
lines(xx, dgngcon(xx, xil = -0.3, phiul = TRUE, xir = 0.3, phiur = TRUE), col = "blue")
legend("topleft", c("phiul = phiur = 0.2", "phiul = phiur = 0.3", "Bulk Tail Fraction"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

gpd

*Generalised Pareto Distribution (GPD)***Description**

Density, cumulative distribution function, quantile function and random number generation for the 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

<code>x</code>	quantiles
<code>q</code>	quantiles
<code>p</code>	cumulative probabilities
<code>n</code>	sample size (positive integer)
<code>u</code>	threshold
<code>sigmau</code>	scale parameter (positive)
<code>xi</code>	shape parameter
<code>phiu</code>	probability of being above threshold $[0, 1]$
<code>log</code>	logical, if TRUE then log density
<code>lower.tail</code>	logical, if FALSE then upper tail probabilities

Details

The GPD with parameters scale σ_u and shape ξ 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 ξ , $x > u$ and $\sigma_u > 0$. Further, $[1 + \xi(x - u)/\sigma_u] > 0$ which for $\xi < 0$ implies $u < x \leq u - \sigma_u/\xi$. In the special case of $\xi = 0$ 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 multiplying 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

Aspects of these functions are deliberately similar in syntax and function with the commonly used functions in the `ismev` and `evd` packages for which their author's contributions are gratefully acknowledged.

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, $p > 1 - \phi_u$, it will give conditional quantiles above threshold, but when below the threshold, $p \leq 1 - \phi_u$, 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

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: [fgpd](#), [lgpd](#), [nlgpd](#)

Examples

```
par(mfrow=c(2,2))
x = rgpd(1000) # simulate sample from GPD
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgpdx(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, dgpdx(xx, u = 0, sigmau = 1, xi = 0), lwd = 2)
lines(xx, dgpdx(xx, u = 0.5, phiu = 1 - pexp(0.5)), col = "red", lwd = 2)
lines(xx, dgpdx(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)
```

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

nmean	normal mean
nsd	normal standard deviation (positive)
x	quantiles
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 ϕu 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 $r = 1 + \text{pnorm}(u, \text{mean} = \text{nmean}, \text{sd} = \text{nsd})$, 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 $\text{sigma}u$ 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. $\text{dnorm}(u, \text{nmean}, \text{nsd})$ and $\text{dgp}(u, u, \text{sigma}u, \text{xi})$). 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 $\text{sigma}u$ 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

[dhp](#) gives the density, [php](#) gives the cumulative distribution function, [qhp](#) gives the quantile function and [rhp](#) 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 [rhp](#) 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 [rhp](#) 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` `hpdcon` `fhpd` `fhpdcon` `normgpd` `normgpdcon` `fnormgpd` `fnormgpdcon`: [dhpcon](#), [hpdcon](#), [phpcon](#), [qhpdcon](#), [rhpcon](#)

Examples

```
## Not run:
par(mfrow = c(2, 2))
xx = seq(-5, 20, 0.01)
f1 = dhp(xx, nmean = 0, nsd = 1, xi = 0.4)
plot(xx, f1, type = "l")
abline(v = 0.4942921)

# three tail behaviours
plot(xx, php(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

<code>nmean</code>	normal mean
<code>nsd</code>	normal standard deviation (positive)
<code>x</code>	quantiles
<code>u</code>	threshold
<code>xi</code>	shape parameter
<code>log</code>	logical, if TRUE then log density
<code>q</code>	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 ϕu 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 $r = 1 + \text{pnorm}(u, \text{mean} = \text{nmean}, \text{sd} = \text{nsd})$, 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. $\text{dnorm}(u, \text{nmean}, \text{nsd})$ and $\text{dgp}(u, u, \text{sigmau}, \text{xi})$).

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 [rhpdcn](#) 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 [rhpdcn](#) 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 [rhpdcn](#) 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 hpdcon fhpd fhpdcon normgpd normgpdcon fnormgpd fnormgpdcon: [dhpd](#), [hpd](#), [phpd](#), [qhpd](#), [rhpd](#)

Examples

```
## Not run:
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 = rhpdcn(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)

```

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)

Details

Internal functions not designed to be used directly. No error checking of the inputs is carried out, so user must be know what they are doing. They are undocumented, but are made visible to the user.

Mostly, these are used in the kernel density estimation functions.

Author(s)

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

See Also

[density](#), [kden](#) and [bckden](#).

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
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)
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")

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:

$$\hat{f}(x) = \frac{1}{n} \sum_{j=1}^n K\left(\frac{x - x_j}{\lambda}\right)$$

where $K(\cdot)$ 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 is 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 $c(\max(\text{kerncentre}) - \text{lambda}, \max(\text{kerncentre}) + \text{lambda})$, or $c(\max(\text{kerncentre}) - 5*\text{lambda}, \max(\text{kerncentre}) + 5*\text{lambda})$ for normal kernel. Outside of this range the quantiles are set to $-\text{Inf}$ for lower tail and Inf 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.

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.

Note

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

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

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

Missing (NA) and Not-a-Number (NaN) values in `x`, `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>. Based on code by Anna MacDonald produced for MATLAB.

References

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

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

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 [kdengpd](#) [kdengpdcon](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fkdengpd](#) [fkdengpdcon](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [bckden](#), [bckdengpd](#), [bckdengpdcon](#), [dbckden](#), [dbckdengpd](#), [dbckdengpdcon](#), [dkdengpd](#), [dkdengpdcon](#), [fbckden](#), [fkden](#), [kdengpd](#), [kdengpdcon](#), [lbckden](#), [lkden](#), [nlbckden](#), [nlkden](#), [pbckden](#), [pbckdengpd](#), [pbckdengpdcon](#), [pkdengpd](#), [pkdengpdcon](#), [qbckden](#), [qbckdengpd](#), [qbckdengpdcon](#), [qkdengpd](#), [qkdengpdcon](#), [rbckden](#), [rbckdengpd](#), [rbckdengpdcon](#), [rkdengpd](#), [rkdengpdcon](#)

Examples

```
## Not run:
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

phiu	probability of being above threshold [0, 1] or TRUE
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)
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter

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 ϕ_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 ϕ_u defined by the upper tail fraction of the kernel 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 KDE and conditional GPD cumulative distribution functions respectively.

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

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

and above the threshold $x > u$:

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

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

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.

Note

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

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

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

Missing (NA) and Not-a-Number (NaN) values in x , 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>. Based on code by Anna MacDonald produced for MATLAB.

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` `kdengpd` `kdengpdcon` `bckden` `bckdengpd` `bckdengpdcon` `fkden` `fkdengpd` `fkdengpdcon` `fbckden` `fbckdengpd` `fbckdengpdcon`: `bckden`, `bckdengpd`, `bckdengpdcon`, `dbckden`, `dbckdengpd`, `dbckdengpdcon`, `dkden`, `dkdengpdcon`, `fbckden`, `fkden`, `kden`, `kdengpdcon`, `lbckden`, `lkden`, `nlbckden`, `nlkden`, `pbckden`, `pbckdengpd`, `pbckdengpdcon`, `pkden`, `pkdengpdcon`, `qbckden`, `qbckdengpd`, `qbckdengpdcon`, `qkden`, `qkdengpdcon`, `rbckden`, `rbckdengpd`, `rbckdengpdcon`, `rkden`, `rkdengpdcon`

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(500, 0, 1)
```

```

xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpd(xx, kerncentres, u = 1.2, sigma = 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, sigma = 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, sigma = 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 λ , threshold u GPD shape ξ and tail fraction ϕ .

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 ϕ_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 ϕ_u defined by the upper tail fraction of the kernel 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 KDE and conditional GPD cumulative distribution functions respectively.

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

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

and above the threshold $x > u$:

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

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

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the 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.

Note

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

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

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

Missing (NA) and Not-a-Number (NaN) values in `x`, `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>. Based on code by Anna MacDonald produced for MATLAB.

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 [kdengpd](#) [kdengpdcon](#) [bckden](#) [bckdengpd](#) [bckdengpdcon](#) [fkden](#) [fkdengpd](#) [fkdengpdcon](#) [fbckden](#) [fbckdengpd](#) [fbckdengpdcon](#): [bckden](#), [bckdengpd](#), [bckdengpdcon](#), [dbckden](#), [dbckdengpd](#), [dbckdengpdcon](#), [dkden](#), [dkdengpd](#), [fbckden](#), [fkden](#), [kden](#), [kdengpd](#), [lbckden](#), [lkden](#), [nlbckden](#), [nlkden](#), [pbckden](#), [pbckdengpd](#), [pbckdengpdcon](#), [pkden](#), [pkdengpd](#), [qbckden](#), [qbckdengpd](#), [qbckdengpdcon](#), [qkden](#), [qkdengpd](#), [rbckden](#), [rbckdengpd](#), [rbckdengpdcon](#), [rkden](#), [rkdengpd](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
kerncentres=rnorm(500, 0, 1)
xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpdcon(xx, kerncentres, u = 1.2, xi = 0.1))

plot(xx, pkdengpdcon(xx, kerncentres), type = "l")
lines(xx, pkdengpdcon(xx, kerncentres, xi = 0.3), col = "red")
lines(xx, pkdengpdcon(xx, kerncentres, xi = -0.3), col = "blue")
legend("topleft", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1, cex = 0.5)

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)
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
z	standardised location put into kernel $z = (x - \text{kerncentres}) / \text{lambda}$
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 $(x - \text{kerncentres}) / \text{lambda}$ and kerncentre=0. A warning 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 $z = (x - \text{kerncentres})/\lambda$.

Value

code `kd*` 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://http://en.wikipedia.org/wiki/Kernel_\(statistics\)](http://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: [check.kbw](#), [check.kernel](#), [check.kinputs](#), [ka0](#), [ka1](#), [ka2](#), [kbw](#), [kfun](#), [klambda](#)

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

truncpoint	upper endpoint as standardised location x/λ
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)
kernel	kernel name (default = "gaussian")
allownull	logical, where TRUE permits NULL values

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. $\text{truncpoint} = x/\lambda$. The exception is the normal kernel which has unbounded support so the $[-5 * \lambda, p]$ where λ is the standard deviation bandwidth. There is a function for each partial moment of degree (0, 1, 2):

- $\text{ka0} - \int_{-1}^p K * (z) dz$
- $\text{ka1} - \int_{-1}^p u K * (z) dz$
- $\text{ka2} - \int_{-1}^p u^2 K * (z) dz$

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 ∞ 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://http://en.wikipedia.org/wiki/Kernel_\(statistics\)](http://http://en.wikipedia.org/wiki/Kernel_(statistics))

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

See Also

`kernels`, `density`, `kden` and `bckden`.

Other kernels: `kdbiweight`, `kdcosine`, `kdepanechnikov`, `kdgaussian`, `kdoptcosine`, `kdparzen`, `kdtriangular`, `kdtricube`, `kdtriweight`, `kduniform`, `kdz`, `kernels`, `kpbiweight`, `kpcosine`, `kpepanechnikov`, `kpgaussian`, `kpoptcosine`, `kpparzen`, `kptriangular`, `kptricube`, `kptriweight`, `kpuniform`, `kpz`

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 $\ln\text{mean}$ and standard deviation $\ln\text{sd}$, threshold u GPD scale sigmau and shape xi and tail fraction phiu .

Usage

```
dlognormgpd(x, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), sigmau = lnstd, xi = 0,
  phiu = TRUE, log = FALSE)
```

```
plognormgpd(q, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), sigmau = lnstd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)
```

```
qlognormgpd(p, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), sigmau = lnstd, xi = 0,
  phiu = TRUE, lower.tail = TRUE)
```

```
rlognormgpd(n = 1, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), sigmau = lnstd, xi = 0,
  phiu = TRUE)
```

Arguments

<code>lnmean</code>	mean on log scale
<code>lnstd</code>	standard deviation on log scale (positive)
<code>phiu</code>	probability of being above threshold $[0, 1]$ or TRUE
<code>x</code>	quantiles
<code>u</code>	threshold
<code>sigmau</code>	scale parameter (positive)
<code>xi</code>	shape parameter
<code>log</code>	logical, if TRUE then log density
<code>q</code>	quantiles
<code>lower.tail</code>	logical, if FALSE then upper tail probabilities
<code>p</code>	cumulative probabilities
<code>n</code>	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 ϕ_u permitting a parameterised value for the tail fraction ϕ_u . Alternatively, when $\phi_u=TRUE$ the tail fraction is estimated as the tail fraction from the log-normal bulk model.

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the log-normal and conditional GPD cumulative distribution functions (i.e. `plnorm(x, lmean, lnsd)` and `pgpd(x, u, sigmau, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The 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 lognormgpdcon normgpd normgpdcon flognormgpd flognormgpdcon fnormgpd fnormgpdcon: [dlognormgpdcon](#), [lognormgpdcon](#), [plognormgpdcon](#), [qlognormgpdcon](#), [rlognormgpdcon](#)

Examples

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

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

x = rlognormgpd(1000, u = 2, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpd(xx, u = 2, phiu = 0.2))

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

## End(Not run)
```

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 $\ln\text{mean}$ and standard deviation $\ln\text{sd}$, threshold u GPD shape ξ and tail fraction ϕ_u .

Usage

```
dlognormgpdcon(x, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), xi = 0, phiu = TRUE,
  log = FALSE)

plognormgpdcon(q, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

qlnormgpdcon(p, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), xi = 0, phiu = TRUE,
  lower.tail = TRUE)

rlognormgpdcon(n = 1, lnmean = 0, lnstd = 1,
  u = qlnorm(0.9, lnmean, lnstd), xi = 0, phiu = TRUE)
```

Arguments

<code>x</code>	quantiles
<code>lnmean</code>	mean on log scale
<code>lnstd</code>	standard deviation on log scale (positive)
<code>u</code>	threshold
<code>xi</code>	shape parameter
<code>phiu</code>	probability of being above threshold $[0, 1]$ or TRUE
<code>log</code>	logical, if TRUE then log density
<code>q</code>	quantiles
<code>lower.tail</code>	logical, if FALSE then upper tail probabilities
<code>p</code>	cumulative probabilities
<code>n</code>	sample size (positive integer)

Details

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

The user can pre-specify ϕ_u permitting a parameterised value for the tail fraction ϕ_u . Alternatively, when $\phi_u = \text{TRUE}$ the tail fraction is estimated as the tail fraction from the log-normal bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the log-normal bulk model ($\phi_u = \text{TRUE}$), upto the threshold $0 < x \leq u$, given by:

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the log-normal and conditional GPD cumulative distribution functions (i.e. `plnorm(x, lnmean, lnsd)` and `pgpd(x, u, sigmau, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The 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 `dgpdc(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 lognormgpdcon normgpd normgpdcon flognormgpd flognormgpdcon fnormgpd fnormgpdcon: [dlognormgpd](#), [lognormgpd](#), [plognormgpd](#), [qlognormgpd](#), [rlognormgpd](#)

Examples

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

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

x = rlognormgpdcon(1000, u = 2, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpdcon(xx, u = 2, phiu = 0.2))

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

## End(Not run)
```

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

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)
x	quantiles
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 $M \geq 1$ 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 (α):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where β 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.

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 [fmgamma](#) [gammagpd](#) [gammagpdcon](#) [fgammagpd](#) [fgammagpdcon](#) [normgpd](#) [fnormgpd](#) [mgammagpd](#) [mgammagpdcon](#) [fmgammagpd](#) [fmgammagpdcon](#): [dgammagpd](#), [dgammagpdcon](#), [dmammagpd](#), [dmammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgammagpd](#), [mgammagpdcon](#), [nlEMgamma](#), [nlEMmgammagpd](#), [nlEMmgammagpdcon](#), [nlammagpd](#), [nlammagpdcon](#), [nlmgamma](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMmgammagpd](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumammagpd](#), [nlumammagpdcon](#), [pgammagpd](#), [pgammagpdcon](#), [pmammagpd](#), [pmammagpdcon](#), [proflumammagpd](#), [proflumammagpdcon](#), [proflumammagpd](#), [proflumammagpdcon](#), [qammagpd](#), [qammagpdcon](#), [qmammagpd](#), [qmammagpdcon](#), [rgammagpd](#), [rgammagpdcon](#), [rmammagpd](#), [rmammagpdcon](#)

Examples

```
## Not run:
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

<code>mgshape</code>	mgamma shape (positive) as list or vector
<code>mgscale</code>	mgamma scale (positive) as list or vector
<code>mgweight</code>	mgamma weights (positive) as list or vector (NULL for equi-weighted)
<code>x</code>	quantiles
<code>u</code>	threshold

sigma	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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the mixture of gammas bulk model.

Suppose there are $M \geq 1$ 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 ϕ_u defined by the upper tail fraction of the mixture of gammas bulk model (phiu=TRUE), upto the threshold $0 < x \leq u$, given by:

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the mixture of gammas and conditional GPD cumulative distribution functions.

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

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

and above the threshold $x > u$:

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

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

The gamma is defined on the non-negative reals, so the threshold must be positive. Though behaviour at zero depends on the shape (α):

- $f(0+) = \infty$ for $0 < \alpha < 1$;

- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where β 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.

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 mgamma [fmgamma](#) [gammagpd](#) [gammagpdcon](#) [fgammagpd](#) [fgammagpdcon](#) [normgpd](#) [fnormgpd](#) [mgammagpd](#) [mgammagpdcon](#) [fmgammagpd](#) [fmgammagpdcon](#): [dmgammagpd](#), [dmgammagpdcon](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpdcon](#), [nLEMgamma](#), [nLEMgammagpd](#), [nLEMgammagpdcon](#), [nlgammagpd](#), [nlgammagpdcon](#),

nlmgamma, nlmgammagpd, nlmgammagpdcon, nluEMmgammagpd, nluEMmgammagpdcon, nlugammagpd, nlugammagpdcon, nlumgammagpd, nlumgammagpdcon, pgammagpd, pgammagpdcon, pmgamma, pmgammagpdcon, proflugammagpd, proflugammagpdcon, proflumgammagpd, proflumgammagpdcon, qgammagpd, qgammagpdcon, qmgamma, qmgammagpdcon, rgammagpd, rgammagpdcon, rmgamma, rmgammagpdcon

Examples

```
## Not run:
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, dmammagpd(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

```
dmammagpdcon(x, mgshape = 1, mgscale = 1,
  mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0,
  phiu = TRUE, log = FALSE)

pmammagpdcon(q, mgshape = 1, mgscale = 1,
  mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0,
  phiu = TRUE, lower.tail = TRUE)

qmammagpdcon(p, mgshape = 1, mgscale = 1,
  mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), xi = 0,
  phiu = TRUE, lower.tail = TRUE)

rmammagpdcon(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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the mixture of gammas bulk model.

Suppose there are $M \geq 1$ 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 ϕ_u defined by the upper tail fraction of the mixture of gammas bulk model (phiu=TRUE), upto the threshold $0 < x \leq u$, given by:

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the mixture of gammas and conditional GPD cumulative distribution functions.

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

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

and above the threshold $x > u$:

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

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

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the 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 (α):

- $f(0+) = \infty$ for $0 < \alpha < 1$;
- $f(0+) = 1/\beta$ for $\alpha = 1$ (exponential);
- $f(0+) = 0$ for $\alpha > 1$;

where β 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.

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 mgamma fmgamma gammagpd gammagpdcon fgammagpd fgammagpdcon normgpd fnormgpd mgammagpd mgammagpdcon fmgammagpd fmgammagpdcon: [dgammagpd](#), [dgammagpdcon](#), [dmgamma](#), [dmgammagpd](#), [fgammagpd](#), [fgammagpdcon](#), [fmgamma](#), [fmgammagpd](#), [fmgammagpdcon](#), [gammagpd](#), [gammagpdcon](#), [lgammagpd](#), [lgammagpdcon](#), [lmgamma](#), [lmgammagpd](#), [lmgammagpdcon](#), [mgamma](#), [mgammagpd](#), [nlEMmgamma](#), [nlEMmgammagpd](#), [nlEMmgammagpdcon](#), [nlgammagpd](#), [nlgammagpdcon](#), [nlmgamma](#), [nlmgammagpd](#), [nlmgammagpdcon](#), [nluEMmgammagpd](#), [nluEMmgammagpdcon](#), [nlugammagpd](#), [nlugammagpdcon](#), [nlumgammagpd](#), [nlumgammagpdcon](#), [pgammagpd](#), [pgammagpdcon](#), [pmgamma](#), [pmgammagpd](#), [proflugammagpd](#), [proflugammagpdcon](#), [proflumgammagpd](#), [proflumgammagpdcon](#), [qgammagpd](#), [qgammagpdcon](#), [qmgamma](#), [qmgammagpd](#), [rgammagpd](#), [rgammagpdcon](#), [rmgamma](#), [rmgammagpd](#)

Examples

```
## Not run:
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

<code>data</code>	vector of sample data
<code>tlim</code>	vector of (lower, upper) limits of range of threshold to plot MRL, or NULL to use default values
<code>nt</code>	number of thresholds for which to evaluate MRL
<code>alpha</code>	logical, significance level (0, 1)
<code>p.or.n</code>	logical, should tail fraction (FALSE) or number of exceedances (TRUE) be given on upper x-axis
<code>ylim</code>	y-axis limits or NULL
<code>legend.loc</code>	location of legend (see legend) or NULL for no legend
<code>try.thresh</code>	vector of threshold to fit GPD using MLE and show theoretical MRL
<code>main</code>	title of plot
<code>xlab</code>	x-axis label
<code>ylab</code>	y-axis label
<code>...</code>	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 $\text{mean}(x[x > v]) - v$.

Symmetric central limit theorem based confidence intervals are provided for all mean excesses, provided there are at least 5 exceedances. The sampling density for the MRL is shown by a greyscale image, where lighter greys indicate low density.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The GPD is fitted to the excesses using maximum likelihood estimation. The estimated parameters are used to plot the linear function for all higher thresholds using a solid line. The threshold should set as low as possible, so a dashed line is shown below the pre-chosen threshold. If the MRL is similar to the dashed line then a lower threshold may be chosen.

If no threshold limits are provided `tlim` = NULL then the lowest threshold is set to be just below the median data point and the maximum threshold is set to the 6th largest datapoint.

The range of permitted thresholds is just below the minimum datapoint and the second largest value. If there are less unique values of data within the threshold range than the number of threshold evaluations requested, then instead of a sequence of thresholds they will be set to each unique datapoint, i.e. the MRL will only be evaluated where there is data.

The missing (NA and NaN) and non-finite values are ignored.

The lower x-axis is the threshold and an upper axis either gives the number of exceedances (`p.or.n` = FALSE) or proportion of excess (`p.or.n` = TRUE). Note that unlike the `gpd` related functions the missing values are ignored, so do not add to the lower tail fraction. But ignoring the missing values is consistent with all the other mixture model functions.

Value

`mrlplot` gives the mean residual life plot. It also returns a matrix containing columns of the threshold, number of exceedances, mean excess, standard deviation of excesses and $100(1 - \alpha)\%$ confidence interval. The standard deviation and confidence interval are NA for less than 5 exceedances.

Note

If the user specifies the threshold range, the thresholds above the second largest are dropped. A warning message is given if any thresholds have at most 5 exceedances, in which case the confidence interval is not calculated as it is unreliable due to small sample. If there are less than 10 exceedances of the minimum threshold then the function will stop.

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

Author(s)

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

References

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

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

See Also

`gpd` and `mrlplot` from `evd` library

Examples

```
x = rnorm(1000)
mrlplot(x)
mrlplot(x, tlim = c(0, 2.2))
mrlplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))
mrlplot(x, tlim = c(0, 3), try.thresh = c(0.5, 1, 1.5))
```

normgpd

Normal Bulk and GPD Tail Extreme Value Mixture Model

Description

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

Usage

```
dnormgpd(x, nmean = 0, nsd = 1,
         u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
         phiu = TRUE, log = FALSE)
```

```
pnormgpd(q, nmean = 0, nsd = 1,
         u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
         phiu = TRUE, lower.tail = TRUE)
```

```
qnormgpd(p, nmean = 0, nsd = 1,
         u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
         phiu = TRUE, lower.tail = TRUE)
```

```
rnormgpd(n = 1, nmean = 0, nsd = 1,
         u = qnorm(0.9, nmean, nsd), sigmau = nsd, xi = 0,
         phiu = TRUE)
```

Arguments

nmean	normal mean
nsd	normal standard deviation (positive)
phiu	probability of being above threshold $[0, 1]$ or TRUE
x	quantiles
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

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail.

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

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. pnorm(x, nmean, nsd) and pgpd(x, u, sigmau, xi)) respectively.

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

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

and above the threshold $x > u$:

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

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

See [gpd](#) for details of GPD upper tail component and [dnorm](#) for details of normal bulk component.

Value

[dnormgpd](#) gives the density, [pnormgpd](#) gives the cumulative distribution function, [qnormgpd](#) gives the quantile function and [rnormgpd](#) gives a random sample.

Note

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

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for [rnormgpd](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in `x`, `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](#) [normgpdcon](#) [gng](#) [gngcon](#) [fnormgpd](#) [fnormgpdcon](#) [fgng](#) [fgngcon](#): [dgng](#), [dgngcon](#), [dnormgpdcon](#), [fgng](#), [fgngcon](#), [fnormgpd](#), [fnormgpdcon](#), [gng](#), [gngcon](#), [lgng](#), [lgngcon](#), [lnormgpd](#), [lnormgpdcon](#), [nlngng](#), [nlngngcon](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugng](#), [nlugngcon](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpdcon](#), [pgng](#), [pgngcon](#), [pnormgpdcon](#), [proflugng](#), [proflugngcon](#), [proflunormgpd](#), [proflunormgpdcon](#), [qgng](#), [qgngcon](#), [qnormgpdcon](#), [rgng](#), [rgngcon](#), [rnormgpdcon](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnormgpd(1000)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx))

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

x = rnormgpd(1000, phiu = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx, phiu = 0.2))

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

## End(Not run)
```

normgpdcon

Normal Bulk and GPD Tail Extreme Value Mixture Model with 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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the normal bulk model.

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x, nmean, nsd)` and `pgpd(x, u, sigmau, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The continuity constraint means that $(1 - \phi_u)h(u)/H(u) = \phi_u g(u)$ where $h(x)$ and $g(x)$ are the normal and conditional GPD density functions (i.e. `dnorm(x, nmean, nsd)` and `dgpdc(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 `mmean` 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 Gaman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

[gpd](#) and [dnorm](#)

Other normgpd normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dgng](#), [dgngcon](#), [dnormgpd](#), [fgng](#), [fgngcon](#), [fnormgpd](#), [fnormgpdcon](#), [gng](#), [gngcon](#), [lgng](#), [lgngcon](#), [lnormgpd](#), [lnormgpdcon](#), [nlngng](#), [nlngngcon](#), [nlnormgpd](#), [nlnormgpdcon](#), [nlugng](#), [nlugngcon](#), [nlunormgpd](#), [nlunormgpdcon](#), [normgpd](#), [pgng](#), [pgngcon](#), [pnormgpd](#), [proflugng](#), [proflugngcon](#), [proflunormgpd](#), [proflunormgpdcon](#), [qgng](#), [qgngcon](#), [qnormgpd](#), [rgng](#), [rgngcon](#), [rnormgpd](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rnormgpdcon(1000)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpdcon(xx))
```

```

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

x = rnormgpdcon(1000, phiu = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpdcon(xx, phiu = 0.2))

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

## End(Not run)

```

tcplot

Parameter Threshold Stability Plots

Description

Plots the sample mean residual life (MRL) plot.

Usage

```

tcplot(data, tlim = NULL, nt = min(100, length(data)),
      p.or.n = FALSE, alpha = 0.05, ylim.xi = NULL,
      ylim.sigtau = NULL, legend.loc = NULL,
      try.thresh = quantile(data, 0.9, na.rm = TRUE), ...)

```

```

tshapeplot(data, tlim = NULL,
  nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Shape Threshold Stability Plot",
  xlab = "Threshold u", ylab = "Shape Parameter", ...)

```

```

tscaleplot(data, tlim = NULL,
  nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Modified Scale Threshold Stability Plot",
  xlab = "Threshold u",
  ylab = "Modified Scale Parameter", ...)

```

Arguments

`ylim.xi` y-axis limits for shape parameter or NULL

<code>ylim.sigtau</code>	y-axis limits for scale parameter or NULL
<code>data</code>	vector of sample data
<code>tlim</code>	vector of (lower, upper) limits of range of threshold to plot MRL, or NULL to use default values
<code>nt</code>	number of thresholds for which to evaluate MRL
<code>p.or.n</code>	logical, should tail fraction (FALSE) or number of exceedances (TRUE) be given on upper x-axis
<code>alpha</code>	logical, significance level (0, 1)
<code>legend.loc</code>	location of legend (see legend) or NULL for no legend
<code>try.thresh</code>	vector of threshold to fit GPD using MLE and show theoretical MRL
<code>...</code>	further arguments to be passed to the plotting functions
<code>ylim</code>	y-axis limits or NULL
<code>main</code>	title of plot
<code>xlab</code>	x-axis label
<code>ylab</code>	y-axis label

Details

The MLE of the (modified) GPD scale and shape (ξ) parameters are plotted against as range of possible threshold. Known as the threshold stability plots (Coles, 2001). The modified scale parameter is $\sigma_u - u\xi$. If the GPD is a suitable model for a threshold u then for all higher threshold $v > u$ it will also be suitable, with the shape and modified scale being constant.

In practice there is sample uncertainty in the parameter estimates, which must be taken into account when choosing a threshold.

The usual asymptotic Wald confidence intervals are shown based on the observed information matrix to measure this 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 plot as a horizontal line which is solid above this threshold where the parameter from smaller tail fraction should be the same if the GPD is a good model (upto sample uncertainty). The threshold should always be chosen to be as low as possible to reduce sample uncertainty. Therefore, below the pre-chosen threshold, where the GPD should not be a good model, the line is dashed and the parameter estimates should now deviate from the dashed line (otherwise a lower threshold could be used). If no threshold limits are provided `tlim = NULL` then the lowest threshold is set to be just below the median data point and the maximum threshold is set to the 11th largest datapoint. This is a slightly lower order statistic compared to that used in the MRL plot `mrlplot` function to account for the fact the maximum likelihood estimation is likely to be very unreliable with 10 or fewer datapoints.

The range of permitted thresholds is just below the minimum datapoint and the second largest value. If there are less unique values of data within the threshold range than the number of threshold evaluations 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 deviation and $100(1-\alpha)\%$ Wald confidence interval. Where the observed information matrix is not obtainable the standard deviation and confidence intervals are NA. For the `tscaleplot` the modified scale quantities are also provided. `tcplot` produces both plots on one graph and outputs a merged dataframe of results.

Note

If the user specifies the threshold range, the thresholds above the sixth largest are dropped. A warning message is given if any thresholds have at most 10 exceedances, in which case the maximum likelihood estimation is unreliable. If there are less than 10 exceedances of the minimum threshold then the function will stop.

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

`mr1plot` 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

<i>wshape</i>	Weibull shape (positive)
<i>wscale</i>	Weibull scale (positive)
<i>x</i>	quantiles
<i>u</i>	threshold
<i>sigmau</i>	scale parameter (positive)
<i>xi</i>	shape parameter
<i>phiu</i>	probability of being above threshold [0, 1] or TRUE
<i>log</i>	logical, if TRUE then log density
<i>q</i>	quantiles
<i>lower.tail</i>	logical, if FALSE then upper tail probabilities
<i>p</i>	cumulative probabilities
<i>n</i>	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 ϕ_u . Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the weibull bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the Weibull bulk model (`phiu=TRUE`), upto the threshold $0 < x \leq u$, 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, sigma, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The 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 weibullgpdcon fweibullgpd fweibullgpdcon: [dweibullgpdcon](#), [pweibullgpdcon](#), [qweibullgpdcon](#), [rweibullgpdcon](#), [weibullgpdcon](#)

Examples

```
## Not run:
par(mfrow=c(2,2))
x = rweibullgpd(1000)
xx = seq(-1, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx))

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

x = rweibullgpd(1000, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx, phiu = 0.2))

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

## End(Not run)
```

weibullgpdcon

Weibull Bulk and GPD Tail Extreme Value Mixture Model with 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),
  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
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)
sigmau	scale parameter (positive)

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 ϕ_u . Alternatively, when phiu=TRUE the tail fraction is estimated as the tail fraction from the weibull bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the Weibull bulk model (phiu=TRUE), upto the threshold $0 < x \leq u$, 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, sigma, xi)`) respectively.

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

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

and above the threshold $x > u$:

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

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

The 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 `dgp(x, u, sigma, 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.

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 weibullgpdcon fweibullgpd fweibullgpdcon: [dweibullgpd](#), [pweibullgpd](#), [qweibullgpd](#), [rweibullgpd](#), [weibullgpd](#)

Examples

```
## Not run:
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

- bckden, [3](#), [6](#), [11](#), [15](#), [33](#), [89](#), [164](#), [167](#), [170](#), [174](#), [177](#), [179](#)
- bckdengpd, [7](#), [8](#), [10](#), [15](#), [33](#), [89](#), [167](#), [170](#), [174](#)
- bckdengpdcon, [7](#), [11](#), [12](#), [15](#), [33](#), [89](#), [167](#), [170](#), [174](#)
- bckdenxbeta1 (internal), [163](#)
- bckdenxbeta2 (internal), [163](#)
- bckdenxcopula (internal), [163](#)
- bckdenxcutnorm (internal), [163](#)
- bckdenxgamma1 (internal), [163](#)
- bckdenxgamma2 (internal), [163](#)
- bckdenxreflect (internal), [163](#)
- bckdenxrenorm (internal), [163](#)
- bckdenxsimple (internal), [163](#)
- betagpd, [16](#), [21](#)
- betagpdcon, [18](#), [19](#)
- bw.nrd0, [7](#), [11](#), [15](#), [33](#), [37](#), [42](#), [62](#), [66](#), [67](#), [89](#), [93](#), [97](#), [142](#), [143](#), [146](#), [147](#), [166](#), [167](#), [169](#), [170](#), [173](#), [174](#)
- check.bcmethode (checking), [22](#)
- check.control (checking), [22](#)
- check.inputn, [23](#)
- check.inputn (checking), [22](#)
- check.kbw, [177](#)
- check.kbw (kfun), [177](#)
- check.kernel, [177](#), [179](#)
- check.kernel (kfun), [177](#)
- check.kinputs, [177](#), [179](#)
- check.kinputs (kfun), [177](#)
- check.klambda, [179](#)
- check.logic (checking), [22](#)
- check.n (checking), [22](#)
- check.nn (checking), [22](#)
- check.nparam (checking), [22](#)
- check.offset (checking), [22](#)
- check.optim (checking), [22](#)
- check.param, [22](#)
- check.param (checking), [22](#)
- check.phiu (checking), [22](#)
- check.posparam, [22](#)
- check.posparam (checking), [22](#)
- check.prob (checking), [22](#)
- check.quant (checking), [22](#)
- check.text (checking), [22](#)
- checking, [22](#)
- condmixt, [3](#), [81](#), [84](#), [159](#), [162](#)
- dbckden, [5](#), [10](#), [11](#), [14](#), [15](#), [33](#), [36](#), [41](#), [89](#), [167](#), [170](#), [174](#)
- dbckden (bckden), [3](#)
- dbckdengpd, [7](#), [10](#), [15](#), [33](#), [89](#), [167](#), [170](#), [174](#)
- dbckdengpd (bckdengpd), [8](#)
- dbckdengpdcon, [7](#), [11](#), [14](#), [33](#), [40](#), [89](#), [167](#), [170](#), [174](#)
- dbckdengpdcon (bckdengpdcon), [12](#)
- dbeta, [18](#), [20](#), [21](#), [45](#), [48](#)
- dbetagpd, [18](#), [21](#)
- dbetagpd (betagpd), [16](#)
- dbetagpdcon, [18](#), [21](#), [47](#)
- dbetagpdcon (betagpdcon), [19](#)
- dcauchy, [26](#)
- ddwm, [25](#)
- ddwm (dwm), [24](#)
- density, [4](#), [7](#), [9](#), [11](#), [13](#), [15](#), [28](#), [30](#), [33](#), [35](#), [37](#), [40](#), [42](#), [60](#), [62](#), [65–67](#), [86](#), [89](#), [92](#), [93](#), [96](#), [97](#), [141–143](#), [145–147](#), [164–167](#), [169](#), [170](#), [172–174](#), [176–179](#)
- densplot, [28](#)
- densplot (evmix.diag), [26](#)
- dgamma, [54](#), [58](#), [108](#), [112](#), [117](#), [135](#), [136](#), [138](#), [139](#), [187](#), [190](#), [194](#)
- dgamma, [54](#), [58](#), [108](#), [112](#), [117](#), [135](#), [139](#), [187](#), [190](#), [194](#)
- dgamma (gamma), [133](#)
- dgamma, [54](#), [56](#), [58](#), [108](#), [112](#), [117](#), [136](#), [138](#), [187](#), [190](#), [194](#)
- dgamma (gamma), [136](#)
- dgkg, [142](#), [147](#)
- dgkg (gkg), [140](#)
- dgkgcon, [65](#), [143](#), [146](#)
- dgkgcon (gkgcon), [144](#)
- dgng, [71](#), [75](#), [123](#), [126](#), [150](#), [153](#), [154](#), [198](#), [201](#)
- dgng (gng), [148](#)
- dgngcon, [71](#), [73](#), [75](#), [123](#), [126](#), [150](#), [153](#), [198](#), [201](#)
- dgngcon (gngcon), [151](#)
- dgpd, [78](#), [156](#)

- gammamixEM, 108
 gkg, 140, 142, 147
 gkgcon, 143, 144, 146
 gng, 71, 75, 123, 126, 148, 154, 198, 201
 gngcon, 71, 75, 123, 126, 150, 151, 198, 201
 gpd, 10, 11, 14, 15, 17, 18, 20, 21, 26, 37, 42, 45, 48, 52, 54, 58, 62, 67, 71, 75, 78, 81, 84, 93, 97, 101, 104, 112, 117, 123, 126, 129, 133, 135, 136, 138, 139, 142, 146, 149, 150, 153, 154, 155, 158, 159, 161, 162, 169, 173, 181, 182, 184, 185, 187, 190, 194, 196, 198, 200, 201, 206, 207, 209, 210
 gpd.diag, 28
 hpareto, 159, 162
 hpareto.fit, 81, 84
 hpareto.negloglike, 81, 84
 hparetomixt, 159, 162
 hpd, 157, 162
 hpdcon, 159, 160
 integrate, 6, 10, 14
 internal, 163
 ismev, 3, 121, 156
 jitter, 30, 32–34, 39, 60, 64, 86, 88, 89, 92, 95
 ka0, 177, 179
 ka0 (kfun), 177
 ka1, 177, 179
 ka1 (kfun), 177
 ka2, 177, 179
 ka2 (kfun), 177
 kbw, 177, 179
 kbw (kfun), 177
 kd*, 177
 kdbiweight, 179
 kdbiweight (kernels), 174
 kdcosine, 179
 kdcosine (kernels), 174
 kden, 4, 7, 11, 15, 33, 89, 164, 164, 166, 170, 174, 177, 179
 kdengpd, 7, 11, 15, 33, 89, 167, 168, 170, 174
 kdengpdcon, 7, 11, 15, 33, 89, 167, 170, 171, 173
 kdenx (internal), 163
 kdepanechnikov, 179
 kdepanechnikov (kernels), 174
 kdgaussian, 179
 kdgaussian (kernels), 174
 kdoptcosine, 179
 kdoptcosine (kernels), 174
 kdparzen, 179
 kdparzen (kernels), 174
 kdtriangular, 179
 kdtriangular (kernels), 174
 kdtricube, 179
 kdtricube (kernels), 174
 kdtriweight, 179
 kdtriweight (kernels), 174
 kduniform, 179
 kduniform (kernels), 174
 kdz, 177, 179
 kdz (kernels), 174
 kernels, 4, 5, 7, 9–11, 13–15, 30, 33, 35, 37, 40, 42, 60, 62, 65, 67, 86, 89, 92, 93, 96, 97, 141, 143, 145, 147, 165, 167, 169, 170, 172, 174, 174, 179
 kfun, 7, 11, 15, 33, 37, 42, 62, 67, 89, 93, 97, 143, 147, 167, 170, 174, 177, 177
 klambda, 89, 177, 179
 klambda (kfun), 177
 kp*, 177
 kpbiweight, 179
 kpbiweight (kernels), 174
 kpcosine, 179
 kpcosine (kernels), 174
 kpepanechnikov, 179
 kpepanechnikov (kernels), 174
 kpgaussian, 179
 kpgaussian (kernels), 174
 kproptcosine, 179
 kproptcosine (kernels), 174
 kpparzen, 179
 kpparzen (kernels), 174
 kptriangular, 179
 kptriangular (kernels), 174
 kptricube, 179
 kptricube (kernels), 174
 kptriweight, 179
 kptriweight (kernels), 174
 kpu, 178
 kpuniform, 179
 kpuniform (kernels), 174
 kpz, 177, 179
 kpz (kernels), 174
 ks, 5, 7, 11, 15, 37, 42, 62, 67, 93, 97, 143, 147, 166, 167, 170, 174
 lbckden, 7, 11, 15, 30, 31, 89, 167, 170, 174
 lbckden (fbckden), 29
 lbckdengpd, 36, 42, 93, 97
 lbckdengpd (fbckdengpd), 34

- lbckdengpdcon, [37](#), [40](#), [93](#), [97](#)
 lbckdengpdcon (fbckdengpdcon), [38](#)
 lbetagpd, [44](#), [48](#)
 lbetagpd (fbetagpd), [43](#)
 lbetagpdcon, [45](#), [47](#)
 lbetagpdcon (fbetagpdcon), [46](#)
 ldwm, [50](#), [51](#)
 ldwm (fdwm), [49](#)
 legend, [195](#), [203](#)
 lgammagpd, [53](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#),
[187](#), [190](#), [194](#)
 lgammagpd (fgammagpd), [52](#)
 lgammagpdcon, [54](#), [57](#), [108](#), [112](#), [117](#), [136](#),
[139](#), [187](#), [190](#), [194](#)
 lgammagpdcon (fgammagpdcon), [55](#)
 lgkg, [61](#), [67](#)
 lgkg (fgkg), [59](#)
 lgkgcon, [62](#), [65](#)
 lgkgcon (fgkgcon), [63](#)
 lgng, [70](#), [75](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
 lgng (fgng), [68](#)
 lgngcon, [71](#), [74](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
 lgngcon (fgngcon), [72](#)
 lgpd, [77](#), [78](#), [157](#)
 lgpd (fgpd), [76](#)
 lhp, [80](#), [81](#), [84](#)
 lhp (fhpd), [79](#)
 lhpcon, [81](#), [83](#)
 lhpcon (fhpdcon), [82](#)
 lkden, [7](#), [11](#), [15](#), [33](#), [87](#), [88](#), [167](#), [170](#), [174](#)
 lkden (fkden), [85](#)
 lkden, [37](#), [42](#), [92](#), [97](#)
 lkden (fkden), [90](#)
 lkden, [37](#), [42](#), [93](#), [96](#)
 lkden (fkden), [94](#)
 llognormgpd, [100](#), [104](#)
 llognormgpd (flognormgpd), [98](#)
 llognormgpdcon, [101](#), [103](#)
 llognormgpdcon (flognormgpdcon), [102](#)
 lmgamma, [54](#), [58](#), [106](#), [107](#), [112](#), [117](#), [136](#), [139](#),
[187](#), [190](#), [194](#)
 lmgamma (fmgamma), [105](#)
 lmgammagpd, [54](#), [58](#), [108](#), [110–112](#), [117](#), [136](#),
[139](#), [187](#), [190](#), [194](#)
 lmgammagpd (fmgammagpd), [108](#)
 lmgammagpdcon, [54](#), [58](#), [108](#), [112](#), [115–117](#),
[136](#), [139](#), [187](#), [190](#), [194](#)
 lmgammagpdcon (fmgammagpdcon), [113](#)
 lnormgpd, [71](#), [75](#), [87](#), [106](#), [120–122](#), [126](#), [150](#),
[154](#), [198](#), [201](#)
 lnormgpd (fnormgpd), [118](#)
 lnormgpdcon, [71](#), [75](#), [123](#), [125](#), [150](#), [154](#), [198](#),
[201](#)
 lnormgpdcon (fnormgpdcon), [123](#)
 lognormgpd, [180](#), [185](#)
 lognormgpdcon, [182](#), [182](#)
 lpgd, [78](#)
 lweibullgpd, [128](#), [133](#)
 lweibullgpd (fweibullgpd), [127](#)
 lweibullgpdcon, [129](#), [131](#)
 lweibullgpdcon (fweibullgpdcon), [130](#)
 mgamma, [54](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#), [185](#),
[190](#), [194](#)
 mgammagpd, [54](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#),
[187](#), [188](#), [194](#)
 mgammagpdcon, [54](#), [58](#), [108](#), [112](#), [117](#), [136](#),
[139](#), [187](#), [190](#), [191](#)
 mrlplot, [194](#), [196](#), [203](#), [204](#)
 nlbckden, [7](#), [11](#), [15](#), [30](#), [31](#), [89](#), [167](#), [170](#), [174](#)
 nlbckden (fbckden), [29](#)
 nlbckdengpd, [36](#), [42](#), [93](#), [97](#)
 nlbckdengpd (fbckdengpd), [34](#)
 nlbckdengpdcon, [37](#), [40](#), [93](#), [97](#)
 nlbckdengpdcon (fbckdengpdcon), [38](#)
 nlbetagpd, [44](#), [48](#)
 nlbetagpd (fbetagpd), [43](#)
 nlbetagpdcon, [45](#), [47](#)
 nlbetagpdcon (fbetagpdcon), [46](#)
 nldwm, [50](#), [51](#)
 nldwm (fdwm), [49](#)
 nlEMmgamma, [54](#), [58](#), [106](#), [107](#), [112](#), [117](#), [136](#),
[139](#), [187](#), [190](#), [194](#)
 nlEMmgamma (fmgamma), [105](#)
 nlEMmgammagpd, [54](#), [58](#), [108](#), [110–112](#), [117](#),
[136](#), [139](#), [187](#), [190](#), [194](#)
 nlEMmgammagpd (fmgammagpd), [108](#)
 nlEMmgammagpdcon, [54](#), [58](#), [108](#), [112](#),
[115–117](#), [136](#), [139](#), [187](#), [190](#), [194](#)
 nlEMmgammagpdcon (fmgammagpdcon), [113](#)
 nlgammagpd, [53](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#),
[187](#), [190](#), [194](#)
 nlgammagpd (fgammagpd), [52](#)
 nlgammagpdcon, [54](#), [57](#), [108](#), [112](#), [117](#), [136](#),
[139](#), [187](#), [190](#), [194](#)
 nlgammagpdcon (fgammagpdcon), [55](#)
 nlgkg, [61](#), [67](#)
 nlgkg (fgkg), [59](#)
 nlgkgcon, [62](#), [65](#)
 nlgkgcon (fgkgcon), [63](#)
 nlng, [70](#), [75](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
 nlng (fgng), [68](#)
 nlngcon, [71](#), [74](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
 nlngcon (fgngcon), [72](#)

- nlgpd, *77, 78, 157*
 nlgpd (fgpd), *76*
 nlhpd, *80, 81, 84*
 nlhpd (fhpd), *79*
 nlhpdcon, *81, 83*
 nlhpdcon (fhpdcon), *82*
 nlkden, *7, 11, 15, 33, 87, 88, 167, 170, 174*
 nlkden (fkden), *85*
 nlkdengpd, *37, 42, 92, 97*
 nlkdengpd (fkdenpd), *90*
 nlkdengpdcon, *37, 42, 93, 96*
 nlkdengpdcon (fkdenpdcon), *94*
 nllognormgpd, *100, 104*
 nllognormgpd (flognormgpd), *98*
 nllognormgpdcon, *101, 103*
 nllognormgpdcon (flognormgpdcon), *102*
 nlmgamma, *54, 58, 106, 107, 112, 117, 136, 139, 187, 191, 194*
 nlmgamma (fmgamma), *105*
 nlmgammagpd, *55, 58, 108, 110–112, 117, 136, 139, 187, 191, 194*
 nlmgammagpd (fmgammagpd), *108*
 nlmgammagpdcon, *55, 58, 108, 112, 115–117, 136, 139, 187, 191, 194*
 nlmgammagpdcon (fmgammagpdcon), *113*
 nlnormgpd, *71, 75, 120–122, 126, 150, 154, 198, 201*
 nlnormgpd (fnormgpd), *118*
 nlnormgpdcon, *71, 75, 123, 125, 150, 154, 198, 201*
 nlnormgpdcon (fnormgpdcon), *123*
 nlnornlkdenmgpd, *87*
 nlpgd, *78*
 nlubckdengpd, *36, 42, 93, 97*
 nlubckdengpd (fbckdengpd), *34*
 nlubckdengpdcon, *37, 40, 93, 97*
 nlubckdengpdcon (fbckdengpdcon), *38*
 nlubetapgd, *44, 48*
 nlubetapgd (fbetapgd), *43*
 nlubetapgdcon, *45, 47*
 nlubetapgdcon (fbetapgdcon), *46*
 nluEMgammagpd, *55, 58, 108, 110–112, 117, 136, 139, 187, 191, 194*
 nluEMgammagpd (fmgammagpd), *108*
 nluEMgammagpdcon, *55, 58, 108, 112, 115–117, 136, 139, 187, 191, 194*
 nluEMgammagpdcon (fmgammagpdcon), *113*
 nlugammagpd, *53, 58, 108, 112, 117, 136, 139, 187, 191, 194*
 nlugammagpd (fgammagpd), *52*
 nlugammagpdcon, *55, 57, 108, 112, 117, 136, 139, 187, 191, 194*
 nlugammagpdcon (fgammagpdcon), *55*
 nlugkg, *61, 67*
 nlugkg (fgkg), *59*
 nlugkgcon, *62, 65*
 nlugkgcon (fgkgcon), *63*
 nlugng, *70, 75, 123, 126, 150, 154, 198, 201*
 nlugng (fgng), *68*
 nlugngcon, *71, 74, 123, 126, 150, 154, 198, 201*
 nlugngcon (fgngcon), *72*
 nluhpdcon, *81, 83*
 nluhpdcon (fhpdcon), *82*
 nlukdengpd, *37, 42, 92, 97*
 nlukdengpd (fkdenpd), *90*
 nlukdengpdcon, *37, 42, 93, 96*
 nlukdengpdcon (fkdenpdcon), *94*
 nlulognormgpd, *100, 104*
 nlulognormgpd (flognormgpd), *98*
 nlulognormgpdcon, *101, 103*
 nlulognormgpdcon (flognormgpdcon), *102*
 nlumgammagpd, *55, 58, 108, 111, 112, 117, 136, 139, 187, 191, 194*
 nlumgammagpd (fmgammagpd), *108*
 nlumgammagpdcon, *55, 58, 108, 112, 116, 117, 136, 139, 187, 191, 194*
 nlumgammagpdcon (fmgammagpdcon), *113*
 nlunormgpd, *71, 75, 120–122, 126, 150, 154, 198, 201*
 nlunormgpd (fnormgpd), *118*
 nlunormgpdcon, *71, 75, 123, 125, 150, 154, 198, 201*
 nlunormgpdcon (fnormgpdcon), *123*
 nluweibullgpd, *128, 133*
 nluweibullgpd (fweibullgpd), *127*
 nluweibullgpdcon, *129, 131*
 nluweibullgpdcon (fweibullgpdcon), *130*
 nlweibullgpd, *128, 133*
 nlweibullgpd (fweibullgpd), *127*
 nlweibullgpdcon, *129, 131*
 nlweibullgpdcon (fweibullgpdcon), *130*
 normgpd, *71, 75, 123, 126, 150, 154, 196, 201*
 normgpdcon, *71, 75, 123, 126, 150, 154, 198, 199*
 optim, *23, 30, 31, 35, 39, 43, 44, 47, 50, 53, 56, 60, 64, 69, 73, 76, 77, 79, 80, 83, 86, 88, 91, 95, 99, 102, 105, 106, 110, 114, 119, 121, 124, 128, 131*
 pbckden, *5, 11, 15, 33, 89, 167, 170, 174*
 pbckden (bckden), *3*
 pbckdengpd, *7, 10, 15, 33, 89, 167, 170, 174*
 pbckdengpd (bckdengpd), *8*

- pbckdengpdcon, *7, 11, 14, 33, 89, 167, 170, 174*
- pbckdengpdcon (bckdengpdcon), *12*
- pbckdenxbeta1 (internal), *163*
- pbckdenxbeta2 (internal), *163*
- pbckdenxcopula (internal), *163*
- pbckdenxcutnorm (internal), *163*
- pbckdenxgamma1 (internal), *163*
- pbckdenxgamma2 (internal), *163*
- pbckdenxlog (internal), *163*
- pbckdenxnn (internal), *163*
- pbckdenxreflect (internal), *163*
- pbckdenxrenorm (internal), *163*
- pbckdenxsimple (internal), *163*
- pbetagpd, *18, 21*
- pbetagpd (betagpd), *16*
- pbetagpdcon, *18, 21*
- pbetagpdcon (betagpdcon), *19*
- pdwm, *25*
- pdwm (dwm), *24*
- pgammagpd, *55, 58, 108, 112, 117, 135, 139, 187, 191, 194*
- pgammagpd (gammagpd), *133*
- pgammagpdcon, *55, 58, 108, 112, 117, 136, 138, 187, 191, 194*
- pgammagpdcon (gammagpdcon), *136*
- pgkg, *142, 147*
- pgkg (gkg), *140*
- pgkgcon, *143, 146*
- pgkgcon (gkgcon), *144*
- pgng, *71, 75, 123, 126, 150, 154, 198, 201*
- pgng (gng), *148*
- pgngcon, *71, 75, 123, 126, 150, 153, 198, 201*
- pgngcon (gngcon), *151*
- pgpd, *78, 156*
- pgpd (gpd), *155*
- phpd, *159, 162*
- phpd (hpd), *157*
- phpdcon, *159, 161*
- phpdcon (hpdcon), *160*
- pkden, *7, 11, 15, 33, 89, 166, 170, 174*
- pkden (kden), *164*
- pkdengpd, *7, 11, 15, 33, 89, 167, 169, 174*
- pkdengpd (kdengpd), *168*
- pkdengpdcon, *7, 11, 15, 33, 89, 167, 170, 173*
- pkdengpdcon (kdengpdcon), *171*
- pkdenx (internal), *163*
- plognormgpd, *181, 185*
- plognormgpd (lognormgpd), *180*
- plognormgpdcon, *182, 184*
- plognormgpdcon (lognormgpdcon), *182*
- plot, *27*
- plot.uvevd, *27, 28*
- pmgamma, *55, 58, 108, 112, 117, 136, 139, 187, 191, 194*
- pmgamma (mgamma), *185*
- pmgammagpd, *55, 58, 108, 112, 117, 136, 139, 187, 190, 194*
- pmgammagpd (mgammagpd), *188*
- pmgammagpdcon, *55, 58, 108, 112, 117, 136, 139, 187, 191, 193*
- pmgammagpdcon (mgammagpdcon), *191*
- pnorm, *178*
- pnormgpd, *71, 75, 123, 126, 150, 154, 198, 201*
- pnormgpd (normgpd), *196*
- pnormgpdcon, *71, 75, 123, 126, 150, 154, 198, 201*
- pnormgpdcon (normgpdcon), *199*
- pplot, *28*
- pplot (evmix.diag), *26*
- ppoints, *27, 28*
- proflubckdengpd, *36, 42, 93, 97*
- proflubckdengpd (fbckdengpd), *34*
- proflubckdengpdcon, *37, 40, 93, 97*
- proflubckdengpdcon (fbckdengpdcon), *38*
- proflubetagpd, *44, 48*
- proflubetagpd (fbetagpd), *43*
- proflubetagpdcon, *45, 47*
- proflubetagpdcon (fbetagpdcon), *46*
- proflugammagpd, *53, 58, 108, 112, 117, 136, 139, 187, 191, 194*
- proflugammagpd (fgammagpd), *52*
- proflugammagpdcon, *55, 57, 108, 112, 117, 136, 139, 187, 191, 194*
- proflugammagpdcon (fgammagpdcon), *55*
- proflugkg, *60, 61, 67*
- proflugkg (fgkg), *59*
- proflugkgcon, *62, 65*
- proflugkgcon (fgkgcon), *63*
- proflugng, *69, 70, 75, 123, 126, 150, 154, 198, 201*
- proflugng (fgng), *68*
- proflugngcon, *71, 74, 123, 126, 150, 154, 198, 201*
- proflugngcon (fgngcon), *72*
- profluhpdcon, *81, 83*
- profluhpdcon (fhpdcon), *82*
- proflukdengpd, *37, 42, 92, 97*
- proflukdengpd (fkdengpd), *90*
- proflukdengpdcon, *37, 42, 93, 96*
- proflukdengpdcon (fkdengpdcon), *94*
- proflulognormgpd, *100, 104*
- proflulognormgpd (flognormgpd), *98*
- proflulognormgpdcon, *101, 103*

- proflunormmgpdcon (flognormmgpdcon),
 102
 proflumgammagpd, 55, 58, 108, 110, 111, 117,
 136, 139, 187, 191, 194
 proflumgammagpd (fmgammagpd), 108
 proflumgammagpdcon, 55, 58, 108, 112, 115,
 116, 136, 139, 187, 191, 194
 proflumgammagpdcon (fmgammagpdcon), 113
 proflunormgpd, 71, 75, 120, 121, 126, 150,
 154, 198, 201
 proflunormgpd (fnormgpd), 118
 proflunormgpdcon, 71, 75, 123, 125, 150,
 154, 198, 201
 proflunormgpdcon (fnormgpdcon), 123
 profluweibullgpd, 128, 133
 profluweibullgpd (fweibullgpd), 127
 profluweibullgpdcon, 129, 131
 profluweibullgpdcon (fweibullgpdcon),
 130
 pweibullgpd, 206, 210
 pweibullgpd (weibullgpd), 205
 pweibullgpdcon, 207, 209
 pweibullgpdcon (weibullgpdcon), 207
 pxb (internal), 163

 qbckden, 4, 5, 11, 15, 33, 89, 167, 170, 174
 qbckden (bckden), 3
 qbckdengpd, 7, 10, 15, 33, 89, 167, 170, 174
 qbckdengpd (bckdengpd), 8
 qbckdengpdcon, 7, 11, 14, 33, 89, 167, 170,
 174
 qbckdengpdcon (bckdengpdcon), 12
 qbetagpd, 18, 21
 qbetagpd (betagpd), 16
 qbetagpdcon, 18, 21
 qbetagpdcon (betagpdcon), 19
 qdwm, 25
 qdwm (dwm), 24
 qgammagpd, 55, 58, 108, 112, 117, 135, 139,
 187, 191, 194
 qgammagpd (gammagpd), 133
 qgammagpdcon, 55, 58, 108, 112, 117, 136,
 138, 187, 191, 194
 qgammagpdcon (gammagpdcon), 136
 qgkg, 142, 147
 qgkg (gkg), 140
 qgkgcon, 143, 146
 qgkgcon (gkgcon), 144
 qgng, 71, 75, 123, 126, 150, 154, 198, 201
 qgng (gng), 148
 qgngcon, 71, 75, 123, 126, 150, 153, 198, 201
 qgngcon (gngcon), 151
 qgpd, 78, 156

 qgpd (gpd), 155
 qhpd, 159, 162
 qhpd (hpd), 157
 qhpdcon, 159, 161
 qhpdcon (hpdcon), 160
 qkde, 5, 166
 qkden, 7, 11, 15, 33, 89, 166, 170, 174
 qkden (kden), 164
 qkdengpd, 7, 11, 15, 33, 89, 167, 169, 174
 qkdengpd (kdengpd), 168
 qkdengpdcon, 7, 11, 15, 33, 89, 167, 170, 173
 qkdengpdcon (kdengpdcon), 171
 qlognormgpd, 181, 185
 qlognormgpd (lognormgpd), 180
 qlognormgpdcon, 182, 184
 qlognormgpdcon (lognormgpdcon), 182
 qmgamma, 55, 58, 108, 112, 117, 136, 139, 187,
 191, 194
 qmgamma (mgamma), 185
 qmammagpd, 55, 58, 108, 112, 117, 136, 139,
 187, 190, 194
 qmammagpd (mammagpd), 188
 qmammagpdcon, 55, 58, 108, 112, 117, 136,
 139, 187, 191, 193
 qmammagpdcon (mammagpdcon), 191
 qnormgpd, 71, 75, 123, 126, 150, 154, 198, 201
 qnormgpd (normgpd), 196
 qnormgpdcon, 71, 75, 123, 126, 150, 154, 198,
 201
 qnormgpdcon (normgpdcon), 199
 qplot, 28
 qplot (evmix.diag), 26
 qweibullgpd, 206, 210
 qweibullgpd (weibullgpd), 205
 qweibullgpdcon, 207, 209
 qweibullgpdcon (weibullgpdcon), 207

 rbckden, 5, 6, 11, 15, 33, 89, 167, 170, 174
 rbckden (bckden), 3
 rbckdengpd, 7, 10, 15, 33, 89, 167, 170, 174
 rbckdengpd (bckdengpd), 8
 rbckdengpdcon, 7, 11, 14, 15, 33, 89, 167,
 170, 174
 rbckdengpdcon (bckdengpdcon), 12
 rbetagpd, 18, 21
 rbetagpd (betagpd), 16
 rbetagpdcon, 18, 21
 rbetagpdcon (betagpdcon), 19
 rdwm, 25
 rdwm (dwm), 24
 rgammagpd, 55, 58, 108, 112, 117, 135, 139,
 187, 191, 194
 rgammagpd (gammagpd), 133

- rgammagpdcon, [55](#), [58](#), [108](#), [112](#), [117](#), [136](#),
[138](#), [187](#), [191](#), [194](#)
- rgammagpdcon (gammagpdcon), [136](#)
- rgkg, [142](#), [147](#)
- rgkg (gkg), [140](#)
- rgkgcon, [143](#), [146](#)
- rgkgcon (gkgcon), [144](#)
- rng, [71](#), [75](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
- rng (gng), [148](#)
- rngcon, [71](#), [75](#), [123](#), [126](#), [150](#), [153](#), [198](#), [201](#)
- rngcon (gngcon), [151](#)
- rgpd, [78](#), [156](#)
- rgpd (gpd), [155](#)
- rhp, [159](#), [162](#)
- rhp (hpd), [157](#)
- rhpcon, [159](#), [161](#)
- rhpcon (hpdcon), [160](#)
- rkden, [7](#), [11](#), [15](#), [33](#), [89](#), [166](#), [170](#), [174](#)
- rkden (kden), [164](#)
- rkdenpd, [7](#), [11](#), [15](#), [33](#), [89](#), [167](#), [169](#), [170](#), [174](#)
- rkdenpd (kdenpd), [168](#)
- rkdenpdcon, [7](#), [11](#), [15](#), [33](#), [89](#), [167](#), [170](#), [173](#)
- rkdenpdcon (kdenpdcon), [171](#)
- rlognormgpd, [181](#), [185](#)
- rlognormgpd (lognormgpd), [180](#)
- rlognormgpdcon, [182](#), [184](#)
- rlognormgpdcon (lognormgpdcon), [182](#)
- rplot, [28](#)
- rplot (evmix.diag), [26](#)
- rgamma, [55](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#), [187](#),
[191](#), [194](#)
- rgamma (mgamma), [185](#)
- rgammagpd, [55](#), [58](#), [108](#), [112](#), [117](#), [136](#), [139](#),
[187](#), [190](#), [194](#)
- rgammagpd (mgammagpd), [188](#)
- rgammagpdcon, [55](#), [58](#), [108](#), [112](#), [117](#), [136](#),
[139](#), [187](#), [191](#), [193](#)
- rgammagpdcon (mgammagpdcon), [191](#)
- rnormgpd, [71](#), [75](#), [123](#), [126](#), [150](#), [154](#), [198](#), [201](#)
- rnormgpd (normgpd), [196](#)
- rnormgpdcon, [71](#), [75](#), [123](#), [126](#), [150](#), [154](#), [198](#),
[201](#)
- rnormgpdcon (normgpdcon), [199](#)
- rweibullgpd, [206](#), [210](#)
- rweibullgpd (weibullgpd), [205](#)
- rweibullgpdcon, [207](#), [209](#)
- rweibullgpdcon (weibullgpdcon), [207](#)

- sapply, [120](#), [176](#)
- splinefun, [5](#), [166](#)

- tcplot, [202](#), [204](#)
- tscaleplot, [204](#)
- tscaleplot (tcplot), [202](#)
- tshapeplot, [204](#)
- tshapeplot (tcplot), [202](#)

- weibullgpd, [205](#), [210](#)
- weibullgpdcon, [207](#), [207](#)