

R documentation

of all in ‘man’

May 27, 2015

R topics documented:

evmix-package	2
bckden	4
bckdengpd	9
bckdengpdcon	14
betagpd	18
betagpdcon	21
checking	24
dwm	26
evmix.diag	29
fbckden	31
fbckdengpd	36
fbckdengpdcon	41
fbetagpd	46
fbetagpdcon	49
fdwm	52
fgammagpd	55
fgammagpdcon	59
fgkg	63
fgkgcon	68
fgng	73
fgngcon	78
fgpd	82
fhpd	85
fhpdcon	88
fitmgng	92
fitmnormgpd	96
fitmweibullgpd	99
fkden	102
fkdengpd	107
fkdengpdcon	111
flognormgpd	115
flognormgpdcon	119
fingamma	122
fingammagpd	126
fingammagpdcon	132

fnormgpd	137
fnormgpdcon	143
fpsden	146
fpsdengpd	150
fweibullgpd	154
fweibullgpdcon	157
gammagpd	161
gammagpdcon	164
gkg	168
gkgcon	172
gng	176
gngcon	180
gpd	184
hillplot	186
hpd	189
hpdcon	192
internal	194
itmngng	196
itmnormgpd	200
itmweibullgpd	203
kden	206
kdengpd	210
kdengpdcon	213
kernels	217
kfun	220
lognormgpd	223
lognormgpdcon	226
mgamma	229
mgammagpd	232
mgammagpdcon	236
mrlplot	240
normgpd	242
normgpdcon	245
pickandsplot	249
psden	251
psdengpd	254
tcplot	257
weibullgpd	260
weibullgpdcon	263
Index	266

 evmix-package

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

Description

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

Details

```

Package:    evmix
Type:       Package
Version:    0.2-6
Date:       2015-05-27
License:    GPL-3
LazyLoad:   yes

```

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

Kernel density estimation including various boundary corrected kernel density estimation methods and a wide choice of kernels, with cross-validation likelihood based bandwidth estimators are included.

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

Author(s)

Carl Scarrott and Yang Hu, 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

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)

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 $c(0, \max(\text{kerncentre}) + 5 \times \text{lambda})$ 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 $[0, \text{xmax}]$ 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 $[0, \text{xmax}]$ rather than $[0, 1]$. In this case the bandwidth is defined as $\lambda = 1 - \rho^2$, so the bandwidth is limited to $(0, 1)$.

Warning

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

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

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

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

Author(s)

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

References

- http://en.wikipedia.org/wiki/Kernel_density_estimation
- [http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](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: [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#); [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#); [dkdengpdcon](#), [dkdengpdcon](#), [dkdengpdcon](#), [dkdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [rkdengpdcon](#), [rkdengpdcon](#), [rkdengpdcon](#), [rkdengpdcon](#); [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [rkdengpd](#), [rkdengpd](#), [rkdengpd](#), [rkdengpd](#), [rkdengpd](#), [rkdengpd](#); [dkden](#), [dkden](#), [dkden](#), [dkden](#), [dkden](#), [kden](#), [kden](#), [kden](#), [kden](#), [kden](#), [kden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [rkden](#), [rkden](#), [rkden](#), [rkden](#), [rkden](#).

rkden; fbckden, fbckden, fbckden, lbckden, lbckden, lbckden, nlbckden, nlbckden, nlbckden;
fkden, fkden, fkden, lkden, lkden, lkden, nlkden, nlkden, nlkden

Examples

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

n=100
x = rgamma(n, shape = 1, scale = 2)
xx = seq(-0.5, 12, 0.01)
plot(xx, dgamma(xx, shape = 1, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 1), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "Simple boundary correction",
"KDE using density function", "Boundary Corrected Kernels"),
lty = c(1, 1, 2, 1), lwd = c(1, 2, 2, 1), col = c("black", "red", "green", "blue"))

n=100
x = rbeta(n, shape1 = 3, shape2 = 2)*5
xx = seq(-0.5, 5.5, 0.01)
plot(xx, dbeta(xx/5, shape1 = 3, shape2 = 2)/5, type = "l", ylim = c(0, 0.8))
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.1, bcmethod = "beta2", proper = TRUE, xmax = 5),
lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "Modified Beta KDE Using evmix",
"KDE using density function"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "green"))

# Demonstrate renormalisation (usually small difference)
n=1000
x = rgamma(n, shape = 1, scale = 2)
xx = seq(-0.5, 15, 0.01)
plot(xx, dgamma(xx, shape = 1, scale = 2), type = "l")
rug(x)
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = TRUE),
lwd = 2, col = "purple")
lines(xx, dbckden(xx, x, lambda = 0.5, bcmethod = "simple", proper = FALSE),
lwd = 2, col = "red", lty = 2)
legend("topright", c("True Density", "Simple BC with renormalisation",
"Simple BC without renormalisation"),
lty = 1, lwd = c(1, 2, 2), col = c("black", "purple", "red"))

## End(Not run)
```

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)), sigmau = sqrt(6 *
  var(kerncentres))/pi, xi = 0, phiu = TRUE, bw = NULL,
  kernel = "gaussian", bcmethod = "simple", proper = TRUE, nn = "jf96",
  offset = NULL, xmax = NULL, log = FALSE)
```

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

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

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

Arguments

<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>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>bw</code>	bandwidth for kernel (as standard deviations of kernel) or NULL
<code>kernel</code>	kernel name (default = "gaussian")
<code>bcmethod</code>	boundary correction method
<code>proper</code>	logical, whether density is renormalised to integrate to unity (where needed)
<code>nn</code>	non-negativity correction method (simple boundary correction only)
<code>offset</code>	offset added to kernel centres (logtrans only) or NULL
<code>xmax</code>	upper bound on support (copula and beta kernels only) or NULL

log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

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

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

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

The user can pre-specify 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 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 ($\phi_u=TRUE$), 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.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

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

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

Author(s)

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

References

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

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

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

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

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

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

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

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

See Also

[gpd](#), [kernels](#), [kfun](#), [density](#), [bw.nrd0](#) and [dkde](#) in [ks](#) package.

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

Examples

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

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

plot(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", type = "l")
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, xi = 0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "red")
lines(xx, pbckdengpd(xx, kerncentres, lambda = 0.5, xi = -0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "blue")
legend("topleft", paste("xi =", c(0, 0.3, -0.3)),
col=c("black", "red", "blue"), lty = 1, cex = 0.5)
```

```

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

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

## End(Not run)

```

bckdengpdcon

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

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with boundary corrected kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the bandwidth 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](#) or [fbckden](#) function for cross-validation MLE for bandwidth.

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

Value

[dbckdengpdcon](#) gives the density, [pbckdengpdcon](#) gives the cumulative distribution function, [qbckdengpdcon](#) gives the quantile function and [rbckdengpdcon](#) gives a random sample.

Boundary Correction Methods

See [dbckden](#) for details of BCKDE methods.

Warning

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

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

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

Other kden kdenkdpd kdenkdpdcon bckden bckdenkdpd bckdenkdpdcon fkden fkdenkdpd fkdenkdpdcon
fbckden fbckdenkdpd fbckdenkdpdcon: bckdenkdpd, bckdenkdpd, bckdenkdpd, bckdenkdpd, bckdenkdpd,
dbckdenkdpd, dbckdenkdpd, dbckdenkdpd, dbckdenkdpd, dbckdenkdpd, pbckdenkdpd, pbckdenkdpd,
pbckdenkdpd, pbckdenkdpd, pbckdenkdpd, qbckdenkdpd, qbckdenkdpd, qbckdenkdpd, qbckdenkdpd,
qbckdenkdpd, rbckdenkdpd, rbckdenkdpd, rbckdenkdpd, rbckdenkdpd, rbckdenkdpd; bckden, bckden,
bckden, bckden, bckden, dbckden, dbckden, dbckden, dbckden, dbckden, pbckden, pbckden,
pbckden, pbckden, pbckden, qbckden, qbckden, qbckden, qbckden, qbckden, rbckden, rbckden,
rbckden, rbckden, rbckden; dkdenkdpdcon, dkdenkdpdcon, dkdenkdpdcon, dkdenkdpdcon, dkdenkdpdcon,
kdenkdpdcon, kdenkdpdcon, kdenkdpdcon, kdenkdpdcon, kdenkdpdcon, pkdenkdpdcon, pkdenkdpdcon,
pkdenkdpdcon, pkdenkdpdcon, pkdenkdpdcon, qkdenkdpdcon, qkdenkdpdcon, qkdenkdpdcon, qkdenkdpdcon,
qkdenkdpdcon, rkdenkdpdcon, rkdenkdpdcon, rkdenkdpdcon, rkdenkdpdcon, rkdenkdpdcon; dkdenkdpd,
dkdenkdpd, dkdenkdpd, dkdenkdpd, dkdenkdpd, kdenkdpd, kdenkdpd, kdenkdpd, kdenkdpd, kdenkdpd,
pkdenkdpd, pkdenkdpd, pkdenkdpd, pkdenkdpd, pkdenkdpd, qkdenkdpd, qkdenkdpd, qkdenkdpd, qkdenkdpd,
qkdenkdpd, rkdenkdpd, rkdenkdpd, rkdenkdpd, rkdenkdpd, rkdenkdpd; dkden, dkden, dkden, dkden,
dkden, kden, kden, kden, kden, kden, kden, pkden, pkden, pkden, pkden, pkden, qkden, qkden, qkden,

qkden, qkden, rkden, rkden, rkden, rkden, rkden; fbckden, fbckden, fbckden, lbckden, lbckden, lbckden, nlbckden, nlbckden, nlbckden, nlbckden; fkden, fkden, fkden, lkden, lkden, lkden, nlkden, nlkden, nlkden

Examples

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

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

plot(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", type = "l")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = 0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "red")
lines(xx, pbckdengpdcon(xx, kerncentres, lambda = 0.5, xi = -0.3, bcmethod = "reflect"),
xlab = "x", ylab = "F(x)", col = "blue")
legend("topleft", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1, cex = 0.5)

kerncentres = rweibull(1000, 2, 1)
x = rbckdengpdcon(1000, kerncentres, lambda = 0.1, phiu = TRUE, bcmethod = "reflect")
xx = seq(0.01, 3.5, 0.01)
hist(x, breaks = 100, freq = FALSE)
lines(xx, dbckdengpdcon(xx, kerncentres, lambda = 0.1, phiu = TRUE, bcmethod = "reflect"),
xlab = "x", ylab = "f(x)")

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

## End(Not run)
```

betagpd

Beta Bulk and GPD Tail Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the beta shape 1 bshape1 and shape 2 bshape2, threshold u GPD scale sigmau and shape xi and tail fraction phiu.

Usage

```

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

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

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

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

```

Arguments

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

Details

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

The user can pre-specify phiu permitting a parameterised value for the tail fraction 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 $\phi_u = 1 - H(u)$.

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](#), [betagpdcon](#), [betagpdcon](#), [betagpdcon](#), [betagpdcon](#), [dbetagpdcon](#), [dbetagpdcon](#), [dbetagpdcon](#), [dbetagpdcon](#), [dbetagpdcon](#), [dbetagpdcon](#), [pbetagpdcon](#), [pbetagpdcon](#), [pbetagpdcon](#), [pbetagpdcon](#), [qbetagpdcon](#), [qbetagpdcon](#), [qbetagpdcon](#), [qbetagpdcon](#), [qbetagpdcon](#), [rbetagpdcon](#), [rbetagpdcon](#), [rbetagpdcon](#), [rbetagpdcon](#), [rbetagpdcon](#)

Examples

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

x = rbetagpd(1000, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2)
xx = seq(-0.1, 2, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, dbetagpd(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2))

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

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

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

## End(Not run)
```

betagpdcon

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

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with beta for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the beta shape 1 bshape1 and shape 2 bshape2, threshold u GPD shape xi and tail fraction phiu.

Usage

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

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

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

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

Arguments

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

Details

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

The user can pre-specify phiu permitting a parameterised value for the tail fraction 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](#), [betagpd](#), [betagpd](#), [betagpd](#), [betagpd](#), [dbetagpd](#), [dbetagpd](#), [dbetagpd](#), [dbetagpd](#), [dbetagpd](#), [pbetagpd](#), [pbetagpd](#), [pbetagpd](#), [pbetagpd](#), [pbetagpd](#), [qbetagpd](#), [qbetagpd](#), [qbetagpd](#), [qbetagpd](#), [qbetagpd](#), [qbetagpd](#), [rbetagpd](#), [rbetagpd](#), [rbetagpd](#), [rbetagpd](#), [rbetagpd](#)

Examples

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

x = rbetagpdcon(1000, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2)
xx = seq(-0.1, 2, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 2))
lines(xx, dbetagpdcon(xx, bshape1 = 1.5, bshape2 = 2, u = 0.7, phiu = 0.2))

# three tail behaviours
```

```

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

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

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

## End(Not run)

```

checking

Internal functions for checking function input arguments

Description

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

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

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

Usage

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

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

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

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

```
check.n(n, allowzero = FALSE)
```

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

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

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

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



```

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

check.optim(method)

check.control(control)

check.bcmethod(bcmethod)

check.nn(nn)

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

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

```

Arguments

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

Value

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

Author(s)

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

dwm

Dynamically Weighted Mixture Model

Description

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

Usage

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

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

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

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

Arguments

<code>x</code>	quantiles
<code>wshape</code>	Weibull shape (positive)
<code>wscale</code>	Weibull scale (positive)
<code>cmu</code>	Cauchy location
<code>ctau</code>	Cauchy scale
<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>qinit</code>	scalar or vector of initial values for the quantile estimate
<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, wsca` and `dgp(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:
set.seed(1)
par(mfrow = c(2, 2))

xx = seq(0.001, 5, 0.01)
f = ddwm(xx, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2,
     ylab = "density", main = "Plot example in Frigessi et al. (2002)")
lines(xx, dgpdx(xx, sigmau = 1, xi = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, dweibull(xx, shape = 2, scale = 1/gamma(1.5)), col = "blue", lty = 2, lwd = 2)
legend('topright', c('DWM', 'Weibull', 'GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

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

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

plot(xx, pdwm(xx, wshape = 2, wscale = 1/gamma(1.5), cmu = 1, ctau = 1, sigmau = 1, xi = 0.1),
     xlim = c(0, 15), type = 'l', lwd = 2,
     xlab = "x", ylab = "F(x)")
lines(xx, pgpdx(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, alpha = 0.05, N = 1000,
  legend = FALSE, ...)
```

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

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

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

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

Arguments

modelfit	fitted extreme value mixture model object
upperfocus	logical, should plot focus on upper tail?
alpha	significance level over range (0, 1), or NULL for no CI
N	number of Monte Carlo simulation for CI (N>=10)
legend	logical, should legend be included
...	further arguments to be passed to the plotting functions

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_u are shown. For the two tailed models both thresholds u_l and u_r and corresponding tail fractions ϕ_{u_l} and ϕ_{u_r} are shown. The approximate pointwise confidence intervals for the quantiles are obtained by Monte Carlo simulation using the estimated parameters. Notice that these intervals ignore the parameter estimation uncertainty.

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

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

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

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

Value

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

Acknowledgments

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

Note

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

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

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

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

Author(s)

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

References

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

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

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

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

See Also

[ppoints](#), [plot.uvevd](#) and [gpd.diag](#).

Examples

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

x = sort(rnorm(1000))
fit = fnormgpd(x)
evmix.diag(fit)

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

## End(Not run)
```

fbckden

Cross-validation MLE Fitting of Boundary Corrected Kernel Density Estimation Using a Variety of Approaches

Description

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

Usage

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

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

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

Arguments

<code>x</code>	location to evaluate KDE (single scalar or vector)
<code>linit</code>	initial value for bandwidth (as kernel half-width) or NULL
<code>bwinit</code>	initial value for bandwidth (as kernel standard deviations) or NULL
<code>kernel</code>	kernel name (default = "gaussian")
<code>extracentres</code>	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
<code>bcmethod</code>	boundary correction method
<code>proper</code>	logical, whether density is renormalised to integrate to unity (where needed)
<code>nn</code>	non-negativity correction method (simple boundary correction only)
<code>offset</code>	offset added to kernel centres (logtrans only) or NULL
<code>xmax</code>	upper bound on support (copula and beta kernels only) or NULL
<code>add.jitter</code>	logical, whether jitter is needed for rounded kernel centres
<code>factor</code>	see jitter
<code>amount</code>	see jitter
<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>...</code>	optional inputs passed to optim
<code>lambda</code>	scalar bandwidth for kernel (as half-width of kernel)
<code>bw</code>	scalar bandwidth for kernel (as standard deviations of kernel)
<code>log</code>	logical, if TRUE then log density

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

lambda:	MLE of lambda (kernel half-width)
bw:	MLE of bw (kernel standard deviations)
kernel:	kernel name
bcmethod:	boundary correction method
proper:	logical, whether renormalisation is requested
nn:	non-negative correction method
offset:	offset for log transformation method
xmax:	maximum value of scale beta or copula

The output list has some duplicate entries and repeats some of the inputs to both provide similar items to those from `fspot` 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 data rounding is the likely culprit). Only use the jittering when the MLE of the bandwidth is far too small.

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

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

Invalid parameter ranges will give 0 for likelihood, $\log(0)=-\text{Inf}$ 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: bckdengpdcon, bckdengpdcon, bckdengpdcon, bckdengpdcon, bckdengpdcon, dbckdengpdcon, dbckdengpdcon, dbckdengpdcon, dbckdengpdcon, dbckdengpdcon, dbckdengpdcon, pbckdengpdcon, pbckdengpdcon, pbckdengpdcon, pbckdengpdcon, pbckdengpdcon, qbckdengpdcon, qbckdengpdcon, qbckdengpdcon, qbckdengpdcon, qbckdengpdcon, rbckdengpdcon, rbckdengpdcon, rbckdengpdcon, rbckdengpdcon; bckdengpd, bckdengpd, bckdengpd, bckdengpd, dbckdengpd, dbckdengpd, dbckdengpd, dbckdengpd, pbckdengpd, pbckdengpd, pbckdengpd, pbckdengpd, qbckdengpd, qbckdengpd, qbckdengpd, qbckdengpd, qbckdengpd, rbckdengpd, rbckdengpd, rbckdengpd, rbckdengpd, rbckdengpd, rbckdengpd; bckden, bckden, bckden, bckden, dbckden, dbckden, dbckden, dbckden, dbckden, dbckden, pbckden, pbckden, pbckden, pbckden, qbckden, qbckden, qbckden, qbckden, qbckden, qbckden, rbckden, rbckden, rbckden, rbckden; dkdengpdcon, dkdengpdcon, dkdengpdcon, dkdengpdcon, dkdengpdcon, kdengpdcon, kdengpdcon, kdengpdcon, kdengpdcon, kdengpdcon, pkdengpdcon, pkdengpdcon, pkdengpdcon, pkdengpdcon, qkdengpdcon, qkdengpdcon, qkdengpdcon, qkdengpdcon, qkdengpdcon, rkden, rkden, rkden, rkden, rkden, rkden; dkden, dkden, dkden, dkden, dkden, kden, kden, kden, kden, kden, kden, pkden, pkden, pkden, pkden, pkden, pkden, qkden, qkden, qkden, qkden, qkden, rkden, rkden, rkden, rkden, rkden, rkden; fkden, fkden, fkden, lkden, lkden, lkden, nlkden, nlkden, nlkden

Examples

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

```

par(mfrow = c(1, 1))

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

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

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

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

## End(Not run)

```

fbckdengpd

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

Description

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

Usage

```
fbckdengpd(x, phiu = TRUE, useq = NULL, fixedu = FALSE, pvector = NULL,
```

```

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

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

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

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

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

```

Arguments

x	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
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

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 [fnormmgpd](#) for details, type help fnormmgpd. Only the different features are outlined below for brevity.

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

Negative data are ignored.

Cross-validation likelihood is used for BCKDE, but standard likelihood is used for GPD component. See help for [fkden](#) for details, type help fkden.

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

The possible kernels are also defined in [kernels](#) with the "gaussian" as the default choice.

Unlike the standard KDE, there is no general rule-of-thumb bandwidth for all these estimators, with only certain methods having a guideline in the literature, so none have been implemented. Hence, a bandwidth must always be specified.

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

Value

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

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

xi:	MLE of GPD shape
phiu:	MLE of tail fraction (bulk model or parameterised approach)
se.phiu:	standard error of MLE of tail fraction
bw:	MLE of bw (kernel standard deviations)
kernel:	kernel name
bcmethod:	boundary correction method
proper:	logical, whether renormalisation is requested
nn:	non-negative correction method
offset:	offset for log transformation method
xmax:	maximum value of scaled beta or copula

Boundary Correction Methods

See [dbckden](#) for details of BCKDE methods.

Warning

See important warnings about cross-validation likelihood estimation in [fkden](#), type `help fkden`.

See important warnings about boundary correction approaches in [dbckden](#), type `help bckden`.

Acknowledgments

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

Note

See notes in [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Author(s)

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

References

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

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

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


```
# 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 = TRUE, ...)
```

```
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 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.

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
nlhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	ϕ_{iu} to be consistent with evd
nlh:	minimum negative log-likelihood
n:	total sample size
lambda:	MLE of λ (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
bcmeth:	boundary correction method
proper:	logical, whether renormalisation is requested
nn:	non-negative correction method
offset:	offset for log transformation method
xmax:	maximum value of scaled beta or copula

Boundary Correction Methods

See [dbckden](#) for details of BCKDE methods.

Warning

See important warnings about cross-validation likelihood estimation in [fkden](#), type `help fkden`.

See important warnings about boundary correction approaches in [dbckden](#), type `help bckden`.

Acknowledgments

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

Note

See notes in [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Author(s)

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

References

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

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

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

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

Examples

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

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

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

# No continuity constraint
pinit = c(0.1, quantile(x, 0.9), 1, 0.1) # initial values required for BCKDE
fit2 = fbckdengpd(x, pvector = pinit, bcmethod = "cutnorm")
with(fit2, lines(xx, dbckdengpd(xx, x, lambda, u, sigma, 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, ...)

lbtetagpd(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)

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

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

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

```

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
bshape1	scalar beta shape 1 (positive)

bshape2	scalar beta shape 2 (positive)
u	scalar threshold over (0, 1)
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormmgpd](#) for details, type `help fnormmgpd`. 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
nlhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nlh:	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 [fnormmgpd](#), type `help fnormmgpd`. Based on code by Anna MacDonald produced for MATLAB.

Note

When pvector=NULL then the initial values are:

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

Author(s)

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

References

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

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

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

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

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

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

See Also

[dbeta](#), [fgpd](#) and [gpd](#)

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

Examples

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

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

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



```

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

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

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

## End(Not run)

```

fbetagpdcon

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

Description

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

Usage

```

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

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

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

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

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

```

Arguments

x	vector of sample data
phiu	probability of being above threshold (0, 1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
bshape1	scalar beta shape 1 (positive)
bshape2	scalar beta shape 2 (positive)
u	scalar threshold over (0, 1)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Value

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

call:	optim call
x:	data vector x
init:	pvector
fixedu:	fixed threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
nlhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output

mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
n:	total sample size
bshape1:	MLE of beta shape1
bshape2:	MLE of beta shape2
u:	threshold (fixed or MLE)
sigmau:	MLE of GPD scale (estimated from other parameters)
xi:	MLE of GPD shape
phiu:	MLE of tail fraction (bulk model or parameterised approach)
se.phi:	standard error of MLE of tail fraction

Acknowledgments

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

Note

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

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

Author(s)

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

References

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

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

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

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

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

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

See Also

[dbeta](#), [fgpd](#) and [gpd](#)

Other [betagpd](#) [betagpdcon](#) [fbetagpd](#) [fbetagpdcon](#) [normgpd](#) [fnormgpd](#): [fbetagpd](#), [fbetagpd](#), [fbetagpd](#), [fbetagpd](#), [fbetagpd](#), [lbetagpd](#), [lbetagpd](#), [lbetagpd](#), [lbetagpd](#), [lbetagpd](#), [lbetagpd](#), [nlbetagpd](#), [nlbetagpd](#), [nlbetagpd](#), [nlbetagpd](#), [nlubetagpd](#), [nlubetagpd](#), [nlubetagpd](#), [nlubetagpd](#), [nlubetagpd](#), [nlubetagpd](#), [proflubetagpd](#), [proflubetagpd](#), [proflubetagpd](#), [proflubetagpd](#), [proflubetagpd](#)

Examples

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

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

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

# No continuity constraint
fit2 = fbetagpd(x, phiu = FALSE)
with(fit2, lines(xx, dbetagpd(xx, bshape1, bshape2, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density", "No continuity constraint", "With 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(0.3, 0.7, length = 20))
fitfix = fbetagpdcon(x, useq = seq(0.3, 0.7, length = 20), fixedu = TRUE)

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

## End(Not run)
```

Description

Maximum likelihood estimation for fitting the dynamically weighted mixture model

Usage

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

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

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

Arguments

x	quantiles
pvector	vector of initial values of parameters (wshape, wscale, cmu, ctau, sigmau, xi) or NULL
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
wshape	Weibull shape (positive)
wscale	Weibull scale (positive)
cmu	Cauchy location
ctau	Cauchy scale
sigmau	scale parameter (positive)
xi	shape parameter
log	logical, if TRUE then log density

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

<code>call:</code>	<code>optim</code> call
<code>x:</code>	data vector <code>x</code>
<code>init:</code>	<code>pvector</code>
<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>wshape:</code>	MLE of Weibull shape
<code>wscale:</code>	MLE of Weibull scale
<code>mu:</code>	MLE of Cauchy location
<code>tau:</code>	MLE of Cauchy scale
<code>sigmau:</code>	MLE of GPD scale
<code>xi:</code>	MLE of GPD shape

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

Acknowledgments

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

Note

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

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

Infinite and missing sample values are dropped.

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

Author(s)

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

References

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

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

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

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

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

See Also

[fgpd](#) and [gpd](#)

Examples

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

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

fit = fdwm(x, std.err = FALSE)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, ddwm(xx, wshape, wscale, cmu, ctau, sigmau, xi), col="red"))

## End(Not run)
```

fgammagpd

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

Description

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

Usage

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

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

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

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

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

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
gshape	scalar gamma shape (positive)
gscale	scalar gamma scale (positive)
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Value

Log-likelihood is given by [lgammagpd](#) and its wrappers for negative log-likelihood from [nlammagpd](#) and [nlugammagpd](#). Profile likelihood for single threshold given by [proflugammagpd](#). Fitting function [fgammagpd](#) returns a simple list with the following elements

```

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

```

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

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

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

Author(s)

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

References

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

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

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

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

Examples

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

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

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

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

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

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

## End(Not run)
```

fgammagpdcon

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

Description

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

Usage

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

lgammagpdcon(x, gshape = 1, gscale = 1, u = qgamma(0.9, gshape, 1/gscale),
  xi = 0, phiu = TRUE, log = TRUE)

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

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

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

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
gshape	scalar gamma shape (positive)
gscale	scalar gamma scale (positive)
u	scalar threshold value
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Value

Log-likelihood is given by [lammagpdcon](#) and it's wrappers for negative log-likelihood from [nlammagpdcon](#) and [nlugammagpdcon](#). Profile likelihood for single threshold given by [proflugammagpdcon](#). Fitting function [fgammagpdcon](#) returns a simple list with the following elements

```

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

```

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

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

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

Author(s)

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

References

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

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

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

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

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

dgamma, fgpd and gpd

[illegible]

Examples

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

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

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

# No continuity constraint
fit2 = fgammagpd(x, phiu = FALSE)
with(fit2, lines(xx, dgammagpd(xx, gshape, gscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density", "No continuity constraint", "With 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, dgammagpdcon(xx, gshape, gscale, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density", "Default initial value (90% quantile)",
  "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
      col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fgkg

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

Description

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

Usage

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

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

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

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

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

Arguments

x	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 ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq)
pvector	vector of initial values of parameters or NULL for default values, see below
kernel	kernel name (default = "gaussian")
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
lambda	scalar bandwidth for kernel (as half-width of kernel)
ul	scalar lower tail threshold
sigmaul	scalar lower tail GPD scale parameter (positive)

xil	scalar lower tail GPD shape parameter
ur	scalar upper tail threshold
sigmaur	scalar upper tail GPD scale parameter (positive)
xir	scalar upper tail GPD shape parameter
bw	scalar bandwidth for kernel (as standard deviations of kernel)
log	logical, if TRUE then log 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 is fitted to the entire dataset using maximum likelihood estimation. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

See help for [fnormgpd](#) and [fgkg](#) for details, type `help fnormgpd` and `help fgkg`. Only the different features are outlined below for brevity.

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

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD components. See help for [fkden](#) for details, type `help fkden`.

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

The possible kernels are also defined in [kernels](#) with the "gaussian" as the default choice.

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

- default values phiul=TRUE and phiur=TRUE - tail fractions specified by KDE distribution and 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](#), [nlkg](#) and [nlugkg](#) it can be logical or numeric:

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

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

Value

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

call: optim call

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

Warning

See important warnings about cross-validation likelihood estimation in [fkden](#), type `help fkden`.

Acknowledgments

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

Note

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

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

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

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
http://en.wikipedia.org/wiki/Kernel_density_estimation
[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/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](#): [fgkgcon](#), [fgkgcon](#), [fgkgcon](#), [fgkgcon](#), [lgkgcon](#), [lgkgcon](#), [lgkgcon](#), [lgkgcon](#), [lgkgcon](#), [nlkgkgcon](#), [nlkgkgcon](#), [nlkgkgcon](#), [nlkgkgcon](#), [nlugkgcon](#), [nlugkgcon](#), [nlugkgcon](#), [nlugkgcon](#), [nlugkgcon](#), [nlugkgcon](#), [proflugkgcon](#), [proflugkgcon](#), [proflugkgcon](#), [proflugkgcon](#), [proflugkgcon](#)

Examples

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

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

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

# Parameterised tail fraction
```

```

fit2 = fgkg(x, phiul = FALSE, phiur = FALSE)
with(fit2, lines(xx, dgkg(xx, x, lambda, ul, sigmaul, xil, phiul,
  ur, sigmaur, xir, phiur), col="blue"))
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density", "Bulk Tail Fraction", "Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

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

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

## End(Not run)

```

fgkgcon

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

Description

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

Usage

```

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

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

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 = TRUE, ...)

nflugkgcon(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 ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq)
pvector	vector of initial values of parameters or NULL for default values, see below
kernel	kernel name (default = "gaussian")
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
lambda	scalar bandwidth for kernel (as half-width of kernel)
ul	scalar lower tail threshold
xil	scalar lower tail GPD shape parameter
ur	scalar upper tail threshold
xir	scalar upper tail GPD shape parameter
bw	scalar bandwidth for kernel (as standard deviations of kernel)
log	logical, if TRUE then log 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 thresholds are also estimated and `(lambda, xil, xir)` for profile likelihood or fixed threshold approach.

Cross-validation likelihood is used for KDE, but standard likelihood is used for GPD components. See help for [fkden](#) for details, type `help fkden`.

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

The possible kernels are also defined in [kernels](#) with the "gaussian" as the default choice.

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

- default values `phiul=TRUE` and `phiur=TRUE` - tail fractions specified by KDE distribution and 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.

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

Value

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

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

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

Warning

See important warnings about cross-validation likelihood estimation in [fkden](#), type `help fkden`.

Acknowledgments

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

Note

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

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

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

Author(s)

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

References

<http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
http://en.wikipedia.org/wiki/Kernel_density_estimation

[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/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](#), [fgkg](#), [fgkg](#), [fgkg](#), [lgkg](#), [lgkg](#), [lgkg](#), [lgkg](#), [lgkg](#), [nlkgk](#), [nlkgk](#), [nlkgk](#), [nlkgk](#), [nlkgk](#), [nlugkg](#), [nlugkg](#), [nlugkg](#), [nlugkg](#), [proflugkg](#), [proflugkg](#), [proflugkg](#), [proflugkg](#), [proflugkg](#)

Examples

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

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

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

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

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



```

fitfix = fgkgcon(x, ulseq = seq(-2, -0.2, length = 10),
  urseq = seq(0.2, 2, length = 10), fixedu = TRUE)

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

## End(Not run)

```

fgng

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

Description

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

Usage

```

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

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

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

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

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

```

Arguments

x	vector of sample data
phiul	probability of being below lower threshold (0, 1) or logical, see Details in help for fgng

phiur	probability of being above upper threshold (0, 1) or logical, see Details in help for fgng
ulseq	vector of lower thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
urseq	vector of upper thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
ul	scalar lower tail threshold
sigmaul	scalar lower tail GPD scale parameter (positive)
xil	scalar lower tail GPD shape parameter
ur	scalar upper tail threshold
sigmaur	scalar upper tail GPD scale parameter (positive)
xir	scalar upper tail GPD shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output
ulr	vector of length 2 giving lower and upper tail thresholds or NULL for default values

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

- default values phiul=TRUE and phiur=TRUE - tail fractions specified by normal distribution `pnorm(ul, nmean, nsd)` and 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](#), [nlngng](#) and [nlugng](#) it can be logical or numeric:

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

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

Value

Log-likelihood is given by [lgng](#) and it's wrappers for negative log-likelihood from [nlngng](#) and [nlugng](#). Profile likelihood for both thresholds given by [proflugng](#). Fitting function [fgng](#) 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 thresholds, logical
<code>ulseq:</code>	lower threshold vector for profile likelihood or scalar for fixed threshold
<code>urseq:</code>	upper threshold vector for profile likelihood or scalar for fixed threshold
<code>nllhuseq:</code>	profile negative log-likelihood at each threshold pair in (ulseq, urseq)
<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 evd
<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>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

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`. Based on code by Xin Zhao produced for MATLAB.

Note

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

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

Author(s)

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

References

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

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

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

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

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

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

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

See Also

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

Other [normgpd](#) [normgpdcon](#) [gng](#) [gngcon](#) [fnormgpd](#) [fnormgpdcon](#) [fgng](#) [fgngcon](#): [dgngcon](#), [dgngcon](#), [dgngcon](#), [dgngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#); [dgng](#), [dgng](#), [dgng](#), [dgng](#), [dgng](#), [gng](#), [gng](#), [gng](#), [gng](#), [gng](#), [pgng](#), [pgng](#), [pgng](#), [pgng](#), [pgng](#), [qngng](#), [qngng](#), [qngng](#), [qngng](#), [qngng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#); [ditmgng](#), [ditmgng](#), [ditmgng](#), [ditmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#); [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#); [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#); [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#); [fitmgng](#), [fitmgng](#), [fitmgng](#), [fitmgng](#), [fitmgng](#), [litmgng](#), [litmgng](#), [litmgng](#), [litmgng](#), [litmgng](#), [litmgng](#), [nluitmgng](#), [nlitmgng](#), [nlitmgng](#), [nlitmgng](#), [nlitmgng](#), [nlitmgng](#), [nluitmgng](#), [nluitmgng](#), [nluitmgng](#), [nluitmgng](#), [nluitmgng](#), [nluitmgng](#), [profluitmgng](#), [profluitmgng](#), [profluitmgng](#), [profluitmgng](#), [profluitmgng](#); [fnormgpdcon](#), [fnormgpdcon](#),

```
fnormgpdcon, fnormgpdcon, fnormgpdcon, lnormgpdcon, lnormgpdcon, lnormgpdcon, lnormgpdcon,
lnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon,
nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon,
proflunormgpdcon, proflunormgpdcon, proflunormgpdcon; fnormgpd, fnormgpd, fnormgpd,
fnormgpd, fnormgpd, lnormgpd, lnormgpd, lnormgpd, lnormgpd, lnormgpd, lnormgpd, nlnormgpd, nlnormgpd,
nlnormgpd, nlnormgpd, nlnormgpd, nlunormgpd, nlunormgpd, nlunormgpd, nlunormgpd, nlunormgpd,
proflunormgpd, proflunormgpd, proflunormgpd, proflunormgpd, proflunormgpd, proflunormgpd
```

Examples

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

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

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

# Parameterised tail fraction
fit2 = fgng(x, phiul = FALSE, phiur = FALSE)
with(fit2, lines(xx, dgng(xx, nmean, nsd, ul, sigmaul, xil, phiul,
  ur, sigmaur, xir, phiur), col="blue"))
abline(v = c(fit2$ul, fit2$ur), col = "blue")
legend("topright", c("True Density", "Bulk Tail Fraction", "Parameterised Tail Fraction"),
  col=c("black", "red", "blue"), lty = 1)

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

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

## End(Not run)
```

fgngcon

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

Description

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

Usage

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

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

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

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

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

Arguments

x	vector of sample data
phiul	probability of being below lower threshold (0,1) or logical, see Details in help for fgng
phiur	probability of being above upper threshold (0,1) or logical, see Details in help for fgng
ulseq	vector of lower thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
urseq	vector of upper thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in ulseq/urseq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in ulseq/urseq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim

nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
ul	scalar lower tail threshold
xil	scalar lower tail GPD shape parameter
ur	scalar upper tail threshold
xir	scalar upper tail GPD shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output
ulr	vector of length 2 giving lower and upper tail thresholds or NULL for default values

Details

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

See help for [fnormgpd](#) and [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 thresholds are also estimated and `(nmean, nsd, xil, xir)` for profile likelihood or fixed threshold approach.

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

Value

Log-likelihood is given by [lgngcon](#) and its wrappers for negative log-likelihood from [nlngngcon](#) and [nlugngcon](#). Profile likelihood for both thresholds given by [proflugngcon](#). Fitting function [fgngcon](#) returns a simple list with the following elements

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

xil:	MLE of lower tail GPD shape
phiul:	MLE of lower tail fraction (bulk model or parameterised approach)
se.phiul:	standard error of MLE of lower tail fraction
ur:	upper threshold (fixed or MLE)
sigmaur:	MLE of upper tail GPD scale (estimated from other parameters)
xir:	MLE of upper tail GPD shape
phiur:	MLE of upper tail fraction (bulk model or parameterised approach)
se.phiur:	standard error of MLE of upper tail fraction

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`. Based on code by Xin Zhao produced for MATLAB.

Note

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

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

Author(s)

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

References

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

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

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

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

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

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

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


```

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

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

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

## End(Not run)

```

fgpd

*MLE Fitting of Generalised Pareto Distribution (GPD)***Description**

Maximum likelihood estimation for fitting the GPD with parameters scale 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>phiu</code>	probability of being above threshold $[0, 1]$ or NULL, see Details
<code>pvector</code>	vector of initial values of GPD parameters (σ , ξ) or NULL
<code>std.err</code>	logical, should standard errors be calculated
<code>method</code>	optimisation method (see optim)

control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

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

Log-likelihood calculations are carried out in [lgpd](#), which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood is a wrapper for [lgpd](#), designed towards making it useable for optimisation (e.g. parameters are given a vector as first input).

The default value for the tail fraction ϕ_u 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 ϕ_u is estimated and output as `se.phi_u`, 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 ϕ_u 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 ϕ_u cannot be NULL but can be over the range $[0, 1]$, i.e. which includes $\phi_u = 0$.

The value of ϕ_u does not effect the GPD parameter estimates, only the value of the likelihood, as:

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

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

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.

Acknowledgments

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

Note

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

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

Default values for the threshold u=0 and tail fraction phiu=NULL are given in the fitting [fpgd](#), in which case the MLE assumes that excesses over the threshold are given, rather than exceedances.

The usual default of phiu=1 is given in the likelihood functions [lpgd](#) and [nlpgd](#).

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

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

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

Author(s)

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

References

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

See Also

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

Other gpd fgpdp: [dgpdp](#), [dgpdp](#), [dgpdp](#), [dgpdp](#), [dgpdp](#), [gpd](#), [gpd](#), [gpd](#), [gpd](#), [gpd](#), [pgpd](#), [pgpd](#), [pgpd](#), [pgpd](#), [pgpd](#), [qgpd](#), [qgpd](#), [qgpd](#), [qgpd](#), [qgpd](#), [rgpd](#), [rgpd](#), [rgpd](#), [rgpd](#), [rgpd](#)

Examples

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

# GPD is conditional model for threshold exceedances
# so tail fraction phiu not relevant when only have exceedances
x = rgpd(1000, u = 10, sigmau = 5, xi = 0.2)
xx = seq(0, 100, 0.1)
hist(x, breaks = 100, freq = FALSE, xlim = c(0, 100))
lines(xx, dgpdp(xx, u = 10, sigmau = 5, xi = 0.2))
fit = fgpd(x, u = 10)
lines(xx, dgpdp(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi), col="red")

# but tail fraction phiu is needed for conditional modelling of population tail
x = rnorm(10000)
xx = seq(-4, 4, 0.01)
hist(x, breaks = 200, freq = FALSE, xlim = c(0, 4))
lines(xx, dnorm(xx), lwd = 2)
fit = fgpd(x, u = 1)
lines(xx, dgpdp(xx, u = fit$u, sigmau = fit$sigmau, xi = fit$xi, phiu = fit$phiu),
      col = "red", lwd = 2)
legend("topright", c("True Density", "Fitted Density"), col=c("black", "red"), lty = 1)
```

fhpd

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

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

Usage

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

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

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

Arguments

x	quantiles
pvector	vector of initial values of parameters (nmean, nsd, xi) or NULL
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)

control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
xi	shape parameter
log	logical, if TRUE then log density

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

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

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

Invalid parameter ranges will give 0 for likelihood, $\log(0)=-\text{Inf}$ 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](#), [fhpdcon](#), [fhpdcon](#), [fhpdcon](#), [fhpdcon](#), [lhpdcon](#), [lhpdcon](#), [lhpdcon](#), [lhpdcon](#), [lhpdcon](#), [nlhpdcon](#), [nlhpdcon](#), [nlhpdcon](#), [nlhpdcon](#), [nlhpdcon](#), [nlhpdcon](#), [nluhpdcon](#), [nluhpdcon](#), [nluhpdcon](#), [nluhpdcon](#), [nluhpdcon](#), [profluhpdcon](#), [profluhpdcon](#), [profluhpdcon](#), [profluhpdcon](#), [profluhpdcon](#)

Examples

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

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

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

# Notice that if tail fraction is included a better fit is obtained
fit2 = fnormgpdcon(x, std.err = FALSE)
with(fit2, lines(xx, dnormgpdcon(xx, nmean, nsd, u, xi), col="blue"))
abline(v = fit2$u)
legend("topright", c("Standard Normal", "Hybrid Pareto", "Normal+GPD Continuous"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

fhpdcon

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

Description

Maximum likelihood estimation for fitting the Hybrid Pareto extreme value mixture model, with only continuity at threshold and not necessarily continuous in first derivative. With options for profile likelihood estimation for threshold and fixed threshold approach.

Usage

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

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

nlhpdcon(pvector, x, finitelik = FALSE)

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

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


Arguments

<code>x</code>	quantiles
<code>useq</code>	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
<code>fixedu</code>	logical, should threshold be fixed (at either scalar value in <code>useq</code> , or estimated from maximum of profile likelihood evaluated at sequence of thresholds in <code>useq</code>)
<code>pvector</code>	vector of initial values of parameters or NULL for default values, see below
<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>...</code>	optional inputs passed to optim
<code>nmean</code>	scalar normal mean
<code>nsd</code>	scalar normal standard deviation (positive)
<code>u</code>	scalar threshold value
<code>xi</code>	shape parameter
<code>log</code>	logical, if TRUE then log density

Details

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

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

Value

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

<code>call:</code>	<code>optim call</code>
<code>x:</code>	data vector <code>x</code>
<code>init:</code>	<code>pvector</code>

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

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 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](#), [fhpd](#), [fhpd](#), [lhpd](#), [lhpd](#), [lhpd](#), [nlhpd](#), [nlhpd](#), [nlhpd](#)

Examples

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

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

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

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

# No continuity constraint
fit2 = fhpd(x)
with(fit2, lines(xx, dhpdcon(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, dhpdcon(xx, nmean, nsd, u, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dhpdcon(xx, nmean, nsd, u, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topleft", c("True Density", "Default initial value (90% quantile)",
  "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
      col=c("black", "red", "purple", "darkgreen"), lty = 1)

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

par(mfrow = c(1, 1))
```

```

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

```

fitmgng

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

Description

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

Usage

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

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

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

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

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

Arguments

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

std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
nmean	normal mean
nsd	normal standard deviation (positive)
epsilon	interval half-width
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (positive)
xil	lower tail GPD shape parameter
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (positive)
xir	upper tail GPD shape parameter
log	logical, if TRUE then log density
eulr	vector of epsilon, lower and upper thresholds considered in profile likelihood

Details

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

See [ditmgng](#) for explanation of GPD-normal-GPD interval transition model, including mixing functions.

See also help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

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

Value

Log-likelihood is given by [litmgng](#) and its wrappers for negative log-likelihood from [nlitmgng](#) and [nluitmgng](#). Profile likelihood for thresholds and interval half-width given by [profluitmgng](#). Fitting function [fitmgng](#) returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
fixedeu:	fixed epsilon and threshold, logical
ulseq:	lower threshold vector for profile likelihood or scalar for fixed threshold
urseq:	upper threshold vector for profile likelihood or scalar for fixed threshold

<code>eseq:</code>	interval half-width vector for profile likelihood or scalar for fixed threshold
<code>nllheuseq:</code>	profile negative log-likelihood at each combination in (<code>eseq</code> , <code>ulseq</code> , <code>urseq</code>)
<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>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>epsilon:</code>	MLE of transition 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>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

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`. Based on code by Xin Zhao produced for MATLAB.

Note

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

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

Author(s)

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

References

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

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

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

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

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

See Also

[fgng](#), [dnorm](#), [fgpd](#) and [gpd](#)

Other normgpd normgpdcon gng gngcon fnormgpd fnormgpdcon fgng fgngcon: [dgngcon](#), [dgngcon](#), [dgngcon](#), [dgngcon](#), [dgngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [qgngcon](#), [qgngcon](#), [qgngcon](#), [qgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [dgng](#), [dgng](#), [dgng](#), [dgng](#), [dgng](#), [gng](#), [gng](#), [gng](#), [gng](#), [gng](#), [pgng](#), [pgng](#), [pgng](#), [pgng](#), [qgng](#), [qgng](#), [qgng](#), [qgng](#), [qgng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#); [ditmgng](#), [ditmgng](#), [ditmgng](#), [ditmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#); [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#); [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#); [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [nlngcon](#), [nlngcon](#), [nlngcon](#), [nlngcon](#), [nlngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#); [fgng](#), [fgng](#), [fgng](#), [fgng](#), [fgng](#), [lgng](#), [lgng](#), [lgng](#), [lgng](#), [lgng](#), [nlng](#), [nlng](#), [nlng](#), [nlng](#), [nlng](#), [nlng](#), [nlng](#), [nlugng](#), [nlugng](#), [nlugng](#), [nlugng](#), [nlugng](#), [proflugng](#), [proflugng](#), [proflugng](#), [proflugng](#), [proflugng](#); [fnormgpdcon](#), [fnormgpdcon](#), [fnormgpdcon](#), [fnormgpdcon](#), [fnormgpdcon](#), [lnormgpdcon](#), [lnormgpdcon](#), [lnormgpdcon](#), [lnormgpdcon](#), [lnormgpdcon](#), [nlnormgpdcon](#), [nlnormgpdcon](#), [nlnormgpdcon](#), [nlnormgpdcon](#), [nlnormgpdcon](#), [nlunormgpdcon](#), [nlunormgpdcon](#), [nlunormgpdcon](#), [nlunormgpdcon](#), [nlunormgpdcon](#), [proflunormgpdcon](#), [proflunormgpdcon](#), [proflunormgpdcon](#), [proflunormgpdcon](#); [fnormgpd](#), [fnormgpd](#), [fnormgpd](#), [fnormgpd](#), [fnormgpd](#), [lnormgpd](#), [lnormgpd](#), [lnormgpd](#), [lnormgpd](#), [lnormgpd](#), [nlnormgpd](#), [nlnormgpd](#), [nlnormgpd](#), [nlunormgpd](#), [nlunormgpd](#), [nlunormgpd](#), [nlunormgpd](#), [nlunormgpd](#), [proflunormgpd](#), [proflunormgpd](#), [proflunormgpd](#), [proflunormgpd](#), [proflunormgpd](#)

Examples

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

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

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

# Profile likelihood for threshold which is then fixed
fitfix = fitmgng(x, eseq = seq(0, 2, 0.1), ulseq = seq(-2.5, 0, 0.25),
                                                                urseq = seq(0, 2.5, 0.25), fixedeu = TRUE)
with(fitfix, lines(xx, ditmgng(xx, nmean, nsd, epsilon, ul, sigmaul, xil,
                                                                ur, sigmaur, xir), col="blue"))
abline(v = fitfix$ul + fitfix$epsilon * seq(-1, 1), col = "blue")
```

```
abline(v = fitfix$ur + fitfix$epsilon * seq(-1, 1), col = "darkblue")
legend("topright", c("True Density", "GPD-normal-GPD ITM", "Profile likelihood"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

fitmnormgpd	<i>MLE Fitting of Normal Bulk and GPD Tail Interval Transition Mixture Model</i>
-------------	--

Description

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

Usage

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

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

nlitmnormgpd(pvector, x, finitelik = FALSE)

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

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

Arguments

x	vector of sample data
eseq	vector of epsilons (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedeu	logical, should threshold and epsilon be fixed (at either scalar value in useq and eseq, or estimated from maximum of profile likelihood evaluated at grid of thresholds and epsilons in useq and eseq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
nmean	scalar normal mean

nsd	scalar normal standard deviation (positive)
epsilon	interval half-width
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output
eu	vector of epsilon and threshold pair considered in profile likelihood

Details

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

See [ditmnormgpd](#) for explanation of normal-GPD interval transition model, including mixing functions.

See also help for [fnormgpd](#) for mixture model fitting details. Only the different features are outlined below for brevity.

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

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

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

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

Value

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

call:	optim call
x:	data vector x
init:	pvector
fixedeu:	fixed epsilon and threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
eseq:	epsilon vector for profile likelihood or scalar for fixed epsilon
nlhuseq:	profile negative log-likelihood at each combination in (eseq, useq)
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
nlh:	minimum negative log-likelihood
n:	total sample size
nmean:	MLE of normal shape

nsd:	MLE of normal scale
epsilon:	MLE of transition half-width
u:	threshold (fixed or MLE)
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

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

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

Author(s)

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

References

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

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

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

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

See Also

[fnormgpd](#), [dnorm](#), [fgpd](#) and [gpd](#)

Examples

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

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

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

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

## End(Not run)
```

fitmweibullgpd	<i>MLE Fitting of Weibull Bulk and GPD Tail Interval Transition Mixture Model</i>
----------------	---

Description

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

Usage

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

litmweibullgpd(x, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2), u = qweibull(0.9,
  wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale *
  gamma(1 + 1/wshape))^2), xi = 0, log = TRUE)

nlitmweibullgpd(pvector, x, finitelik = FALSE)

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

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

Arguments

x	vector of sample data
eseq	vector of epsilons (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedeu	logical, should threshold and epsilon be fixed (at either scalar value in useq and eseq, or estimated from maximum of profile likelihood evaluated at grid of thresholds and epsilons in useq and eseq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)

control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
wshape	scalar Weibull shape (positive)
wscale	scalar Weibull scale (positive)
epsilon	interval half-width
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output
eu	vector of epsilon and threshold pair considered in profile likelihood

Details

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

See [ditmweibullgpd](#) for explanation of Weibull-GPD interval transition model, including mixing functions.

See also help for [fnormgpd](#) for mixture model fitting details. Only the different features are outlined below for brevity.

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

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

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

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

Negative data are ignored.

Value

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

call:	optim call
x:	data vector x
init:	pvector
fixedeu:	fixed epsilon and threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
eseq:	epsilon vector for profile likelihood or scalar for fixed epsilon
nllheuseq:	profile negative log-likelihood at each combination in (eseq, useq)

optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
nllh:	minimum negative log-likelihood
n:	total sample size
wshape:	MLE of Weibull shape
wscale:	MLE of Weibull scale
epsilon:	MLE of transition half-width
u:	threshold (fixed or MLE)
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

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

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

Author(s)

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

References

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

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

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

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

See Also

[dweibull](#), [fgpd](#) and [gpd](#)

Examples

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

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

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

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

## End(Not run)
```

fkden

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

Description

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

Usage

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

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

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

Arguments

x	quantiles
linit	initial value for bandwidth (as kernel half-width) or NULL
bwinit	initial value for bandwidth (as kernel standard deviations) or NULL
kernel	kernel name (default = "gaussian")
extracentres	extra kernel centres used in KDE, but likelihood contribution not evaluated, or NULL
add.jitter	logical, whether jitter is needed for rounded kernel centres
factor	see jitter
amount	see jitter
std.err	logical, should standard errors be calculated

method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
lambda	bandwidth for kernel (as half-width of kernel) or NULL
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
log	logical, if TRUE then log density

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

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

It will display a warning for non-zero convergence result comes from `optim` function call or for common indicators of lack of convergence (e.g. estimated bandwidth equal to initial value).

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

Value

Log-likelihood is given by `lkden` and its 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>nlh:</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 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 code by Anna MacDonald produced for MATLAB.

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: [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [bckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [dbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [pbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [qbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#), [rbckdengpdcon](#); [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [bckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [dbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [pbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [qbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#), [rbckdengpd](#); [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [bckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [pbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [qbckden](#), [rbckden](#), [rbckden](#), [rbckden](#), [rbckden](#); [dkdengpdcon](#), [dkdengpdcon](#), [dkdengpdcon](#), [dkdengpdcon](#), [dkdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [kdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [pkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#), [qkdengpdcon](#); [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [dkdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [kdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [pkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#), [qkdengpd](#); [dkden](#), [dkden](#), [dkden](#), [dkden](#), [dkden](#), [kden](#), [kden](#), [kden](#), [kden](#), [kden](#), [kden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [pkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [qkden](#), [rkden](#), [rkden](#), [rkden](#), [rkden](#), [rkden](#), [rkden](#); [fbckden](#), [fbckden](#), [fbckden](#), [fbckden](#), [fbckden](#), [lbckden](#), [lbckden](#), [lbckden](#), [lbckden](#), [lbckden](#), [nlbckden](#), [nlbckden](#), [nlbckden](#), [nlbckden](#)

Examples

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

nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 5), ylim = c(0,0.6))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$lambda)*0.05)
```

```

lines(xx,dnorm(xx), col = "black")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
lines(density(x, bw = fit$bw), lwd = 2, lty = 2, col = "blue")
legend("topright", c("True Density", "KDE fitted evmix",
"KDE Using density, default bandwidth", "KDE Using density, c-v likelihood bandwidth"),
lty = c(1, 1, 2, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

par(mfrow = c(2, 1))

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

# remove heavy tails from cv-likelihood evaluation, but still include them in KDE within likelihood
# often gives better bandwidth (see MacDonald et al (2011) for justification)
nk=100
x = rt(nk, df = 2)
xx = seq(-8, 8, 0.01)
fit2 = fkden(x[(x > -4) & (x < 4)], extracentres = x[(x <= -4) | (x >= 4)])
hist(x, seq(floor(min(x)), ceiling(max(x)), 0.5), freq = FALSE, xlim = c(-8, 10))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit2$lambda)*0.05)
lines(xx,dt(xx , df = 2), col = "black")
lines(xx, dkden(xx, x, lambda = fit2$lambda), lwd = 2, col = "red")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "blue")
legend("topright", c("True Density", "KDE fitted evmix, tails removed",
"KDE fitted evmix, tails included"),
lty = c(1, 1, 1), lwd = c(1, 2, 2), col = c("black", "red", "blue"))

## End(Not run)

```

fkdengpd

MLE Fitting of Kernel Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model

Description

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

Usage

```
fkdengpd(x, phiu = TRUE, useq = NULL, fixedu = FALSE, pvector = NULL,
```

```

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

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

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

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

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

```

Arguments

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

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The full parameter vector is $(\lambda, u, \sigma, \xi)$ 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

<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>nllhuseq:</code>	profile negative log-likelihood at each threshold in <code>useq</code>
<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>	ϕ to be consistent with evd
<code>nllh:</code>	minimum negative log-likelihood
<code>n:</code>	total sample size
<code>lambda:</code>	MLE of λ (kernel half-width)
<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>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 code by Anna MacDonald produced for MATLAB.

Note

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

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

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

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
http://en.wikipedia.org/wiki/Kernel_density_estimation
[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/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`: `fbckdengpdcon`, `fbckdengpdcon`, `fbckdengpdcon`, `fbckdengpdcon`, `fbckdengpdcon`, `lbckdengpdcon`, `lbckdengpdcon`, `lbckdengpdcon`, `lbckdengpdcon`, `lbckdengpdcon`, `nlbckdengpdcon`, `nlbckdengpdcon`, `nlbckdengpdcon`, `nlbckdengpdcon`, `nlbckdengpdcon`, `nlubckdengpdcon`, `nlubckdengpdcon`, `nlubckdengpdcon`, `nlubckdengpdcon`, `nlubckdengpdcon`, `proflubckdengpdcon`, `proflubckdengpdcon`, `proflubckdengpdcon`, `proflubckdengpdcon`, `proflubckdengpdcon`; `fbckdengpd`, `fbckdengpd`, `fbckdengpd`, `fbckdengpd`, `fbckdengpd`, `lbckdengpd`, `lbckdengpd`, `lbckdengpd`, `lbckdengpd`, `lbckdengpd`, `nlbckdengpd`, `nlbckdengpd`, `nlbckdengpd`, `nlbckdengpd`, `nlbckdengpd`, `nlubckdengpd`, `nlubckdengpd`, `nlubckdengpd`, `nlubckdengpd`, `proflubckdengpd`, `proflubckdengpd`, `proflubckdengpd`, `proflubckdengpd`; `fkdengpdcon`, `fkdengpdcon`, `fkdengpdcon`, `lkdengpdcon`, `lkdengpdcon`, `lkdengpdcon`, `lkdengpdcon`, `lkdengpdcon`, `nlkdengpdcon`, `nlkdengpdcon`, `nlkdengpdcon`, `nlkdengpdcon`, `nlkdengpdcon`, `nlkdengpdcon`, `nlukdengpdcon`, `nlukdengpdcon`, `nlukdengpdcon`, `nlukdengpdcon`, `proflukdengpdcon`, `proflukdengpdcon`, `proflukdengpdcon`

Examples

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

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

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

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

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

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

## End(Not run)
```

fkdengpdcon

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

Description

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

Usage

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

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

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

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

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

Arguments

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

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

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

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

The possible kernels are also defined in [kernels](#) with the "gaussian" as the default choice.

Value

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

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

Note

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

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

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

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
- http://en.wikipedia.org/wiki/Kernel_density_estimation
- [http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/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`: [fbckdengpdcon](#), [fbckdengpdcon](#), [fbckdengpdcon](#), [fbckdengpdcon](#), [fbckdengpdcon](#), [lbckdengpdcon](#), [lbckdengpdcon](#), [lbckdengpdcon](#), [lbckdengpdcon](#), [lbckdengpdcon](#), [nlbckdengpdcon](#), [nlbckdengpdcon](#), [nlbckdengpdcon](#), [nlbckdengpdcon](#), [nlbckdengpdcon](#), [nlubckdengpdcon](#), [nlubckdengpdcon](#), [nlubckdengpdcon](#), [nlubckdengpdcon](#), [nlubckdengpdcon](#),

```
proflubckdengpdcon, proflubckdengpdcon, proflubckdengpdcon, proflubckdengpdcon, proflubckdengpdcon;
fbckdengpd, fbckdengpd, fbckdengpd, fbckdengpd, fbckdengpd, lbckdengpd, lbckdengpd,
lbckdengpd, lbckdengpd, lbckdengpd, nlbckdengpd, nlbckdengpd, nlbckdengpd, nlbckdengpd,
nlbckdengpd, nlubckdengpd, nlubckdengpd, nlubckdengpd, nlubckdengpd, nlubckdengpd, proflubckdengpd,
proflubckdengpd, proflubckdengpd, proflubckdengpd, proflubckdengpd; fkdengpd, fkdengpd,
fkdengpd, fkdengpd, fkdengpd, lkdengpd, lkdengpd, lkdengpd, lkdengpd, lkdengpd, nlkdengpd,
nlkdengpd, nlkdengpd, nlkdengpd, nlkdengpd, nlukdengpd, nlukdengpd, nlukdengpd, nlukdengpd,
nlukdengpd, proflukdengpd, proflukdengpd, proflukdengpd, proflukdengpd, proflukdengpd
```

Examples

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

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

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

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

Description

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

Usage

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

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

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

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

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

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
lnmean	scalar mean on log scale
lnsd	scalar standard deviation on log scale (positive)
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

Non-positive data are ignored.

Value

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

call:	optim call
x:	data vector x
init:	pvector
fixedu:	fixed threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
nlhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nlh:	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.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 parameters above threshold.


```

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

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

## End(Not run)

```

flognormgpdcon	<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, lnsd = 1, u = qlnorm(0.9, lnmean, lnsd),
  xi = 0, phiu = TRUE, log = TRUE)

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

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

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

```

Arguments

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

pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
lnmean	scalar mean on log scale
lnsd	scalar standard deviation on log scale (positive)
u	scalar threshold value
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type help `fnormgpd`. Only the different features are outlined below for brevity.

The GPD sigmau parameter is now specified as function of other parameters, see help for [dlognormgpdcon](#) for details, type help `lognormgpdcon`. Therefore, sigmau should not be included in the parameter vector if initial values are provided, making the full parameter vector (lnmean, lnstd, u, xi) if threshold is also estimated and (lnmean, lnstd, 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
nlhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nlh:	minimum negative log-likelihood
n:	total sample size
lnmean:	MLE of log-normal mean
lnstd:	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


```

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

# Continuity constraint
fit = flognormgpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 10), ylim = c(0, 0.8))
lines(xx, y)
with(fit, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = flognormgpd(x, phiu = FALSE)
with(fit2, lines(xx, dlognormgpdcon(xx, lnmean, lnsd, 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)
```

```
nlgamma(pvector, x, M, finitelik = FALSE)
```

```
nLEMgamma(pvector, tau, mgweight, x, M, finitelik = FALSE)
```

Arguments

x	vector of sample data
M	number of gamma components in mixture
pvector	vector of initial values of GPD parameters (sigmau, xi) or NULL
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
mgshape	mgamma shape (positive) as vector of length M
mgscale	mgamma scale (positive) as vector of length M
mgweight	mgamma weights (positive) as vector of length M
log	logical, if TRUE then log-likelihood rather than likelihood is output
tau	matrix of posterior probability of being in each component (nxM where n is length(x))

Details

The weighted mixture of gammas distribution is fitted to the entire dataset by maximum likelihood estimation using the EM algorithm. The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

The expectation step estimates the expected probability of being in each component conditional on gamma component parameters. The maximisation step optimizes the negative log-likelihood conditional on posterior probabilities of each observation being in each component.

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector, as often there are numerous local modes. This is an inherent feature of such models and the EM algorithm. The EM algorithm is guaranteed to reach the maximum of the local mode. Multiple initial values should be considered to find the global maximum. If the pvector is input as NULL then random component probabilities are simulated as the initial values, so multiple such runs should be run to check the sensitivity to initial values. Alternatives to black-box likelihood optimisers (e.g. simulated annealing), or moving to computational Bayesian inference, are also worth considering.

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood functions [nlmgamma](#) and [nlEMmgamma](#).

Log-likelihood calculations are carried out in [lmgamma](#), which takes parameters as inputs in the same form as the distribution functions. The negative log-likelihood function [nlmgamma](#) is a wrapper for [lmgamma](#) designed towards making it useable for optimisation, i.e. [nlmgamma](#) has complete parameter vector as first input. Similarly, for the maximisation step negative log-likelihood [nlEMmgamma](#), which also has the second input as the component probability vector mgweight.

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

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

The default optimisation algorithm in the "maximisation step" is "BFGS", which requires a finite negative log-likelihood function evaluation finitelik=TRUE. For invalid parameters, a zero likelihood is replaced with $\exp(-1e6)$. The "BFGS" optimisation algorithms require finite values for

likelihood, so any user input for `finitelik` will be overridden and set to `finitelik=TRUE` if either of these optimisation methods is chosen.

It will display a warning for non-zero convergence result comes from `optim` function call or for common indicators of lack of convergence (e.g. any estimated parameters same as initial values).

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

Suppose there are M gamma components with (scalar) shape and scale parameters and weight for each component. Only $M - 1$ are to be provided in the initial parameter vector, as the M th components weight is uniquely determined from the others.

For the fitting function `fmgamma` and negative log-likelihood functions the parameter vector `pvector` is a $3*M-1$ length vector containing all M gamma component shape parameters first, followed by the corresponding M gamma scale parameters, then all the corresponding $M - 1$ probability weight parameters. The full parameter vector is then `c(mgshape, mgscale, mgweight[1:(M-1)])`.

For the maximisation step negative log-likelihood functions the parameter vector `pvector` is a $2*M$ length vector containing all M gamma component shape parameters first followed by the corresponding M gamma scale parameters. The partial parameter vector is then `c(mgshape, mgscale)`.

For identifiability purposes the mean of each gamma component must be in ascending in order. If the initial parameter vector does not satisfy this constraint then an error is given.

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

Value

Log-likelihood is given by `lmgamma` and it's wrapper for negative log-likelihood from `nlmgamma`. The conditional negative log-likelihood using the posterior probabilities is given by `nlEMmgamma`. Fitting function `fmgammagpd` using EM algorithm returns a simple list with the following elements

<code>call:</code>	<code>optim</code> call
<code>x:</code>	data vector <code>x</code>
<code>init:</code>	<code>pvector</code>
<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>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>EMresults:</code>	EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation st
<code>posterior:</code>	posterior probabilities

Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

Description

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

Usage

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

lmgammagpd(x, mgshape, mgscale, mgweight, u, sigmau, xi, phiu = TRUE,
  log = TRUE)

nlmgammagpd(pvector, x, M, phiu = TRUE, finitelik = FALSE)

nlumgammagpd(pvector, u, x, M, phiu = TRUE, finitelik = FALSE)

nlEMmgammagpd(pvector, tau, mgweight, x, M, phiu = TRUE, finitelik = FALSE)

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

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

Arguments

x	vector of sample data
M	number of gamma components in mixture
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
mgshape	mgamma shape (positive) as vector of length M
mgscale	mgamma scale (positive) as vector of length M
mgweight	mgamma weights (positive) as vector of length M
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter

log	logical, if TRUE then log-likelihood rather than likelihood is output
tau	matrix of posterior probability of being in each component (nxM where n is length(x))

Details

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

See help for [fnormmgpd](#) for details, type help `fnormmgpd`. 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 `nlummgammagpd`. The conditional negative log-likelihoods using the posterior probabilities are `nlEMmgammagpd` and `nluEMmgammagpd`. Profile likelihood for single threshold given by `proflummgammagpd` 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 <code>x</code>
<code>init:</code>	pvector
<code>fixedu:</code>	fixed threshold, logical
<code>useq:</code>	threshold vector for profile likelihood or scalar for fixed threshold
<code>nllhuseq:</code>	profile negative log-likelihood at each threshold in <code>useq</code>
<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>	<code>phiu</code> 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

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

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

Note

In the fitting and profile likelihood functions, when `pvector=NULL` then the default initial values are obtained under the following scheme:


```

qmgammagpd, qmgammagpd, qmgammagpd, rmgammagpd, rmgammagpd, rmgammagpd, rmgammagpd,
rmgammagpd; dmgamma, dmgamma, dmgamma, dmgamma, dmgamma, mgamma, mgamma, mgamma, mgamma,
mgamma, pmgamma, pmgamma, pmgamma, pmgamma, pmgamma, qmgamma, qmgamma, qmgamma, qmgamma,
qmgamma, rmgamma, rmgamma, rmgamma, rmgamma, rmgamma; fgammagpdcon, fgammagpdcon, fgammagpdcon,
fgammagpdcon, fgammagpdcon, lgammagpdcon, lgammagpdcon, lgammagpdcon, lgammagpdcon,
lgammagpdcon, nlammagpdcon, nlammagpdcon, nlammagpdcon, nlammagpdcon, nlammagpdcon, nlammagpdcon,
nlugammagpdcon, nlugammagpdcon, nlugammagpdcon, nlugammagpdcon, nlugammagpdcon, proflugammagpdcon,
proflugammagpdcon, proflugammagpdcon, proflugammagpdcon, proflugammagpdcon; fgammagpd,
fgammagpd, fgammagpd, fgammagpd, lgammagpd, lgammagpd, lgammagpd, lgammagpd,
lgammagpd, nlammagpd, nlammagpd, nlammagpd, nlammagpd, nlammagpd, nlugammagpd,
nlugammagpd, nlugammagpd, nlugammagpd, nlugammagpd, proflugammagpd, proflugammagpd,
proflugammagpd, proflugammagpd, proflugammagpd; fmgammagpdcon, fmgammagpdcon, fmgammagpdcon,
fmgammagpdcon, fmgammagpdcon, lmgammagpdcon, lmgammagpdcon, lmgammagpdcon, lmgammagpdcon,
lmgammagpdcon, nlEMmgammagpdcon, nlEMmgammagpdcon, nlEMmgammagpdcon, nlEMmgammagpdcon, nlEMmgammagpdcon,
nlEMmgammagpdcon, nlmgammagpdcon, nlmgammagpdcon, nlmgammagpdcon, nlmgammagpdcon, nlmgammagpdcon,
nlmgammagpdcon, nlmgammagpdcon, nluEMmgammagpdcon, nluEMmgammagpdcon, nluEMmgammagpdcon,
nluEMmgammagpdcon, nluEMmgammagpdcon, nluEMmgammagpdcon, nlumgammagpdcon, nlumgammagpdcon,
nlumgammagpdcon, nlumgammagpdcon, nlumgammagpdcon, nlumgammagpdcon, proflumgammagpdcon,
proflumgammagpdcon, proflumgammagpdcon, proflumgammagpdcon, proflumgammagpdcon,
proflumgammagpdcon, proflumgammagpdcon, proflumgammagpdcon; fmgamma, fmgamma, fmgamma,
fmgamma, lmgamma, lmgamma, lmgamma, lmgamma, lmgamma, nlEMmgamma, nlEMmgamma, nlEMmgamma, nlEMmgamma,
nlmgamma, nlmgamma, nlmgamma, nlmgamma

```

Examples

```

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

n=1000
x = c(rgamma(n*0.25, shape = 1, scale = 1), rgamma(n*0.75, shape = 6, scale = 2))
xx = seq(-1, 40, 0.01)
y = (0.25*dgamma(xx, shape = 1, scale = 1) + 0.75 * dgamma(xx, shape = 6, scale = 2))

# Bulk model based tail fraction
# very sensitive to initial values, so best to provide sensible ones
fit.noinit = fmgammagpd(x, M = 2)
fit.withinit = fmgammagpd(x, M = 2, pvector = c(1, 6, 1, 2, 0.5, 15, 4, 0.1))
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, y)
with(fit.noinit, lines(xx, 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	<i>MLE Fitting of Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint using the EM algorithm.</i>
---------------	---

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)
```

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

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

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

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

Arguments

x	vector of sample data
M	number of gamma components in mixture

phiu	probability of being above threshold (0, 1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
mgshape	mgamma shape (positive) as vector of length M
mgscale	mgamma scale (positive) as vector of length M
mgweight	mgamma weights (positive) as vector of length M
u	scalar threshold value
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output
tau	matrix of posterior probability of being in each component (nxM where n is length(x))

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

The expectation step estimates the expected probability of being in each component conditional on gamma component parameters. The maximisation step optimizes the negative log-likelihood conditional on posterior probabilities of each observation being in each component.

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector, as often there are numerous local modes. This is an inherent feature of such models and the EM algorithm. The EM algorithm is guaranteed to reach the maximum of the local mode. Multiple initial values should be considered to find the global maximum. If the pvector is input as NULL then random component probabilities are simulated as the initial values, so multiple such runs should be run to check the sensitivity to initial values. Alternatives to black-box likelihood optimisers (e.g. simulated annealing), or moving to computational Bayesian inference, are also worth considering.

The log-likelihood functions are provided for wider usage, e.g. constructing profile likelihood functions. The parameter vector pvector must be specified in the negative log-likelihood functions [nlmgammagpdcon](#) and [nlEMmgammagpdcon](#).

Log-likelihood calculations are carried out in [lmgammagpdcon](#), which takes parameters as inputs in the same form as the distribution functions. The negative log-likelihood function [nlmgammagpdcon](#) is a wrapper for [lmgammagpdcon](#) designed towards making it useable for optimisation, i.e. [nlmgammagpdcon](#)

has complete parameter vector as first input. Though it is not directly used for optimisation here, as the EM algorithm due to mixture of gammas for the bulk component of this model

The EM algorithm for the mixture of gammas utilises the negative log-likelihood function [nlEMmgammagpdcon](#) which takes the posterior probabilities *tau* and component probabilities *mgweight* as secondary inputs.

The profile likelihood for the threshold [proflumgammagpdcon](#) also implements the EM algorithm for the mixture of gammas, utilising the negative log-likelihood function [nluEMmgammagpdcon](#) which takes the threshold, posterior probabilities *tau* and component probabilities *mgweight* as secondary inputs.

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

Suppose there are M gamma components with (scalar) shape and scale parameters and weight for each component. Only $M - 1$ are to be provided in the initial parameter vector, as the M th components weight is uniquely determined from the others.

The initial parameter vector *pvector* always has the M gamma component shape parameters followed by the corresponding M gamma scale parameters. However, subsets of the other parameters are needed depending on which function is being used:

- [fmgammagpdcon](#) - `c(mgshape, mgscale, mgweight[1:(M-1)], u, xi)`
- [nlmgammagpdcon](#) - `c(mgshape, mgscale, mgweight[1:(M-1)], u, xi)`
- [nlumgammagpdcon](#) and [proflumgammagpdcon](#) - `c(mgshape, mgscale, mgweight[1:(M-1)], xi)`
- [nlEMmgammagpdcon](#) - `c(mgshape, mgscale, u, xi)`
- [nluEMmgammagpdcon](#) - `c(mgshape, mgscale, xi)`

Notice that when the component probability weights are included only the first $M - 1$ are specified, as the remaining one can be uniquely determined from these. Where some parameters are left out, they are always taken as secondary inputs to the functions.

For identifiability purposes the mean of each gamma component must be in ascending in order. If the initial parameter vector does not satisfy this constraint then an error is given.

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

Value

Log-likelihood is given by [lmgammagpdcon](#) and it's wrappers for negative log-likelihood from [nlmgammagpdcon](#) and [nlumgammagpdcon](#). The conditional negative log-likelihoods using the posterior probabilities are [nlEMmgammagpdcon](#) and [nluEMmgammagpdcon](#). Profile likelihood for single threshold given by [proflumgammagpdcon](#) using EM algorithm. Fitting function [fmgammagpdcon](#) using EM algorithm returns a simple list with the following elements

<code>call:</code>	optim call
<code>x:</code>	data vector <i>x</i>
<code>init:</code>	<i>pvector</i>
<code>fixedu:</code>	fixed threshold, logical
<code>useq:</code>	threshold vector for profile likelihood or scalar for fixed threshold
<code>nllhuseq:</code>	profile negative log-likelihood at each threshold in <i>useq</i>
<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>	<i>phi</i> u to be consistent with evd
<code>nllh:</code>	minimum negative log-likelihood

n: total sample size
 M: number of gamma components
 mgshape: MLE of gamma shapes
 mgscale: MLE of gamma scales
 mgweight: MLE of gamma weights
 u: threshold (fixed or MLE)
 sigmau: MLE of GPD scale
 xi: MLE of GPD shape
 phiu: MLE of tail fraction (bulk model or parameterised approach)
 se.phiu: standard error of MLE of tail fraction
 EMresults: EM results giving complete negative log-likelihood, estimated parameters and conditional "maximisation s
 posterior: posterior probabilities

Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

In the fitting and profile likelihood functions, when `pvector=NULL` then the default initial values are obtained under the following scheme:

- number of sample from each component is simulated from symmetric multinomial distribution;
- sample data is then sorted and split into groups of this size (works well when components have modes which are well separated);
- for data within each component approximate MLE's for the gamma shape and scale parameters are estimated;
- threshold is specified as sample 90% quantile; and
- MLE of GPD shape parameter above threshold.

The other likelihood functions [lmgammagpdcon](#), [nlmgammagpdcon](#), [nlumgammagpdcon](#) and [nlEMmgammagpdcon](#) and [nluEMmgammagpdcon](#) have no defaults.

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
http://en.wikipedia.org/wiki/Gamma_distribution
http://en.wikipedia.org/wiki/Mixture_model
http://en.wikipedia.org/wiki/Generalized_Pareto_distribution
 McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.
 Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>


```

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

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

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
nmean	scalar normal mean
nsd	scalar normal standard deviation (positive)
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

The optimisation of the likelihood for these mixture models can be very sensitive to the initial parameter vector (particularly the threshold), as often there are numerous local modes where multiple thresholds give similar fits. This is an inherent feature of such models. Options are provided by the arguments pvector, useq and fixedu to implement various commonly used likelihood inference approaches for such models:

1. (default) pvector=NULL, useq=NULL and fixedu=FALSE - to set initial value for threshold at 90% quantile along with usual defaults for other parameters as defined in Notes below. Standard likelihood optimisation is used;
2. pvector=c(nmean, nsd, u, sigmau, xi) - where initial values of all 5 parameters are manually set. Standard likelihood optimisation is used;

3. useq as vector - to specify a sequence of thresholds at which to evaluate profile likelihood and extract threshold which gives maximum profile likelihood; or
4. useq as scalar - to specify a single value for threshold to be considered.

In options (3) and (4) the threshold can be treated as:

- initial value for maximum likelihood estimation when fixedu=FALSE, using either profile likelihood estimate (3) or pre-chosen threshold (4); or
- a fixed threshold with MLE for other parameters when fixedu=TRUE, using either profile likelihood estimate (3) or pre-chosen threshold (4).

The latter approach can be used to implement the traditional fixed threshold modelling approach with threshold pre-chosen using, for example, graphical diagnostics. Further, in either such case (3) or (4) the pvector could be:

- NULL for usual defaults for other four parameters, defined in Notes below; or
- vector of initial values for remaining 4 parameters (nmean, nsd, sigmau, xi).

If the threshold is treated as fixed, then the likelihood is separable between the bulk and tail components. However, in practice we have found black-box optimisation of the combined likelihood works sufficiently well, so is used herein.

The following functions are provided:

- `fnormgpd` - maximum likelihood fitting with all the above options;
- `lnormgpd` - log-likelihood;
- `nlnormgpd` - negative log-likelihood;
- `proflunormgpd` - profile likelihood for given threshold; and
- `nlunormgpd` - negative log-likelihood (threshold specified separately).

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

Defaults values for the parameter vector pvector are given in the fitting `fnormgpd` and profile likelihood functions `proflunormgpd`. The parameter vector pvector must be specified in the negative log-likelihood functions `nlnormgpd` and `nlunormgpd`. The threshold u must also be specified in the profile likelihood function `proflunormgpd` and `nlunormgpd`.

Log-likelihood calculations are carried out in `lnormgpd`, which takes parameters as inputs in the same form as distribution functions. The negative log-likelihood functions `nlnormgpd` and `nlunormgpd` are wrappers for likelihood function `lnormgpd` designed towards optimisation, i.e. `nlnormgpd` has vector of all 5 parameters as first input and `nlunormgpd` has threshold as second input and vector of remaining 4 parameters as first input. The profile likelihood function `proflunormgpd` has threshold u as the first input, to permit use of `sapply` function to evaluate profile likelihood over vector of potential thresholds.

The tail fraction phiu is treated separately to the other parameters, to allow for all 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>	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>nllhuseq:</code>	profile negative log-likelihood at each threshold in useq
<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>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

These functions are deliberately similar in syntax and functionality to the commonly used functions in the `ismev` and `evd` packages for which their author's contributions are gratefully acknowledged.

Anna MacDonald and Xin Zhao laid some of the groundwork with programs they wrote for MATLAB.

Clement Lee and Emma Eastoe suggested providing inbuilt profile likelihood estimation for threshold and fixed threshold approach.

Note

Unlike most of the distribution functions for the extreme value mixture models, the MLE fitting only permits single scalar values for each parameter and `phiu`.

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

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

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

A default value for the tail fraction `phiu=TRUE` is given. The `lnormgpd` also has the usual defaults for the other parameters, but `nlnormgpd` and `nlunormgpd` has no defaults.

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

Invalid parameter ranges will give 0 for likelihood, $\log(0)=-\text{Inf}$ 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.


```

fitu = fnormgpd(x, useq = seq(0, 3, length = 20))
fitfix = fnormgpd(x, useq = seq(0, 3, length = 20), fixedu = TRUE)

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

## End(Not run)

```

fnormgpdcon

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

Description

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

Usage

```

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

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

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

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

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

```

Arguments

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

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

Details

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

See help for [fnormgpd](#) for full details, type help fnormgpd. Only the different features are outlined below for brevity.

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

Value

Log-likelihood is given by [lnormgpdcon](#) and it's wrappers for negative log-likelihood from [nlnormgpdcon](#) and [nlnunormgpdcon](#). Profile likelihood for single threshold given by [proflunormgpdcon](#). Fitting function [fnormgpdcon](#) returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
fixedu:	fixed threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
nllhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
n:	total sample size
nmean:	MLE of normal mean
nsd:	MLE of normal standard deviation
u:	threshold (fixed or MLE)
sigmau:	MLE of GPD scale (estimated from other parameters)
xi:	MLE of GPD shape
phiu:	MLE of tail fraction (bulk model or parameterised approach)
se.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`: [dgngcon](#), [dgngcon](#), [dgngcon](#), [dgngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#); [dgng](#), [dgng](#), [dgng](#), [dgng](#), [dgng](#), [gng](#), [gng](#), [gng](#), [gng](#), [gng](#), [pgng](#), [pgng](#), [pgng](#), [pgng](#), [pgng](#), [qngng](#), [qngng](#), [qngng](#), [qngng](#), [qngng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#), [rgng](#); [ditmgng](#), [ditmgng](#), [ditmgng](#), [ditmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#), [ritmgng](#); [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [dnormgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [normgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [pnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [qnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#), [rnormgpdcon](#); [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [dnormgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [normgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [pnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [qnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#), [rnormgpd](#); [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [fgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [lgngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlngngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [nlugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#), [proflugngcon](#); [fgng](#), [fgng](#), [fgng](#), [fgng](#), [fgng](#), [lgng](#), [lgng](#), [lgng](#), [lgng](#), [lgng](#), [nlngng](#), [nlngng](#), [nlngng](#), [nlngng](#), [nlngng](#), [nlugng](#), [nlugng](#), [nlugng](#), [nlugng](#), [nlugng](#), [nlugng](#).

```

proflugng, proflugng, proflugng, proflugng, proflugng; fitmgng, fitmgng, fitmgng, fitmgng,
fitmgng, litmgng, litmgng, litmgng, litmgng, litmgng, nleuitmgng, nlitmgng, nlitmgng,
nlitmgng, nlitmgng, nlitmgng, nluitmgng, nluitmgng, nluitmgng, nluitmgng, nluitmgng,
profleuitmgng, profluitmgng, profluitmgng, profluitmgng, profluitmgng, profluitmgng;
fnormgpd, fnormgpd, fnormgpd, fnormgpd, fnormgpd, lnormgpd, lnormgpd, lnormgpd, lnormgpd,
lnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlunormgpd, nlunormgpd,
nlunormgpd, nlunormgpd, nlunormgpd, proflunormgpd, proflunormgpd, proflunormgpd, proflunormgpd,
proflunormgpd

```

Examples

```

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

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

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

# No continuity constraint
fit2 = fnormgpd(x)
with(fit2, lines(xx, dnormgpd(xx, nmean, nsd, u, 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)

```

Description

Maximum likelihood estimation for P-splines density estimation. Histogram binning produces frequency counts, which are modelled by constrained B-splines in a Poisson regression. A penalty based on differences in the sequences B-spline coefficients is used to smooth/interpolate the counts. Iterated weighted least squares (IWLS) for a mixed model representation of the P-splines regression, conditional on a particular penalty coefficient, is used for estimating the B-spline coefficients. Leave-one-out cross-validation deviances are available for estimation of the penalty coefficient.

Usage

```
fpsden(x, lambdaseq = NULL, breaks = NULL, xrange = NULL, nseg = 10,
       degree = 3, design.knots = NULL, ord = 2)

lpsden(x, beta = NULL, bsplines = NULL, nbinwidth = 1, log = TRUE)

nlpsden(pvector, x, bsplines = NULL, nbinwidth = 1, finitelik = FALSE)

cvpsden(lambda = 1, counts, bsplines, ord = 2)

iwlpsden(counts, bsplines, ord = 2, lambda = 10)
```

Arguments

x	quantiles
lambdaseq	vector of λ 's (or scalar) to be considered in profile likelihood. Required.
breaks	histogram breaks (as in hist function)
xrange	vector of minimum and maximum of B-spline (support of density)
nseg	number of segments between knots
degree	degree of B-splines (0 is constant, 1 is linear, etc.)
design.knots	spline knots for <code>splineDesign</code> function
ord	order of difference used in the penalty term
beta	vector of B-spline coefficients (required)
bsplines	matrix of B-splines
nbinwidth	scaling to convert count frequency into proper density
log	logical, if TRUE then log density
pvector	vector of initial values of GPD parameters (σ , ξ) or NULL
finitelik	logical, should log-likelihood return finite value for invalid parameters
lambda	penalty coefficient
counts	counts from histogram binning

Details

The P-splines density estimator is fitted using maximum likelihood estimation, following the approach of Eilers and Marx (1996). Histogram binning produces frequency counts, which are modelled by constrained B-splines in a Poisson regression. A penalty based on differences in the sequences B-spline coefficients is used to smooth/interpolate the counts.

The B-splines are defined as in Eiler and Marx (1996), so that those are meet the boundary are simply shifted and truncated version of the internal B-splines. No renormalisation is carried out.

They are not "natural" B-spline which are also commonly in use. Note that atural B-splines can be obtained by suitable linear combinations of these B-splines. Hence, in practice there is little difference in the fit obtained from either B-spline definition, even with the penalty constraining the coefficients. If the user desires they can force the use of natural B-splines, by prior specification of the design.knots with appropriate replication of the boundaries, see [dpsden](#).

Iterated weighted least squares (IWLS) for a mixed model representation of the P-splines regression, conditional on a particular penalty coefficient, is used for estimating the B-spline coefficients which is equivalent to maximum likelihood estimation. Leave-one-out cross-validation deviances are available for estimation of the penalty coefficient.

The parameter vector is the B-spline coefficients beta, no matter whether the penalty coefficient is fixed or estimated. The penalty coefficient lambda is treated separately.

The log-likelihood functions [lpsden](#) and [nlpsden](#) evaluate the likelihood for the original dataset, using the fitted P-splines density estimator. The log-likelihood is output as nllh from the fitting function [fpsden](#). They do not provide the likelihood for the Poisson regression of the histogram counts, which is usually evaluated using the deviance. The deviance (via CVMSE for Poisson counts) is also output as cvlambda from the fitting function [fpsden](#).

The [iwlpsden](#) function performs the IWLS. The [cvpsden](#) function calculates the leave-one-out cross-validation sum of the squared errors. They are not designed to be used directly by users. No checks of the inputs are carried out.

Value

Log-likelihood for original data is given by [lpsden](#) and it's wrappers for negative log-likelihood from [nlpsden](#). Cross-validation sum of square of errors is provided by [cvpsden](#). Poisson regression fitting by IWLS is carried out in [iwlpsden](#). Fitting function [fpsden](#) returns a simple list with the following elements

call:	optim call
x:	data vector x
xrange:	range of support of B-splines
degree:	degree of B-splines
nseg:	number of internal segments
design.knots:	knots used in splineDesign
ord:	order of penalty term
binned:	histogram results
breaks:	histogram breaks
mids:	histogram mid-bins
counts:	histogram counts
nbinwidth:	scaling factor to convert counts to density
bsplines:	B-splines matrix used for binned counts
databsplines:	B-splines matrix used for data
counts:	histogram counts
lambdaseq:	λ vector for profile likelihood or scalar for fixed λ
cvlambda:	CV MSE for each λ
mle and beta:	vector of MLE of coefficients
nllh:	negative log-likelihood for original data
n:	total original sample size
lambda:	Estimated or fixed λ

Acknowledgments

The Poisson regression and leave-one-out cross-validation functions are based on the code of Eilers and Marx (1996) available from Brian Marx's website <http://www.stat.lsu.edu/faculty/marx>, which is gratefully acknowledged.

Note

The data are both vectors. Infinite and missing sample values are dropped.

No initial values for the coefficients are needed.

It is advised to specify the range of support xrange, using finite end-points. This is especially important when the support is bounded. By default xrange is simply the range of the input data range(x).

Further, it is advised to always set the histogram bin breaks, especially if the support is bounded. By default $10 \cdot \ln(n)$ equi-spaced bins are defined between xrange.

Author(s)

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

References

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

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

<http://en.wikipedia.org/wiki/B-spline>

<http://www.stat.lsu.edu/faculty/marx>

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science* 11(2), 89-121.

See Also

[kden](#).

Other psden fpsden: [dpsden](#), [dpsden](#), [dpsden](#), [dpsden](#), [dpsden](#), [ppsden](#), [ppsden](#), [ppsden](#), [ppsden](#), [ppsden](#), [psden](#), [psden](#), [psden](#), [psden](#), [psden](#), [psden](#), [qpsden](#), [qpsden](#), [qpsden](#), [qpsden](#), [qpsden](#), [rpsden](#), [rpsden](#), [rpsden](#), [rpsden](#)

Examples

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

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

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 10 internal segments
# CV search for penalty coefficient.
fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
```

```

psdensity = exp(fit$bsplines %>% fit$mle)

hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

lines(fit$mids, psdensity/fit$nbwidth, lwd = 2, col = "blue") # P-splines density

# check density against dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbwidth, design = design.knots),
                lwd = 2, col = "red", lty = 2))

# vertical lines for all knots
with(fit, abline(v = design.knots, col = "red"))

# internal knots
with(fit, abline(v = design.knots[(degree + 2):(length(design.knots) - degree - 1)], col = "blue"))

# boundary knots (support of B-splines)
with(fit, abline(v = design.knots[c(degree + 1, length(design.knots) - degree)], col = "green"))

legend("topright", c("True Density", "P-spline density", "Using dpsdens function"),
      col=c("black", "blue", "red"), lty = c(1, 1, 2))
legend("topleft", c("Internal Knots", "Boundaries", "Extra Knots"),
      col=c("blue", "green", "red"), lty = 1)

## End(Not run)

```

fpsdengpd

MLE Fitting of P-splines Density Estimate for Bulk and GPD Tail Extreme Value Mixture Model

Description

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

Usage

```

fpsdengpd(x, phiu = TRUE, useq = NULL, fixedu = FALSE, pvector = NULL,
          lambdaseq = NULL, breaks = NULL, xrange = NULL, nseg = 10,
          degree = 3, design.knots = NULL, ord = 2, std.err = TRUE,
          method = "BFGS", control = list(maxit = 10000), finitelik = TRUE, ...)

lpsdengpd(x, psdenx, u = NULL, sigmau = NULL, xi = 0, phiu = TRUE,
          bsplinefit = NULL, phib = NULL, log = TRUE)

nlpsdengpd(pvector, x, psdenx, phiu = TRUE, bsplinefit, phib = NULL,
          finitelik = FALSE)

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

```

```
nlupsdengpd(pvector, u, x, psdenx, phiu = TRUE, bsplinefit = bsplinefit,
  phib = NULL, finitelik = FALSE)
```

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
lambdaseq	vector of λ 's (or scalar) to be considered in profile likelihood. Required.
breaks	histogram breaks (as in hist function)
xrange	vector of minimum and maximum of B-spline (support of density)
nseg	number of segments between knots
degree	degree of B-splines (0 is constant, 1 is linear, etc.)
design.knots	spline knots for splineDesign function
ord	order of difference used in the penalty term
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
psdenx	P-splines based density estimate for each datapoint in x
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
bsplinefit	list output from P-splines density fitting fpsden function
phib	renormalisation constant for bulk model density $(1 - \phi_u)/H(u)$, to make it integrate to 1-phiu
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

The extreme value mixture model with P-splines density estimate for bulk and GPD tail is fitted to the entire dataset. A two-stage maximum likelihood inference approach is taken. The first stage consists fitting of the P-spline density estimator, which is achieved by MLE using the [fpsden](#) function. The second stage, conditions on the B-spline coefficients, using MLE for the extreme value mixture model (GPD parameters and threshold, if requested). The estimated parameters, variance-covariance matrix and their standard errors are automatically output.

See help for [fnormgpd](#) for details of extreme value mixture models, type `help fnormgpd`. Only the different features are outlined below for brevity.

As the second stage conditions on the B-spline coefficients, the full parameter vector is (u, sigmau, xi) if threshold is also estimated and (sigmau, xi) for profile likelihood or fixed threshold approach. (Penalized) MLE estimation of the B-Spline coefficients is carried out using Poisson regression based on histogram bin counts. See help for [fpsden](#) for details, type `help fpsden`.

Value

Log-likelihood is given by `lpsdengpd` and its wrappers for negative log-likelihood from `nlpsdengpd` and `nlupsdengpd`. Profile likelihood for single threshold given by `proflupsdengpd`. Fitting function `fpsdengpd` returns a simple list with the following elements

<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>nllhuseq:</code>	profile negative log-likelihood at each threshold in <code>useq</code>
<code>bsplinefit:</code>	complete <code>fpsden</code> output
<code>psdenx:</code>	P-splines based density estimate for each datapoint in x
<code>xrange:</code>	range of support of B-splines
<code>degree:</code>	degree of B-splines
<code>nseg:</code>	number of internal segments
<code>design.knots:</code>	knots used in <code>splineDesign</code>
<code>nbinwidth:</code>	scaling factor to convert counts to density
<code>optim:</code>	complete optim output
<code>conv:</code>	indicator for "possible" convergence
<code>mle:</code>	vector of MLE of (GPD and threshold, if relevant) 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>	ϕ_{iu} to be consistent with <code>evd</code>
<code>nllh:</code>	minimum negative log-likelihood
<code>n:</code>	total sample size
<code>beta:</code>	vector of MLE of B-spline coefficients
<code>lambda:</code>	Estimated or fixed λ
<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

Acknowledgments

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

The Poisson regression and leave-one-out cross-validation functions are based on the code of Eilers and Marx (1996) available from Brian Marx's website <http://www.stat.lsu.edu/faculty/marx>, which is gratefully acknowledged.

Note

The data are both vectors. Infinite and missing sample values are dropped.

No initial values for the coefficients are needed.

It is advised to specify the range of support `xrange`, using finite end-points. This is especially important when the support is bounded. By default `xrange` is simply the range of the input data `range(x)`.

Further, it is advised to always set the histogram bin breaks, especially if the support is bounded. By default $10 \cdot \ln(n)$ equi-spaced bins are defined between `xrange`.

When pvector=NULL then the initial values are:

- threshold 90% quantile (not relevant for profile likelihood for threshold or fixed threshold approaches);
- MLE of GPD parameters above threshold.

Author(s)

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

References

<http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
http://en.wikipedia.org/wiki/Generalized_Pareto_distribution
[http://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](http://en.wikipedia.org/wiki/Cross-validation_(statistics))
<http://en.wikipedia.org/wiki/B-spline>
<http://www.stat.lsu.edu/faculty/marx>

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science* 11(2), 89-121.

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

See Also

[fpsden](#), [fnormgpd](#), [fgpd](#) and [gpd](#)

Examples

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

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

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 10 internal segments
# CV search for penalty coefficient.
fit = fpsdengpd(x, useq = seq(0, 3, 0.1), fixedu = TRUE,
               lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
               xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)

hist(x, freq = FALSE, breaks = breaks, xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

# P-splines+GPD
with(fit, lines(xx, dpsdengpd(xx, beta, nbinwidth,
                             u = u, sigmau = sigmau, xi = xi, design = design.knots),
                             lwd = 2, col = "red"))
abline(v = fit$u, col = "red", lwd = 2, lty = 3)
```

```

# P-splines density estimate
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots),
               lwd = 2, col = "blue", lty = 2))

# vertical lines for all knots
with(fit, abline(v = design.knots, col = "red"))

# internal knots
with(fit, abline(v = design.knots[(degree + 2):(length(design.knots) - degree - 1)], col = "blue"))

# boundary knots (support of B-splines)
with(fit, abline(v = design.knots[c(degree + 1, length(design.knots) - degree)], col = "green"))

legend("topright", c("True Density", "P-spline density", "P-spline+GPD"),
      col=c("black", "blue", "red"), lty = c(1, 2, 1))
legend("topleft", c("Internal Knots", "Boundaries", "Extra Knots", "Threshold"),
      col=c("blue", "green", "red", "red"), lty = c(1, 1, 1, 2))

## End(Not run)

```

fweibullgpd

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

Description

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

Usage

```

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

lweibullgpd(x, wshape = 1, wscale = 1, u = qweibull(0.9, wshape, wscale),
            sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
            1/wshape))^2), xi = 0, phiu = TRUE, log = TRUE)

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

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

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

```

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd

useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
wshape	scalar Weibull shape (positive)
wscale	scalar Weibull scale (positive)
u	scalar threshold value
sigmau	scalar scale parameter (positive)
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

Non-positive data are ignored ($f(0)$ is infinite for $wshape < 1$).

Value

Log-likelihood is given by [lweibullgpd](#) and it's wrappers for negative log-likelihood from [nlweibullgpd](#) and [nluweibullgpd](#). Profile likelihood for single threshold given by [profluweibullgpd](#). Fitting function [fweibullgpd](#) returns a simple list with the following elements

call:	optim call
x:	data vector x
init:	pvector
fixedu:	fixed threshold, logical
useq:	threshold vector for profile likelihood or scalar for fixed threshold
nllhuseq:	profile negative log-likelihood at each threshold in useq
optim:	complete optim output
mle:	vector of MLE of parameters
cov:	variance-covariance matrix of MLE of parameters
se:	vector of standard errors of MLE of parameters
rate:	phiu to be consistent with evd
nllh:	minimum negative log-likelihood
n:	total sample size
wshape:	MLE of Weibull shape

wscale:	MLE of Weibull scale
u:	threshold (fixed or MLE)
sigmau:	MLE of GPD scale
xi:	MLE of GPD shape
phiu:	MLE of tail fraction (bulk model or parameterised approach)
se.phiu:	standard error of MLE of tail fraction

Acknowledgments

See Acknowledgments in [fnormgpd](#), type `help fnormgpd`.

Note

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

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

Author(s)

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

References

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

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

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

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

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

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

See Also

[dweibull](#), [fgpd](#) and [gpd](#)

Other `weibullgpd` `weibullgpdcon` `fweibullgpd` `fweibullgpdcon` `normgpd` `fnormgpd`: [fweibullgpdcon](#), [fweibullgpdcon](#), [fweibullgpdcon](#), [fweibullgpdcon](#), [lweibullgpdcon](#), [lweibullgpdcon](#), [lweibullgpdcon](#), [lweibullgpdcon](#), [nluweibullgpdcon](#), [nluweibullgpdcon](#), [nluweibullgpdcon](#), [nluweibullgpdcon](#), [nlweibullgpdcon](#), [nlweibullgpdcon](#), [nlweibullgpdcon](#), [profluweibullgpdcon](#), [profluweibullgpdcon](#), [profluweibullgpdcon](#)

Examples

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

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

# Bulk model based tail fraction
fit = fweibullgpd(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")

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

# Profile likelihood for initial value of threshold and fixed threshold approach
fitu = fweibullgpd(x, useq = seq(0.5, 2, length = 20))
fitfix = fweibullgpd(x, useq = seq(0.5, 2, length = 20), fixedu = TRUE)

hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="red"))
abline(v = fit$u, col = "red")
with(fitu, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="purple"))
abline(v = fitu$u, col = "purple")
with(fitfix, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi), col="darkgreen"))
abline(v = fitfix$u, col = "darkgreen")
legend("topright", c("True Density", "Default initial value (90% quantile)",
  "Prof. lik. for initial value", "Prof. lik. for fixed threshold"),
      col=c("black", "red", "purple", "darkgreen"), lty = 1)

## End(Not run)
```

fweibullgpdcon

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

Description

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

Usage

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

```
lweibullgpdcon(x, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), xi = 0, phiu = TRUE, log = TRUE)
```

```
nlweibullgpdcon(pvector, x, phiu = TRUE, finitelik = FALSE)
```

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

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

Arguments

x	vector of sample data
phiu	probability of being above threshold (0,1) or logical, see Details in help for fnormgpd
useq	vector of thresholds (or scalar) to be considered in profile likelihood or NULL for no profile likelihood
fixedu	logical, should threshold be fixed (at either scalar value in useq, or estimated from maximum of profile likelihood evaluated at sequence of thresholds in useq)
pvector	vector of initial values of parameters or NULL for default values, see below
std.err	logical, should standard errors be calculated
method	optimisation method (see optim)
control	optimisation control list (see optim)
finitelik	logical, should log-likelihood return finite value for invalid parameters
...	optional inputs passed to optim
wshape	scalar Weibull shape (positive)
wscale	scalar Weibull scale (positive)
u	scalar threshold value
xi	scalar shape parameter
log	logical, if TRUE then log-likelihood rather than likelihood is output

Details

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

See help for [fnormgpd](#) for details, type `help fnormgpd`. Only the different features are outlined below for brevity.

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

Negative data are ignored.

Value

Log-likelihood is given by [lweibullgpdcon](#) and its wrappers for negative log-likelihood from [nlweibullgpdcon](#) and [nluweibullgpdcon](#). Profile likelihood for single threshold given by [profluweibullgpdcon](#). Fitting function [fweibullgpdcon](#) returns a simple list with the following elements

```

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

Examples

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

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

# Continuity constraint
fit = fweibullgpdcon(x)
hist(x, breaks = 100, freq = FALSE, xlim = c(-0.1, 4))
lines(xx, y)
with(fit, lines(xx, dweibullgpdcon(xx, wshape, wscale, u, xi), col="red"))
abline(v = fit$u, col = "red")

# No continuity constraint
fit2 = fweibullgpd(x, phiu = FALSE)
with(fit2, lines(xx, dweibullgpd(xx, wshape, wscale, u, sigmau, xi, phiu), col="blue"))
abline(v = fit2$u, col = "blue")
legend("topright", c("True Density", "No continuity constraint", "With 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)
```


gammagpd

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

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

Usage

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

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

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

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

Arguments

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

Details

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

The user can pre-specify phiu permitting a parameterised value for the tail fraction 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>


```
nlungammagpd, proflumgammagpd, proflumgammagpd, proflumgammagpd, proflumgammagpd, proflumgammagpd,
proflumgammagpd, proflumgammagpd; fmgamma, fmgamma, fmgamma, fmgamma, lmgamma, lmgamma,
lmgamma, lmgamma, nLEMgamma, nLEMgamma, nLEMgamma, nLEMgamma, nlmgamma, nlmgamma,
nlmgamma, nlmgamma
```

Examples

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

x = rgammagpd(1000, gshape = 2)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammagpd(xx, gshape = 2))

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

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

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

## End(Not run)
```

gammagpdcon

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

Description

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

Usage

```
dgammagpdcon(x, gshape = 1, gscale = 1, u = qgamma(0.9, gshape, 1/gscale),
  xi = 0, phiu = TRUE, log = FALSE)

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

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

```
rgammagpdcon(n = 1, gshape = 1, gscale = 1, u = qgamma(0.9, gshape,
              1/gscale), xi = 0, phiu = TRUE)
```

Arguments

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

Details

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

The user can pre-specify 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 gamma bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the gamma 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 gamma and conditional GPD cumulative distribution functions (i.e. `pgamma(x, gshape, 1/gscale)` and `pgpd(x, u, sigma_u, 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 `dgp(x, u, sigmau, xi)`) respectively. The resulting GPD scale parameter is then:

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

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

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

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

gpd and dgamma

Examples

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

x = rgammagpdcon(1000, gshape = 2)
```

```

xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammapdcon(xx, gshape = 2))

# three tail behaviours
plot(xx, pgammapdcon(xx, gshape = 2), type = "l")
lines(xx, pgammapdcon(xx, gshape = 2, xi = 0.3), col = "red")
lines(xx, pgammapdcon(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 = rgammapdcon(1000, gshape = 2, u = 3, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgammapdcon(xx, gshape = 2, u = 3, phiu = 0.2))

plot(xx, dgammapdcon(xx, gshape = 2, u = 3, xi=0, phiu = 0.2), type = "l")
lines(xx, dgammapdcon(xx, gshape = 2, u = 3, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dgammapdcon(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_{maul} and shape ξ_l and tail fraction ϕ_{iul}) and upper tail (threshold u_r , GPD scale σ_{maur} and shape ξ_r and tail fraction ϕ_{iur}).

Usage

```

dgkg(x, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), sigmaur = sqrt(6 *
  var(kerncentres))/pi, xir = 0, phiur = TRUE, bw = NULL,
  kernel = "gaussian", log = FALSE)

pgkg(q, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), sigmaur = sqrt(6 *
  var(kerncentres))/pi, xir = 0, phiur = TRUE, bw = NULL,
  kernel = "gaussian", lower.tail = TRUE)

qgkg(p, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), sigmaul = sqrt(6 * var(kerncentres))/pi, xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), sigmaur = sqrt(6 *
  var(kerncentres))/pi, xir = 0, phiur = TRUE, bw = NULL,

```



```

kernel = "gaussian", lower.tail = TRUE)

rgkg(n = 1, kerncentres, lambda = NULL,
     ul = as.vector(quantile(kerncentres, 0.1)), sigmaul = sqrt(6 *
     var(kerncentres))/pi, xil = 0, phiul = TRUE,
     ur = as.vector(quantile(kerncentres, 0.9)), sigmaur = sqrt(6 *
     var(kerncentres))/pi, xir = 0, phiur = TRUE, bw = NULL,
     kernel = "gaussian")

```

Arguments

x	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
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (positive)
xil	lower tail GPD shape parameter
phiul	probability of being below lower threshold $[0, 1]$ or TRUE
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (positive)
xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold $[0, 1]$ or TRUE
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
kernel	kernel name (default = "gaussian")
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk between thresholds and GPD beyond thresholds.

The user can pre-specify phiul and phiur permitting a parameterised value for the tail fractions 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 $\phi_{ul} + \phi_{ur} < 1$, 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.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

Due to symmetry, the lower tail can be described by GPD by negating the quantiles.

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

Author(s)

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

References

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

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

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

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

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

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

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

See Also

[kernels](#), [kfun](#), [density](#), [bw.nrd0](#) and [dkde](#) in [ks](#) package.

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

Examples

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

kerncentres=rnorm(1000,0,1)
x = rgkg(1000, kerncentres, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkg(xx, kerncentres, phiul = 0.15, phiur = 0.15))

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

# asymmetric tail behaviours
x = rgkg(1000, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkg(xx, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1))
```

```

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

## End(Not run)

```

gkgcon

Kernel Density Estimate and GPD Both Upper and Lower Tails Extreme Value Mixture Model With Single Continuity Constraint at Both

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with kernel density estimate for bulk distribution between thresholds and conditional GPD beyond thresholds and continuity at both of them. The parameters are the kernel bandwidth `lambda`, lower tail (threshold `ul`, GPD shape `xil` and tail fraction `phiul`) and upper tail (threshold `ur`, GPD shape `xir` and tail fraction `phiur`).

Usage

```

dgkgcon(x, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), xil = 0, phiul = TRUE, ur = as.vector(quantile(kerncentres,
  0.9)), xir = 0, phiur = TRUE, bw = NULL, kernel = "gaussian",
  log = FALSE)

pgkgcon(q, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), xil = 0, phiul = TRUE, ur = as.vector(quantile(kerncentres,
  0.9)), xir = 0, phiur = TRUE, bw = NULL, kernel = "gaussian",
  lower.tail = TRUE)

qgkgcon(p, kerncentres, lambda = NULL, ul = as.vector(quantile(kerncentres,
  0.1)), xil = 0, phiul = TRUE, ur = as.vector(quantile(kerncentres,
  0.9)), xir = 0, phiur = TRUE, bw = NULL, kernel = "gaussian",
  lower.tail = TRUE)

rgkgcon(n = 1, kerncentres, lambda = NULL,
  ul = as.vector(quantile(kerncentres, 0.1)), xil = 0, phiul = TRUE,
  ur = as.vector(quantile(kerncentres, 0.9)), xir = 0, phiur = TRUE,
  bw = NULL, kernel = "gaussian")

```

Arguments

<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

xil	lower tail GPD shape parameter
phiul	probability of being below lower threshold $[0, 1]$ or TRUE
ur	upper tail threshold
xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold $[0, 1]$ or TRUE
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
kernel	kernel name (default = "gaussian")
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk between thresholds and GPD beyond thresholds and continuity at both of them.

The user can pre-specify phiul and phiur permitting a parameterised value for the tail fractions 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 $\phi_{ul} + \phi_{ur} < 1$, so the lower threshold must be less than the upper, $u_l < u_r$.

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 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 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 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

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

If no bandwidth is provided `lambda=NULL` and `bw=NULL` then the normal reference rule is used, using the `bw.nrd0` function, which is consistent with the `density` function. At least two kernel centres must be provided as the variance needs to be estimated.

See `gpd` for details of GPD upper tail component and `dkden` for details of KDE bulk component.

Value

`dgkgcon` gives the density, `pgkgcon` gives the cumulative distribution function, `qgkgcon` gives the quantile function and `rgkgcon` gives a random sample.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

Due to symmetry, the lower tail can be described by GPD by negating the quantiles.

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

Author(s)

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

References

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

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

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

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

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

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

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

See Also

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

Other kden kdengpd kdengpdcon gkg gkgcon bckden bckdengpd bckdengpdcon fkden fkdengpd fkdengpdcon fgkg fgkgcon fbckden fbckdengpd fbckdengpdcon: `dgkg`, `dgkg`, `dgkg`, `dgkg`, `dgkg`, `gkg`, `gkg`, `gkg`, `gkg`, `pgkg`, `pgkg`, `pgkg`, `pgkg`, `pgkg`, `pgkg`, `qgkg`, `qgkg`, `qgkg`, `qgkg`, `rgkg`, `rgkg`, `rgkg`, `rgkg`

Examples

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

kerncentres=rnorm(1000,0,1)
x = rgkgcon(1000, kerncentres, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkgcon(xx, kerncentres, phiul = 0.15, phiur = 0.15))

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

```

legend("topleft", paste("Symmetric xil=xir=",c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

# asymmetric tail behaviours
x = rgkgcon(1000, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgkgcon(xx, kerncentres, xil = -0.3, phiul = 0.1, xir = 0.3, phiur = 0.1))

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

## End(Not run)

```

gng

Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails. The parameters are the normal mean $nmean$ and standard deviation nsd , lower tail (threshold ul , GPD scale $sigmaul$ and shape xil and tail fraction $phiul$) and upper tail (threshold ur , GPD scale $sigmaur$ and shape xir and tail fraction $phiur$).

Usage

```

dgng(x, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), sigmaul = nsd,
     xil = 0, phiul = TRUE, ur = qnorm(0.9, nmean, nsd), sigmaur = nsd,
     xir = 0, phiur = TRUE, log = FALSE)

pgng(q, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), sigmaul = nsd,
     xil = 0, phiul = TRUE, ur = qnorm(0.9, nmean, nsd), sigmaur = nsd,
     xir = 0, phiur = TRUE, lower.tail = TRUE)

qgng(p, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), sigmaul = nsd,
     xil = 0, phiul = TRUE, ur = qnorm(0.9, nmean, nsd), sigmaur = nsd,
     xir = 0, phiur = TRUE, lower.tail = TRUE)

rgng(n = 1, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
     sigmaul = nsd, xil = 0, phiul = TRUE, ur = qnorm(0.9, nmean, nsd),
     sigmaur = nsd, xir = 0, phiur = TRUE)

```


Arguments

x	quantiles
nmean	normal mean
nsd	normal standard deviation (positive)
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (positive)
xil	lower tail GPD shape parameter
phiul	probability of being below lower threshold [0, 1] or TRUE
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (positive)
xir	upper tail GPD shape parameter
phiur	probability of being above upper threshold [0, 1] or TRUE
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining normal distribution for the bulk between the lower and upper thresholds and GPD for upper and lower tails. The user can pre-specify phiul and phiur permitting a parameterised value for the lower and upper tail fraction respectively. Alternatively, when phiul=TRUE or phiur=TRUE the corresponding tail fraction is estimated as from the normal bulk model.

Notice that the tail fraction cannot be 0 or 1, and the sum of upper and lower tail fractions phiul+phiur<1, so the lower threshold must be less than the upper, ul<ur.

The cumulative distribution function now has three components. The lower tail with tail fraction 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](#), [dgngcon](#), [dgngcon](#), [dgngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [gngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [pgngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [qngngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#), [rgngcon](#); [ditmgng](#), [ditmgng](#), [ditmgng](#), [ditmgng](#), [ditmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [itmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [pitmgng](#), [qitmgng](#), [qitmgng](#), [qitmgng](#).

```

qitmngng, qitmngng, ritmngng, ritmngng, ritmngng, ritmngng, ritmngng; dnormgpdcon, dnormgpdcon,
dnormgpdcon, dnormgpdcon, dnormgpdcon, normgpdcon, normgpdcon, normgpdcon, normgpdcon,
normgpdcon, pnormgpdcon, pnormgpdcon, pnormgpdcon, pnormgpdcon, pnormgpdcon, qnormgpdcon,
qnormgpdcon, qnormgpdcon, qnormgpdcon, qnormgpdcon, rnormgpdcon, rnormgpdcon, rnormgpdcon,
rnormgpdcon, rnormgpdcon; dnormgpd, dnormgpd, dnormgpd, dnormgpd, dnormgpd, normgpd,
normgpd, normgpd, normgpd, normgpd, pnormgpd, pnormgpd, pnormgpd, pnormgpd, pnormgpd,
qnormgpd, qnormgpd, qnormgpd, qnormgpd, qnormgpd, qnormgpd, rnormgpd, rnormgpd, rnormgpd, rnormgpd,
rnormgpd; fgngcon, fgngcon, fgngcon, fgngcon, fgngcon, fgngcon, lgngcon, lgngcon, lgngcon, lgngcon,
lgngcon, nlngcon, nlngcon, nlngcon, nlngcon, nlngcon, nlugngcon, nlugngcon, nlugngcon, nlugngcon,
nlugngcon, nlugngcon, proflugngcon, proflugngcon, proflugngcon, proflugngcon, proflugngcon;
fgng, fgng, fgng, fgng, fgng, lgng, lgng, lgng, lgng, lgng, nlng, nlng, nlng, nlng, nlng, nlng,
nlugng, nlugng, nlugng, nlugng, nlugng, proflugng, proflugng, proflugng, proflugng, proflugng;
fitmngng, fitmngng, fitmngng, fitmngng, fitmngng, litmngng, litmngng, litmngng, litmngng, litmngng,
nleuitmngng, nleuitmngng, nleuitmngng, nleuitmngng, nleuitmngng, nleuitmngng, nleuitmngng, nleuitmngng,
nleuitmngng, nleuitmngng, profluitmngng, profluitmngng, profluitmngng, profluitmngng, profluitmngng,
profluitmngng; fnormgpdcon, fnormgpdcon, fnormgpdcon, fnormgpdcon, fnormgpdcon, lnormgpdcon,
lnormgpdcon, lnormgpdcon, lnormgpdcon, lnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon,
nlnormgpdcon, nlnormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon,
nlunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon,
proflunormgpdcon; fnormgpd, fnormgpd, fnormgpd, fnormgpd, fnormgpd, lnormgpd, lnormgpd,
lnormgpd, lnormgpd, lnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd,
nlunormgpd, nlunormgpd, nlunormgpd, nlunormgpd, nlunormgpd, proflunormgpd, proflunormgpd,
proflunormgpd, proflunormgpd, proflunormgpd

```

Examples

```

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

x = rgng(1000, phiul = 0.15, phiur = 0.15)
xx = seq(-6, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-6, 6))
lines(xx, dgng(xx, phiul = 0.15, phiur = 0.15))

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

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

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

## End(Not run)

```

gngcon

Normal Bulk with GPD Upper and Lower Tails Extreme Value Mixture Model with Single Continuity Constraint at Thresholds

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails with continuity at the lower and upper thresholds. The parameters are the normal mean `nmean` and standard deviation `nsd`, lower tail (threshold `ul`, GPD shape `xil` and tail fraction `phiul`) and upper tail (threshold `ur`, GPD shape `xir` and tail fraction `phiur`).

Usage

```
dgngcon(x, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), xil = 0,
        phiul = TRUE, ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
        log = FALSE)

pgngcon(q, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), xil = 0,
        phiul = TRUE, ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
        lower.tail = TRUE)

qgngcon(p, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd), xil = 0,
        phiul = TRUE, ur = qnorm(0.9, nmean, nsd), xir = 0, phiur = TRUE,
        lower.tail = TRUE)

rgngcon(n = 1, nmean = 0, nsd = 1, ul = qnorm(0.1, nmean, nsd),
        xil = 0, phiul = TRUE, ur = qnorm(0.9, nmean, nsd), xir = 0,
        phiur = TRUE)
```

Arguments

<code>x</code>	quantiles
<code>nmean</code>	normal mean
<code>nsd</code>	normal standard deviation (positive)
<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
<code>ur</code>	upper tail threshold
<code>xir</code>	upper tail GPD shape parameter
<code>phiur</code>	probability of being above upper 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 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 ϕ_{ul} and ϕ_{ur} permitting a parameterised value for the lower and upper tail fraction respectively. Alternatively, when $\phi_{ul}=\text{TRUE}$ or $\phi_{ur}=\text{TRUE}$ 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 $\phi_{ul}+\phi_{ur}<1$, so the lower threshold must be less than the upper, $u_l < u_r$.

The cumulative distribution function now has three components. The lower tail with tail fraction ϕ_{ul} defined by the normal bulk model ($\phi_{ul}=\text{TRUE}$) 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. $\text{pnorm}(u_r, \text{nmean}, \text{nsd})$). The $G_l(X)$ is the conditional GPD cumulative distribution function with negated data and threshold, i.e. $\text{dgp}(-x, -u_l, \text{sigma}_l, x_l, \phi_{ul})$. 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:

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

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>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>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 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

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

Note

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

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default threshold `u=0` and tail fraction `phiu=1` which essentially assumes the user provide excesses above `u` by default, rather than exceedances. The default sample size for `rgpd` is 1.

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

Some key differences arise for `phiu=1` and `phiu<1` (see examples below):

1. For `phiu=1` the `dgpd` evaluates as zero for quantiles below the threshold `u` and `pgpd` evaluates over $[0, 1]$.
2. For `phiu=1` then `pgpd` evaluates as zero below the threshold `u`. For `phiu<1` it evaluates as $1 - \phi_u$ at the threshold and NA below the threshold.
3. For `phiu=1` the quantiles from `qgpd` are above threshold and equal to threshold for `phiu=0`. For `phiu<1` then within upper tail, $p > 1 - \phi_u$, 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](#), [fgpd](#), [fgpd](#), [lgpd](#), [lgpd](#), [lgpd](#), [nlgpd](#), [nlgpd](#), [nlgpd](#)

Examples

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

x = rgpd(1000) # simulate sample from GPD
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgpd(xx))

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

# GPD when xi=0 is exponential, and demonstrating phiu
x = rexp(1000)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dgpd(xx, u = 0, sigmau = 1, xi = 0), lwd = 2)
lines(xx, dgpd(xx, u = 0.5, phiu = 1 - pexp(0.5)), col = "red", lwd = 2)
lines(xx, dgpd(xx, u = 1.5, phiu = 1 - pexp(1.5)), col = "blue", lwd = 2)
legend("topright", paste("u =", c(0, 0.5, 1.5)),
      col=c("black", "red", "blue"), lty = 1, lwd = 2)

# Quantile function and phiu
p = pgpd(xx)
plot(qgpd(p), p, type = "l")
lines(xx, pgpd(xx, u = 2), col = "red")
lines(xx, pgpd(xx, u = 5, phiu = 0.2), col = "blue")
legend("bottomright", c("u = 0 phiu = 1", "u = 2 phiu = 1", "u = 5 phiu = 0.2"),
      col=c("black", "red", "blue"), lty = 1)
```

hillplot

Hill Plot

Description

Plots the Hill plot and some its variants.

Usage

```
hillplot(data, orderlim = NULL, tlim = NULL, hill.type = "Hill", r = 2,
  x.theta = FALSE, y.alpha = FALSE, alpha = 0.05, ylim = NULL,
  legend.loc = "topright", try.thresh = quantile(data[data > 0], 0.9, na.rm
= TRUE), main = paste(ifelse(x.theta, "Alt", ""), hill.type, " Plot", sep =
""), xlab = ifelse(x.theta, "theta", "order"),
  ylab = paste(ifelse(x.theta, "Alt", ""), hill.type, ifelse(y.alpha,
" alpha", " xi"), ">0", sep = ""), ...)
```

Arguments

data	vector of sample data
orderlim	vector of (lower, upper) limits of order statistics to plot estimator, or NULL to use default values
tlim	vector of (lower, upper) limits of range of threshold to plot estimator, or NULL to use default values
hill.type	"Hill" or "SmooHill"
r	smoothing factor for "SmooHill" (integer > 1)
x.theta	logical, should order (FALSE) or theta (TRUE) be given on x-axis
y.alpha	logical, should shape xi (FALSE) or tail index alpha (TRUE) be given on y-axis
alpha	significance level over range (0, 1), or NULL for no CI
ylim	y-axis limits or NULL
legend.loc	location of legend (see legend) or NULL for no legend
try.thresh	vector of thresholds to consider
main	title of plot
xlab	x-axis label
ylab	y-axis label
...	further arguments to be passed to the plotting functions

Details

Produces the Hill, AltHill, SmooHill and AltSmooHill plots, including confidence intervals.

For an ordered iid sequence $X_{(1)} \geq X_{(2)} \geq \dots \geq X_{(n)} > 0$ the Hill (1975) estimator using k order statistics is given by

$$H_{k,n} = \frac{1}{k} \sum_{i=1}^k \log\left(\frac{X_{(i)}}{X_{(k+1)}}\right)$$

which is the pseudo-likelihood estimator of reciprocal of the tail index $\xi = 1/\alpha > 0$ for regularly varying tails (e.g. Pareto distribution). The Hill estimator is defined on orders $k > 2$, as when $k = 1$ the

$$H_{1,n} = 0$$

. The function will calculate the Hill estimator for $k \geq 1$. The simple Hill plot is shown for `hill.type="Hill"`.

Once a sufficiently low order statistic is reached the Hill estimator will be constant, upto sample uncertainty, for regularly varying tails. The Hill plot is a plot of

$$H_{k,n}$$

against the k . Symmetric asymptotic normal confidence intervals assuming Pareto tails are provided.

These so called Hill's horror plots can be difficult to interpret. A smooth form of the Hill estimator was suggested by Resnick and Starica (1997):

$$smooH_{k,n} = \frac{1}{(r-1)k} \sum_{j=k+1}^{rk} H_{j,n}$$

giving the smooHill plot which is shown for `hill.type="SmooHill"`. The smoothing factor is `r=2` by default.

It has also been suggested to plot the order on a log scale, by plotting the points $(\theta, H_{[n^\theta],n})$ for $0 \leq \theta \leq 1$. This gives the so called AltHill and AltSmooHill plots. The alternative x-axis scale is chosen by `x.theta=TRUE`.

The Hill estimator is for the GPD shape $\xi > 0$, or the reciprocal of the tail index $\alpha = 1/\xi > 0$. The shape is plotted by default using `y.alpha=FALSE` and the tail index is plotted when `y.alpha=TRUE`.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The estimated parameter (ξ or α) at each threshold are plot by a horizontal solid line for all higher thresholds. The threshold should be set as low as possible, so a dashed line is shown below the pre-chosen threshold. If the Hill estimator is similar to the dashed line then a lower threshold may be chosen.

If no order statistic (or threshold) limits are provided `orderlim = tlim = NULL` then the lowest order statistic is set to $X_{(3)}$ and highest possible value $X_{(n-1)}$. However, the Hill estimator is always output for all $k = 1, \dots, n-1$ and $k = 1, \dots, \text{floor}(n/k)$ for smooHill estimator.

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

The lower x-axis is the order k or θ , chosen by the option `x.theta=FALSE` and `x.theta=TRUE` respectively. The upper axis is for the corresponding threshold.

Value

`hillplot` gives the Hill plot. It also returns a dataframe containing columns of the order statistics, order, Hill estimator, it's standard deviation and $100(1 - \alpha)\%$ confidence interval (when requested). When the SmooHill plot is selected, then the corresponding SmooHill estimates are appended.

Acknowledgments

Thanks to Younes Mouatasim, Risk Dynamics, Brussels for reporting various bugs in these functions.

Note

Warning: Hill plots are not location invariant.

Asymptotic Wald type CI's are estimated for non-NULL significance level α for the shape parameter, assuming exactly Pareto tails. When plotting on the tail index scale, then a simple reciprocal transform of the CI is applied which may be sub-optimal.

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

Author(s)

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

References

- Hill, B.M. (1975). A simple general approach to inference about the tail of a distribution. *Annals of Statistics* 13, 331-341.
- Resnick, S. and Starica, C. (1997). Smoothing the Hill estimator. *Advances in Applied Probability* 29, 271-293.
- Resnick, S. (1997). Discussion of the Danish Data of Large Fire Insurance Losses. *Astin Bulletin* 27, 139-151.

See Also[hill](#)**Examples**

```
## Not run:
# Reproduce graphs from Figure 2.4 of Resnick (1997)
data(danish, package="evir")
par(mfrow = c(2, 2))

# Hill plot
hillplot(danish, y.alpha=TRUE, ylim=c(1.1, 2))

# Althill plot
hillplot(danish, y.alpha=TRUE, x.theta=TRUE, ylim=c(1.1, 2))

# AltSmooHill plot
hillplot(danish, hill.type="SmooHill", r=3, y.alpha=TRUE, x.theta=TRUE, ylim=c(1.35, 1.85))

# Althill and AltSmooHill plot (no CI's or legend)
hillout = hillplot(danish, hill.type="SmooHill", r=3, y.alpha=TRUE,
  x.theta=TRUE, try.thresh = c(), alpha=NULL, ylim=c(1.1, 2), legend.loc=NULL, lty=2)
n = length(danish)
with(hillout[3:n,], lines(log(ks)/log(n), 1/H, type="s"))

## End(Not run)
```

hpd

*Hybrid Pareto Extreme Value Mixture Model***Description**

Density, cumulative distribution function, quantile function and random number generation for the hybrid Pareto extreme value mixture model. The parameters are the normal mean `nmean` and standard deviation `nsd` and GPD shape `xi`.

Usage

```
dhpd(x, nmean = 0, nsd = 1, xi = 0, log = FALSE)

phpd(q, nmean = 0, nsd = 1, xi = 0, lower.tail = TRUE)

qhpd(p, nmean = 0, nsd = 1, xi = 0, lower.tail = TRUE)

rhpd(n = 1, nmean = 0, nsd = 1, xi = 0)
```

Arguments

<code>x</code>	quantiles
<code>nmean</code>	normal mean
<code>nsd</code>	normal standard deviation (positive)
<code>xi</code>	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 ϕ/u 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 sigmau being replaced by a function of the normal mean, standard deviation and GPD shape parameters. Determined from setting $h(u) = g(u)$ where $h(x)$ and $g(x)$ are the normal and unscaled GPD density functions (i.e. $\text{dnorm}(u, \text{nmean}, \text{nsd})$ and $\text{dgp}(u, u, \text{sigmau}, \text{xi})$). 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 sigmau in terms of the normal mean, standard deviation and GPD shape xi .

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

Value

[dhpd](#) gives the density, [phpd](#) gives the cumulative distribution function, [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: [dhpdcon](#), [dhpdcon](#), [dhpdcon](#), [dhpdcon](#), [dhpdcon](#), [hpdcon](#), [hpdcon](#), [hpdcon](#), [hpdcon](#), [hpdcon](#), [phpdcon](#), [phpdcon](#), [phpdcon](#), [phpdcon](#), [phpdcon](#), [qhpdcn](#), [qhpdcn](#), [qhpdcn](#), [qhpdcn](#), [qhpdcn](#), [rhpdcn](#), [rhpdcn](#), [rhpdcn](#), [rhpdcn](#), [rhpdcn](#)

Examples

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

xx = seq(-5, 20, 0.01)
f1 = dhpd(xx, nmean = 0, nsd = 1, xi = 0.4)
plot(xx, f1, type = "l")
abline(v = 0.4942921)

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

sim = rhpd(10000, nmean = 0, nsd = 1.5, xi = 0.2)
hist(sim, freq = FALSE, 100, xlim = c(-5, 20), ylim = c(0, 0.2))
lines(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = 0.2), col = "blue")

plot(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = 0), type = "l")
lines(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = 0.2), col = "red")
lines(xx, dhpd(xx, nmean = 0, nsd = 1.5, xi = -0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

hpdcon	<i>Hybrid Pareto Extreme Value Mixture Model with Single Continuity Constraint</i>
--------	--

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)

rhpdcn(n = 1, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd), xi = 0)
```

Arguments

<code>x</code>	quantiles
<code>nmean</code>	normal mean
<code>nsd</code>	normal standard deviation (positive)
<code>u</code>	threshold
<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 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. `dnorm(u, nmean, nsd)` and `dgp(u, u, sigmau, xi)`).

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

Value

[dhpdcn](#) gives the density, [phpdcn](#) gives the cumulative distribution function, [qhpdcon](#) gives the quantile function and [rhpdcon](#) gives a random sample.

Note

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

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

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

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

Author(s)

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

References

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

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

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

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

See Also

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

The [condmixt](#) package written by one of the original authors of the hybrid Pareto model (Carreau and Bengio, 2008) also has similar functions for the hybrid Pareto [hpareto](#) and mixture of hybrid Paretos [hparetomixt](#), which are more flexible as they also permit the model to be truncated at zero.

Other hpd hpdcon fhpd fhpdcon normgpd normgpdcon fnormgpd fnormgpdcon: [dhpdp](#), [dhpdp](#), [dhpdp](#), [dhpdp](#), [dhpdp](#), [hpd](#), [hpd](#), [hpd](#), [hpd](#), [hpd](#), [hpd](#), [phdp](#), [phdp](#), [phdp](#), [phdp](#), [phdp](#), [phdp](#), [qhdp](#), [qhdp](#), [qhdp](#), [qhdp](#), [qhdp](#), [rhpdp](#), [rhpdp](#), [rhpdp](#), [rhpdp](#), [rhpdp](#)

Examples

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

xx = seq(-5, 20, 0.01)
f1 = dhpdpcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0.4)
plot(xx, f1, type = "l")
abline(v = 4)

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

sim = rhpdpdpcon(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, dhpdpdpcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0.2), col = "blue")

plot(xx, dhpdpdpcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0), type = "l")
lines(xx, dhpdpdpcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = 0.2), col = "red")
lines(xx, dhpdpdpcon(xx, nmean = 0, nsd = 1.5, u = 1, xi = -0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "u = 1, xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

 internal

Internal Functions

Description

Internal functions not designed to be used directly, but are all exported to make them visible to users.

Usage

```
kdenx(x, kerncentres, lambda, kernel = "gaussian")
```

```
pkdenx(x, kerncentres, lambda, kernel = "gaussian")
```

```
bckdenxsimple(x, kerncentres, lambda, kernel = "gaussian")
pbckdenxsimple(x, kerncentres, lambda, kernel = "gaussian")
bckdenxcutnorm(x, kerncentres, lambda, kernel = "gaussian")
pbckdenxcutnorm(x, kerncentres, lambda, kernel = "gaussian")
bckdenxrenorm(x, kerncentres, lambda, kernel = "gaussian")
pbckdenxrenorm(x, kerncentres, lambda, kernel = "gaussian")
bckdenxreflect(x, kerncentres, lambda, kernel = "gaussian")
pbckdenxreflect(x, kerncentres, lambda, kernel = "gaussian")
pxb(x, lambda)

bckdenxbeta1(x, kerncentres, lambda, xmax)
pbckdenxbeta1(x, kerncentres, lambda, xmax)
bckdenxbeta2(x, kerncentres, lambda, xmax)
pbckdenxbeta2(x, kerncentres, lambda, xmax)
bckdenxgamma1(x, kerncentres, lambda)
pbckdenxgamma1(x, kerncentres, lambda)
bckdenxgamma2(x, kerncentres, lambda)
pbckdenxgamma2(x, kerncentres, lambda)
bckdenxcopula(x, kerncentres, lambda, xmax)
pbckdenxcopula(x, kerncentres, lambda, xmax)
pbckdenxlog(x, kerncentres, lambda, offset, kernel = "gaussian")
pbckdenxnn(x, kerncentres, lambda, kernel = "gaussian", nn)
qmix(x, u, epsilon)
qmixprime(x, u, epsilon)
qgbgmix(x, ul, ur, epsilon)
qgbgmixprime(x, ul, ur, epsilon)
pscounts(x, beta, design.knots, degree)
```

Arguments

x	quantiles
kerncentres	kernel centres (typically sample data vector or scalar)
lambda	bandwidth for kernel (as half-width of kernel) or NULL
kernel	kernel name (default = "gaussian")
xmax	upper bound on support (copula and beta kernels only) or NULL
offset	offset added to kernel centres (logtrans only) or NULL
nn	non-negativity correction method (simple boundary correction only)
u	threshold
epsilon	interval half-width
ul	lower tail threshold
ur	upper tail threshold
beta	vector of B-spline coefficients (required)
design.knots	spline knots for splineDesign function
degree	degree of B-splines (0 is constant, 1 is linear, etc.)

Details

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

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

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Author(s)

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

See Also

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

itmngng

Normal Bulk with GPD Upper and Lower Tails Interval Transition Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution between the upper and lower thresholds with conditional GPD's for the two tails and interval transition. The parameters are the normal mean nmean and standard deviation nsd, interval half-width epsilon, lower tail (threshold ul, GPD scale sigmaul and shape xil and tail fraction phiul) and upper tail (threshold ur, GPD scale sigmaur and shape xiR and tail fraction phiuR).

Usage

```

ditmgng(x, nmean = 0, nsd = 1, epsilon = nsd, ul = qnorm(0.1, nmean,
  nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, log = FALSE)

pitmgng(q, nmean = 0, nsd = 1, epsilon = nsd, ul = qnorm(0.1, nmean,
  nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0, lower.tail = TRUE)

qitmgng(p, nmean = 0, nsd = 1, epsilon, ul = qnorm(0.1, nmean, nsd),
  sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd), sigmaur = nsd,
  xir = 0, lower.tail = TRUE)

ritmgng(n = 1, nmean = 0, nsd = 1, epsilon = sd, ul = qnorm(0.1,
  nmean, nsd), sigmaul = nsd, xil = 0, ur = qnorm(0.9, nmean, nsd),
  sigmaur = nsd, xir = 0)

```

Arguments

x	quantiles
nmean	normal mean
nsd	normal standard deviation (positive)
epsilon	interval half-width
ul	lower tail threshold
sigmaul	lower tail GPD scale parameter (positive)
xil	lower tail GPD shape parameter
ur	upper tail threshold
sigmaur	upper tail GPD scale parameter (positive)
xir	upper tail GPD shape parameter
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

The interval transition extreme value mixture model combines a normal distribution for the bulk between the lower and upper thresholds and GPD for upper and lower tails, with a smooth transition over the interval $(u - \epsilon, u + \epsilon)$ (where u can be exchanged for the lower and upper thresholds). The mixing function warps the normal to map from $(u - \epsilon, u)$ to $(u - \epsilon, u + \epsilon)$ and warps the GPD from $(u, u + \epsilon)$ to $(u - \epsilon, u + \epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(G_l(q(x)) + H_t(r(x)) + G_u(p(x)))$$

where $H_t(x)$ is the truncated normal cdf, i.e. `pnorm(x, nmean, nsd)`. The conditional GPD for the upper tail has cdf $G_u(x)$, i.e. `pgpd(x, ur, sigmaur, xir)` and lower tail cdf $G_l(x)$ is for the

negated support, i.e. $1 - \text{pgpd}(-x, -u_l, \text{sigmaul}, xil)$. The truncated normal is not renormalised to be proper, so $H_t(x)$ contributes $\text{pnorm}(ur, \text{nmean}, \text{nsd}) - \text{pnorm}(u_l, \text{nmean}, \text{nsd})$ to the cdf for all $x \geq (u_r + \epsilon)$ and zero below $x \leq (u_l - \epsilon)$. The normalisation constant κ ensures a proper density, given by $1/(2 + \text{pnorm}(ur, \text{nmean}, \text{nsd}) - \text{pnorm}(u_l, \text{nmean}, \text{nsd}))$ where the 2 is from two GPD components and latter is contribution from normal component.

The mixing functions $q(x)$, $r(x)$ and $p(x)$ are reformulated from the $q_i(x)$ suggested by Holden and Haug (2013). These are symmetric about each threshold, which for convenience will be referred to a simply u . So for computational convenience only a single $q(x; u)$ has been implemented for the lower and upper GPD components called `qmix` for a given u , with the complementary mixing function then defined as $p(x; u) = -q(-x; -u)$. The bulk model mixing function $r(x)$ utilises the equivalent of the $q(x)$ for the lower threshold and $p(x)$ for the upper threshold, so these are reused in the bulk mixing function `qgbgmix`.

A minor adaptation of the mixing function has been applied following a similar approach to that explained in `ditmnormgpd`. For the bulk model mixing function $r(x)$, we need $r(x) \leq u_l$ for all $x \leq u_l - \epsilon$ and $r(x) \geq u_r$ for all $x \geq u_r + \epsilon$, as then the bulk model will contribute zero below the lower interval and the constant $H_t(ur) = H(ur) - H(u_l)$ for all x above the upper interval. Holden and Haug (2013) define $r(x) = x - \epsilon$ for all $x \geq u_r$ and $r(x) = x + \epsilon$ for all $x \leq u_l$. For more straightforward and interpretable computational implementation the mixing function has been set to the lower threshold $r(x) = u_l$ for all $x \leq u_l - \epsilon$ and to the upper threshold $r(x) = u_r$ for all $x \geq u_r + \epsilon$, so the cdf/pdf of the normal model can be used directly. We do not have to define cdf/pdf for the non-proper truncated normal separately. As such $r'(x) = 0$ for all $x \leq u_l - \epsilon$ and $x \geq u_r + \epsilon$ in `qmixxprime`, which also makes it clearer that normal does not contribute to either tails beyond the intervals and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the normal and GPD components directly.

Value

`ditmngng` gives the density, `pitmngng` gives the cumulative distribution function, `qitmngng` gives the quantile function and `ritmngng` 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 `ritmngng` 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 `ritmngng` is 1.

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

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

Author(s)

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


```

lines(xx, kappa * dgpd(-xx, -ul, sigmau = 1, xi = 0.5), col = "blue", lty = 2, lwd = 2)
lines(xx, kappa * dnorm(xx, 0, 1), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dgpd(xx, ur, sigmau = 1, xi = 0.5), col = "green", lty = 2, lwd = 2)
abline(v = ul + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "blue")
abline(v = ur + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "green")
legend('topright', c('Normal-GPD ITM', 'kappa*GPD Lower', 'kappa*Normal', 'kappa*GPD Upper'),
      col = c("black", "blue", "red", "green"), lty = c(1, 2, 2, 2), lwd = 2)

# cdf contributions
F = pitmgng(xx, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5, ur, sigmaur = 1, xir = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(-5, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx < ul], kappa * (1 - pgpd(-xx[xx < ul], -ul, 1, 0.5)), col = "blue", lty = 2, lwd = 2)
lines(xx[(xx >= ul) & (xx <= ur)], kappa * (1 + pnorm(xx[(xx >= ul) & (xx <= ur)], 0, 1) -
      pnorm(ul, 0, 1)), col = "red", lty = 2, lwd = 2)
lines(xx[xx > ur], kappa * (1 + (pnorm(ur, 0, 1) - pnorm(ul, 0, 1)) +
      pgpd(xx[xx > ur], ur, sigmau = 1, xi = 0.5)), col = "green", lty = 2, lwd = 2)
abline(v = ul + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "blue")
abline(v = ur + epsilon * seq(-1, 1), lty = c(2, 1, 2), col = "green")
legend('topleft', c('Normal-GPD ITM', 'kappa*GPD Lower', 'kappa*Normal', 'kappa*GPD Upper'),
      col = c("black", "blue", "red", "green"), lty = c(1, 2, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmgng(10000, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5,
      ur, sigmaur = 1, xir = 0.5)
hist(x, freq = FALSE, breaks = seq(-1000, 1000, 0.1), xlim = c(-5, 5))
lines(xx, ditmgng(xx, nmean = 0, nsd = 1, epsilon, ul, sigmaul = 1, xil = 0.5,
      ur, sigmaur = 1, xir = 0.5), lwd = 2, col = 'black')

## End(Not run)

```

itmnormgpd

Normal Bulk and GPD Tail Interval Transition Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the normal bulk and GPD tail interval transition mixture model. The parameters are the normal mean `nmean` and standard deviation `nsd`, threshold `u`, interval half-width `epsilon`, GPD scale `sigmau` and shape `xi`.

Usage

```

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

pitmnormgpd(q, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9, nmean,
      nsd), sigmau = nsd, xi = 0, lower.tail = TRUE)

qitmnormgpd(p, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9, nmean,
      nsd), sigmau = nsd, xi = 0, lower.tail = TRUE)

ritmnormgpd(n = 1, nmean = 0, nsd = 1, epsilon = nsd, u = qnorm(0.9,
      nmean, nsd), sigmau = nsd, xi = 0)

```


Arguments

<code>x</code>	quantiles
<code>nmean</code>	normal mean
<code>nsd</code>	normal standard deviation (positive)
<code>epsilon</code>	interval half-width
<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

The interval transition mixture model combines a normal for the bulk model with GPD for the tail model, with a smooth transition over the interval $(u - \epsilon, u + \epsilon)$. The mixing function warps the normal to map from $(u - \epsilon, u)$ to $(u - \epsilon, u + \epsilon)$ and warps the GPD from $(u, u + \epsilon)$ to $(u - \epsilon, u + \epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(H_t(q(x)) + G(p(x)))$$

where $H_t(x)$ and $G(x)$ are the truncated normal and conditional GPD cumulative distribution functions (i.e. `pnorm(x, nmean, nsd)` and `pgpd(x, u, sigmau, xi)`) respectively. The truncated normal is not renormalised to be proper, so $H_t(x)$ contributes `pnorm(u, nmean, nsd)` to the cdf for all $x \geq (u + \epsilon)$. The normalisation constant κ ensures a proper density, given by $1/(1 + \text{pnorm}(u, \text{nmean}, \text{nsd}))$ where 1 is from GPD component and latter is contribution from normal component.

The mixing functions $q(x)$ and $p(x)$ suggested by Holden and Haug (2013) have been implemented. These are symmetric about the threshold u . So for computational convenience only $q(x; u)$ has been implemented as `qmix` for a given u , with the complementary mixing function is then defined as $p(x; u) = -q(-x; -u)$.

A minor adaptation of the mixing function has been applied. For the mixture model to function correctly $q(x) \geq u$ for all $x \geq u + \epsilon$, as then the bulk model will contribute the constant $H_t(u) = H(u)$ for all x above the interval. Holden and Haug (2013) define $q(x) = x - \epsilon$ for all $x \geq u$. For more straightforward and interpretable computational implementation the mixing function has been set to the threshold $q(x) = u$ for all $x \geq u$, so the cdf/pdf of the normal model can be used directly. We do not have to define cdf/pdf for the non-proper truncated normal separately. As such $q'(x) = 0$ for all $x \geq u$ in `qmixprime`, which also makes it clearer that normal does not contribute to the tail above the interval and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the normal and GPD components directly.

Value

`ditmnormgpd` gives the density, `pitmnormgpd` gives the cumulative distribution function, `qitmnormgpd` gives the quantile function and `ritmnormgpd` gives a random sample.

Note

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

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

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

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

Author(s)

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

References

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

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

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

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

See Also

`normgpd`, `gpd` and `dnorm`

Examples

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

xx = seq(-4, 5, 0.01)
u = 1.5
epsilon = 0.4
kappa = 1/(1 + pnorm(u, 0, 1))

f = ditmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(-4, 5), type = 'l', lwd = 2, xlab = "x", ylab = "density")
lines(xx, kappa * dgpd(xx, u, sigmau = 1, xi = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dnorm(xx, 0, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Normal-GPD ITM', 'kappa*Normal', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# cdf contributions
F = pitmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(-4, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx > u], kappa * (pnorm(u, 0, 1) + pgpd(xx[xx > u], u, sigmau = 1, xi = 0.5)),
      col = "red", lty = 2, lwd = 2)
```

```

lines(xx[xx <= u], kappa * pnorm(xx[xx <= u], 0, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topleft', c('Normal-GPD ITM', 'kappa*Normal', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmnormgpd(10000, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5)
hist(x, freq = FALSE, breaks = seq(-4, 1000, 0.1), xlim = c(-4, 5))
lines(xx, ditmnormgpd(xx, nmean = 0, nsd = 1, epsilon, u, sigmau = 1, xi = 0.5),
      lwd = 2, col = 'black')

## End(Not run)

```

itmweibullgpd

Weibull Bulk and GPD Tail Interval Transition Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the Weibull bulk and GPD tail interval transition mixture model. The parameters are the Weibull shape and scale w_{scale} , threshold u , interval half-width ϵ , GPD scale σ_{mau} and shape ξ .

Usage

```

ditmweibullgpd(x, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2), u = qweibull(0.9,
  wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale *
  gamma(1 + 1/wshape))^2), xi = 0, log = FALSE)

pitmweibullgpd(q, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2), u = qweibull(0.9,
  wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale *
  gamma(1 + 1/wshape))^2), xi = 0, lower.tail = TRUE)

qitmweibullgpd(p, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2), u = qweibull(0.9,
  wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale *
  gamma(1 + 1/wshape))^2), xi = 0, lower.tail = TRUE)

ritmweibullgpd(n = 1, wshape = 1, wscale = 1, epsilon = sqrt(wscale^2 *
  gamma(1 + 2/wshape) - (wscale * gamma(1 + 1/wshape))^2), u = qweibull(0.9,
  wshape, wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale *
  gamma(1 + 1/wshape))^2), xi = 0)

```

Arguments

<code>x</code>	quantiles
<code>wshape</code>	Weibull shape (positive)
<code>wscale</code>	Weibull scale (positive)
<code>epsilon</code>	interval half-width

<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

The interval transition mixture model combines a Weibull for the bulk model with GPD for the tail model, with a smooth transition over the interval $(u - \epsilon, u + \epsilon)$. The mixing function warps the Weibull to map from $(u - \epsilon, u)$ to $(u - \epsilon, u + \epsilon)$ and warps the GPD from $(u, u + \epsilon)$ to $(u - \epsilon, u + \epsilon)$.

The cumulative distribution function is defined by

$$F(x) = \kappa(H_t(q(x)) + G(p(x)))$$

where $H_t(x)$ and $G(X)$ are the truncated Weibull and conditional GPD cumulative distribution functions (i.e. `pweibull(x, wshape, wscale)` and `pgpd(x, u, sigmau, xi)`) respectively. The truncated Weibull is not renormalised to be proper, so $H_t(x)$ contributes `pweibull(u, wshape, wscale)` to the cdf for all $x \geq (u + \epsilon)$. The normalisation constant κ ensures a proper density, given by $1/(1 + \text{pweibull}(u, wshape, wscale))$ where 1 is from GPD component and latter is contribution from Weibull component.

The mixing functions $q(x)$ and $p(x)$ suggested by Holden and Haug (2013) have been implemented. These are symmetric about the threshold u . So for computational convenience only $q(x; u)$ has been implemented as `qmix` for a given u , with the complementary mixing function is then defined as $p(x; u) = -q(-x; -u)$.

A minor adaptation of the mixing function has been applied. For the mixture model to function correctly $q(x) \geq u$ for all $x \geq u + \epsilon$, as then the bulk model will contribute the constant $H_t(u) = H(u)$ for all x above the interval. Holden and Haug (2013) define $q(x) = x - \epsilon$ for all $x \geq u$. For more straightforward and interpretable computational implementation the mixing function has been set to the threshold $q(x) = u$ for all $x \geq u$, so the cdf/pdf of the Weibull model can be used directly. We do not have to define cdf/pdf for the non-proper truncated Weibull separately. As such $q'(x) = 0$ for all $x \geq u$ in `qmixprime`, which also it makes clearer that Weibull does not contribute to the tail above the interval and vice-versa.

The quantile function within the transition interval is not available in closed form, so has to be solved numerically. Outside of the interval, the quantile are obtained from the Weibull and GPD components directly.

Value

`ditmweibullgpd` gives the density, `pitmweibullgpd` gives the cumulative distribution function, `qitmweibullgpd` gives the quantile function and `ritmweibullgpd` gives a random sample.

Note

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

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

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

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

Author(s)

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

References

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

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

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

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

See Also

[weibullgpd](#), [gpd](#) and [dweibull](#)

Examples

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

xx = seq(0.001, 5, 0.01)
u = 1.5
epsilon = 0.4
kappa = 1/(1 + pweibull(u, 2, 1))

f = ditmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, f, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2, xlab = "x", ylab = "density")
lines(xx, kappa * dgpd(xx, u, sigmau = 1, xi = 0.5), col = "red", lty = 2, lwd = 2)
lines(xx, kappa * dweibull(xx, 2, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Weibull-GPD ITM', 'kappa*Weibull', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# cdf contributions
F = pitmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
plot(xx, F, ylim = c(0, 1), xlim = c(0, 5), type = 'l', lwd = 2, xlab = "x", ylab = "cdf")
lines(xx[xx > u], kappa * (pweibull(u, 2, 1) + pgpd(xx[xx > u], u, sigmau = 1, xi = 0.5)),
      col = "red", lty = 2, lwd = 2)
```

```

lines(xx[xx <= u], kappa * pweibull(xx[xx <= u], 2, 1), col = "blue", lty = 2, lwd = 2)
abline(v = u + epsilon * seq(-1, 1), lty = c(2, 1, 2))
legend('topright', c('Weibull-GPD ITM', 'kappa*Weibull', 'kappa*GPD'),
      col = c("black", "blue", "red"), lty = c(1, 2, 2), lwd = 2)

# simulated data density histogram and overlay true density
x = ritmweibullgpd(10000, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5)
hist(x, freq = FALSE, breaks = seq(0, 1000, 0.1), xlim = c(0, 5))
lines(xx, ditmweibullgpd(xx, wshape = 2, wscale = 1, epsilon, u, sigmau = 1, xi = 0.5),
      lwd = 2, col = 'black')

## End(Not run)

```

kden

Kernel Density Estimation, With Variety of Kernels

Description

Density, cumulative distribution function, quantile function and random number generation for the kernel density estimation using the kernel specified by `kernel`, with a constant bandwidth specified by either `lambda` or `bw`.

Usage

```

dkden(x, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
      log = FALSE)

pkden(q, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
      lower.tail = TRUE)

qkden(p, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian",
      lower.tail = TRUE)

rkden(n = 1, kerncentres, lambda = NULL, bw = NULL, kernel = "gaussian")

```

Arguments

<code>x</code>	quantiles
<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>bw</code>	bandwidth for kernel (as standard deviations of kernel) or NULL
<code>kernel</code>	kernel name (default = "gaussian")
<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

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 also rather inefficient, as it can be carried out more efficiently using a mixture representation.

The quantile function is rather complicated as there is no closed form solution, so is obtained by numerical approximation of the inverse cumulative distribution function $P(X \leq q) = p$ to find q . The quantile function [qkden](#) evaluates the KDE cumulative distribution function over the range from `c(max(kerncentre) - lambda, max(kerncentre) + lambda)`, or `c(max(kerncentre) - 5*lambda, max(kerncentre) + 5*lambda)` for normal kernel. Outside of this range the quantiles are set to `-Inf` for lower tail and `Inf` for upper tail. A sequence of values of length fifty times the number of kernels (with minimum of 1000) is first calculated. Spline based interpolation using [splinefun](#), with default `monoH.FC` method, is then used to approximate the quantile function. This is a similar approach to that taken by Matt Wand in the [qkde](#) in the [ks](#) package.

If no bandwidth is provided `lambda=NULL` and `bw=NULL` then the normal reference rule is used, using the [bw.nrd0](#) function, which is consistent with the [density](#) function. At least two kernel centres must be provided as the variance needs to be estimated.

Value

[dkden](#) gives the density, [pkden](#) gives the cumulative distribution function, [qkden](#) gives the quantile function and [rkden](#) gives a random sample.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

lbckden, lbckden, lbckden, nlbckden, nlbckden, nlbckden; fkden, fkden, fkden, lkden, lkden, lkden, nlkden, nlkden, nlkden

Examples

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

nk=50
x = rnorm(nk)
xx = seq(-5, 5, 0.01)
plot(xx, dnorm(xx))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = bw.nrd0(x))*0.05)
lines(xx, dkden(xx, x), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
legend("topright", c("True Density", "KDE Using evmix", "KDE Using density function"),
lty = c(1, 1, 2), lwd = c(1, 2, 2), col = c("black", "red", "green"))

# Estimate bandwidth using cross-validation likelihood
x = rnorm(nk)
fit = fkden(x)
hist(x, nk/5, freq = FALSE, xlim = c(-5, 5), ylim = c(0, 0.6))
rug(x)
for (i in 1:nk) lines(xx, dnorm(xx, x[i], sd = fit$bw)*0.05)
lines(xx, dnorm(xx), col = "black")
lines(xx, dkden(xx, x, lambda = fit$lambda), lwd = 2, col = "red")
lines(density(x), lty = 2, lwd = 2, col = "green")
lines(density(x, bw = fit$bw), lwd = 2, lty = 2, col = "blue")
legend("topright", c("True Density", "KDE fitted evmix",
"KDE Using density, default bandwidth", "KDE Using density, c-v likelihood bandwidth"),
lty = c(1, 1, 2, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

plot(xx, pnorm(xx), type = "l")
rug(x)
lines(xx, pkden(xx, x), lwd = 2, col = "red")
lines(xx, pkden(xx, x, lambda = fit$lambda), lwd = 2, col = "green")
# green and blue (quantile) function should be same
p = seq(0, 1, 0.001)
lines(qkden(p, x, lambda = fit$lambda), p, lwd = 2, lty = 2, col = "blue")
legend("topleft", c("True Density", "KDE using evmix, normal reference rule",
"KDE using evmix, c-v likelihood", "KDE quantile function, c-v likelihood"),
lty = c(1, 1, 1, 2), lwd = c(1, 2, 2, 2), col = c("black", "red", "green", "blue"))

xnew = rkden(10000, x, lambda = fit$lambda)
hist(xnew, breaks = 100, freq = FALSE, xlim = c(-5, 5))
rug(xnew)
lines(xx, dnorm(xx), col = "black")
lines(xx, dkden(xx, x), lwd = 2, col = "red")
legend("topright", c("True Density", "KDE Using evmix"),
lty = c(1, 2), lwd = c(1, 2), col = c("black", "red"))

## End(Not run)
```

kdengpd

*Kernel Density Estimate and GPD Tail Extreme Value Mixture Model***Description**

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with kernel density estimate for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the bandwidth λ , threshold u GPD scale σ and shape ξ and tail fraction ϕ .

Usage

```
dkdengpd(x, kerncentres, lambda = NULL, u = as.vector(quantile(kerncentres,
  0.9)), sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", log = FALSE)
```

```
pkdengpd(q, kerncentres, lambda = NULL, u = as.vector(quantile(kerncentres,
  0.9)), sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)
```

```
qkdengpd(p, kerncentres, lambda = NULL, u = as.vector(quantile(kerncentres,
  0.9)), sigmau = sqrt(6 * var(kerncentres))/pi, xi = 0, phiu = TRUE,
  bw = NULL, kernel = "gaussian", lower.tail = TRUE)
```

```
rkdengpd(n = 1, kerncentres, lambda = NULL,
  u = as.vector(quantile(kerncentres, 0.9)), sigmau = sqrt(6 *
  var(kerncentres))/pi, xi = 0, phiu = TRUE, bw = NULL,
  kernel = "gaussian")
```

Arguments

x	quantiles
kerncentres	kernel centres (typically sample data vector or scalar)
lambda	bandwidth for kernel (as half-width of kernel) or NULL
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter
phiu	probability of being above threshold $[0, 1]$ or TRUE
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
kernel	kernel name (default = "gaussian")
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining kernel density estimate (KDE) for the bulk below the threshold and GPD for upper tail.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction 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.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

Examples

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

kerncentres=rnorm(500, 0, 1)
xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpd(xx, kerncentres, u = 1.2, 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_u .

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)
```

```
rkdeingpdcon(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.

Acknowledgments

Based on code by Anna MacDonald produced for MATLAB.

Note

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

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

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

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

Due to symmetry, the lower tail can be described by GPD by negating the quantiles.

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

Author(s)

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

References

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

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

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

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

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

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

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

See Also

[kernels](#), [kfun](#), [density](#), [bw.nrd0](#) and [dkde](#) in [ks](#) package.

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

Examples

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

kerncentres=rnorm(500, 0, 1)
xx = seq(-4, 4, 0.01)
hist(kerncentres, breaks = 100, freq = FALSE)
lines(xx, dkdengpdcon(xx, kerncentres, u = 1.2, xi = 0.1))

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

x = rkdengpdcon(1000, kerncentres, phiu = 0.2, u = 1, xi = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dkdengpdcon(xx, kerncentres, phiu = 0.2, u = 1, xi = -0.1))

plot(xx, dkdengpdcon(xx, kerncentres, xi=0, u = 1, phiu = 0.2), type = "l")
```



```

lines(xx, dkdengpdcon(xx, kerncentres, xi=0.2, u = 1, phiu = 0.2), col = "red")
lines(xx, dkdengpdcon(xx, kerncentres, xi=-0.2, u = 1, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)

```

kernels

*Kernel functions***Description**

Functions for commonly used kernels for kernel density estimation. The density and cumulative distribution functions are provided.

Usage

```

kdgaussian(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kduniform(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdtriangular(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdepanechnikov(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdbiweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdtriweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdtricube(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdparzen(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kdoptcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kpgaussian(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kpuniform(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kptriangular(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kpepanechnikov(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kpbiweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kptriweight(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)
kptricube(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

```

```

kpparzen(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kpoptcosine(x = 0, lambda = NULL, bw = NULL, kerncentres = 0)

kdz(z, kernel = "gaussian")

kpz(z, kernel = "gaussian")

```

Arguments

x	location to evaluate KDE (single scalar or vector)
lambda	bandwidth for kernel (as half-width of kernel) or NULL
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
kerncentres	kernel centres (typically sample data vector or scalar)
z	standardised location put into kernel $z = (x - \text{kerncentres}) / \text{lambda}$
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 $\text{lambda}=1$. 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 $\text{bw}=\text{lambda}$ 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 [apply](#) 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 $\text{kerncentre}=0$. 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 $z = (x - \text{kerncentres})/\text{lambda}$.

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

Examples

```
xx = seq(-2, 2, 0.01)
plot(xx, kdgussian(xx), type = "l", col = "black", ylim = c(0, 1.2))
lines(xx, kduniform(xx), col = "grey")
lines(xx, kdtriangular(xx), col = "blue")
lines(xx, kdepnechnikov(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	<i>Various subsidiary kernel function, conversion of bandwidths and evaluating certain kernel integrals.</i>
------	--

Description

Functions for checking the inputs to the kernel functions, evaluating integrals $\int u^l K * (u) du$ for $l = 0, 1, 2$ and conversion between the two bandwidth definitions.

Usage

```
check.kinputs(x, lambda, bw, kerncentres, allownull = FALSE)

check.kernel(kernel)

check.kbw(lambda, bw, allownull = FALSE)

klambda(bw = NULL, kernel = "gaussian", lambda = NULL)

kbw(lambda = NULL, kernel = "gaussian", bw = NULL)

ka0(truncpoint, kernel = "gaussian")

ka1(truncpoint, kernel = "gaussian")

ka2(truncpoint, kernel = "gaussian")
```

Arguments

x	location to evaluate KDE (single scalar or vector)
lambda	bandwidth for kernel (as half-width of kernel) or NULL
bw	bandwidth for kernel (as standard deviations of kernel) or NULL
kerncentres	kernel centres (typically sample data vector or scalar)
allownull	logical, where TRUE permits NULL values
kernel	kernel name (default = "gaussian")
truncpoint	upper endpoint as standardised location x/λ

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):

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

<code>sigmau</code>	scale parameter (positive)
<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.

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

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

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

and above the threshold $x > u$:

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

where $H(x)$ and $G(X)$ are the log-normal and conditional GPD cumulative distribution functions (i.e. `plnorm(x, 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.

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`, `dlognormgpdcon`, `dlognormgpdcon`, `dlognormgpdcon`, `dlognormgpdcon`, `lognormgpdcon`, `lognormgpdcon`, `lognormgpdcon`, `lognormgpdcon`, `lognormgpdcon`, `plognormgpdcon`, `plognormgpdcon`, `plognormgpdcon`, `plognormgpdcon`, `plognormgpdcon`, `qlognormgpdcon`, `qlognormgpdcon`, `qlognormgpdcon`, `qlognormgpdcon`, `qlognormgpdcon`, `rlognormgpdcon`, `rlognormgpdcon`, `rlognormgpdcon`, `rlognormgpdcon`, `rlognormgpdcon`

Examples

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

x = rlognormgpd(1000)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpd(xx))

# three tail behaviours
plot(xx, plognormgpd(xx), type = "l")
lines(xx, plognormgpd(xx, xi = 0.3), col = "red")
lines(xx, plognormgpd(xx, xi = -0.3), col = "blue")
```

```

legend("bottomright", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

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

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

## End(Not run)

```

lognormgpdcon	<i>Log-Normal Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint</i>
---------------	---

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 .

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)

qlognormgpdcon(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

lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining log-normal distribution for the bulk below the threshold and GPD for upper tail 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 `dgp(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](#), [dlognormgpd](#), [dlognormgpd](#), [dlognormgpd](#), [dlognormgpd](#), [lognormgpd](#), [lognormgpd](#), [lognormgpd](#), [lognormgpd](#), [plognormgpd](#), [plognormgpd](#), [plognormgpd](#), [plognormgpd](#), [plognormgpd](#), [qlognormgpd](#), [qlognormgpd](#), [qlognormgpd](#), [qlognormgpd](#), [qlognormgpd](#), [qlognormgpd](#), [rlognormgpd](#), [rlognormgpd](#), [rlognormgpd](#), [rlognormgpd](#), [rlognormgpd](#)

Examples

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

x = rlognormgpdcon(1000)
xx = seq(-1, 10, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 10))
lines(xx, dlognormgpdcon(xx))

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

```

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

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

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

## End(Not run)

```

mgamma

*Mixture of Gammas Distribution***Description**

Density, cumulative distribution function, quantile function and random number generation for the mixture of gammas distribution. The parameters are the multiple gamma shapes mgshape scales mgscale and weights mgweights.

Usage

```

dmgamma(x, mgshape = 1, mgscale = 1, mgweight = NULL, log = FALSE)

pmgamma(q, mgshape = 1, mgscale = 1, mgweight = NULL, lower.tail = TRUE)

qmgamma(p, mgshape = 1, mgscale = 1, mgweight = NULL, lower.tail = TRUE)

rmgamma(n = 1, mgshape = 1, mgscale = 1, mgweight = NULL)

```

Arguments

x	quantiles
mgshape	mgamma shape (positive) as list or vector
mgscale	mgamma scale (positive) as list or vector
mgweight	mgamma weights (positive) as list or vector (NULL for equi-weighted)
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Distribution functions for weighted mixture of gammas.

Suppose there are $M \geq 1$ 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.

Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

Note

All inputs are vectorised except `log` and `lower.tail`, and the gamma mixture parameters can be vectorised within the list. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of `rmgamma` any input vector must be of length `n`. The only exception is when the parameters are single scalar values, input as vector of length M .

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

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

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

Author(s)

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

<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.

gammagpd, gpd and dgamma

[illegible]

fmgamma, fmgamma, lmgamma, lmgamma, lmgamma, lmgamma, nLEMmgamma, nLEMmgamma, nLEMmgamma, nLEMmgamma, nlmgamma, nlmgamma, nlmgamma, nlmgamma

Examples

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

n = 1000
x = rmgamma(n, mgshape = c(1, 6), mgscale = c(1,2), mgweight = c(1, 2))
xx = seq(-1, 40, 0.01)

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgamma(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2)))

# By direct simulation
n1 = rbinom(1, n, 1/3) # sample size from population 1
x = c(rgamma(n1, shape = 1, scale = 1), rgamma(n - n1, shape = 6, scale = 2))

hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, dmgamma(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2)))

## End(Not run)
```

mgammagpd

Mixture of Gammas Bulk and GPD Tail Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with mixture of gammas for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the multiple gamma shapes mgshape, scales mgscale and mgweights, threshold u GPD scale sigmau and shape xi and tail fraction phiu.

Usage

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

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

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

rmgammagpd(n = 1, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgscale[[1]]), sigmau = sqrt(mgshape[[1]])
  * mgscale[[1]], xi = 0, phiu = TRUE)
```


Arguments

x	quantiles
mgshape	mgamma shape (positive) as list or vector
mgscale	mgamma scale (positive) as list or vector
mgweight	mgamma weights (positive) as list or vector (NULL for equi-weighted)
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter
phiu	probability of being above threshold $[0, 1]$ or TRUE
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining mixture of gammas for the bulk below the threshold and GPD for upper tail.

The user can pre-specify phiu permitting a parameterised value for the tail fraction 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.

Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

Note

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

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

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

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

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
- http://en.wikipedia.org/wiki/Gamma_distribution
- http://en.wikipedia.org/wiki/Generalized_Pareto_distribution
- http://en.wikipedia.org/wiki/Mixture_model
- McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

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.

gpd and dgamma

[illegible]

Examples

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

x = rmgammagpd(1000, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2),
  u = 15, sigmau = 4, xi = 0)
xx = seq(-1, 40, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 40))
lines(xx, 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/mgshape[[1]]), xi = 0, phiu = TRUE,
  log = FALSE)

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

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

rmammagpdcon(n = 1, mgshape = 1, mgscale = 1, mgweight = NULL,
  u = qgamma(0.9, mgshape[[1]], 1/mgshape[[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.

Acknowledgments

Thanks to Daniela Laas, University of St Gallen, Switzerland for reporting various bugs in these functions.

Note

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

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

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

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

Author(s)

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

References

- <http://www.math.canterbury.ac.nz/~c.scarrott/evmix>
- http://en.wikipedia.org/wiki/Gamma_distribution
- http://en.wikipedia.org/wiki/Generalized_Pareto_distribution
- http://en.wikipedia.org/wiki/Mixture_model
- McLachlan, G.J. and Peel, D. (2000). Finite Mixture Models. Wiley.

do Nascimento, F.F., Ganerman, D. and Lopes, H.F. (2011). A semiparametric Bayesian approach to extreme value estimation. *Statistical Computing*, 22(2), 661-675.

See Also

[illegible]

Examples

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

x = rmgammaepdcon(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, dmgammaepdcon(xx, mgshape = c(1, 6), mgscale = c(1, 2), mgweight = c(1, 2),
  u = 15, xi = 0))
abline(v = 15)

## End(Not run)
```

mrlplot

Mean Residual Life Plot

Description

Plots the sample mean residual life (MRL) plot.

Usage

```
mrlplot(data, tlim = NULL, nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomleft",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Mean Residual Life Plot", xlab = "Threshold u",
  ylab = "Mean Excess", ...)
```

Arguments

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

Details

Plots the sample mean residual life plot, which is also known as the mean excess plot.

If the generalised Pareto distribution (GPD) is an appropriate model for the excesses $X - u$ above u then their expected value is:

$$E(X - u | X > u) = \sigma_u / (1 - \xi).$$

For any higher threshold $v > u$ the expected value is

$$E(X - v | X > v) = [\sigma_u + \xi * (v - u)] / (1 - \xi)$$

which is linear in higher thresholds v with intercept given by $[\sigma_u - \xi * u] / (1 - \xi)$ and gradient $\xi / (1 - \xi)$. The estimated mean residual life above a threshold v is given by the sample mean excess $\text{mean}(x[x > v]) - v$.

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

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

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

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

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

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

Value

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

Acknowledgments

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

Note

If the user specifies the threshold range, the thresholds above the second largest are dropped. A warning message is given if any thresholds have at most 5 exceedances, in which case the confidence

interval is not calculated as it is unreliable due to small sample. If there are less than 10 exceedances of the minimum threshold then the function will stop.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. *REVSTAT - Statistical Journal* 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>

Coles S.G. (2004). *An Introduction to the Statistical Modelling of Extreme Values*. Springer-Verlag: London.

See Also

[gpd](#) and [mrlplot](#) from [evd](#) library

Examples

```
x = rnorm(1000)
mrlplot(x)
mrlplot(x, tlim = c(0, 2.2))
mrlplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))
mrlplot(x, tlim = c(0, 3), try.thresh = c(0.5, 1, 1.5))
```

normgpd

Normal Bulk and GPD Tail Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the normal mean *nmean* and standard deviation *nsd*, threshold *u* GPD scale *sigmau* and shape *xi* and tail fraction *phiu*.

Usage

```
dnormgpd(x, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd),
  sigmau = nsd, xi = 0, phiu = TRUE, log = FALSE)

pnormgpd(q, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd),
  sigmau = nsd, xi = 0, phiu = TRUE, lower.tail = TRUE)

qnormgpd(p, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd),
  sigmau = nsd, xi = 0, phiu = TRUE, lower.tail = TRUE)

rnormgpd(n = 1, nmean = 0, nsd = 1, u = qnorm(0.9, nmean, nsd),
  sigmau = nsd, xi = 0, phiu = TRUE)
```

Arguments

x	quantiles
nmean	normal mean
nsd	normal standard deviation (positive)
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter
phiu	probability of being above threshold $[0, 1]$ or TRUE
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining normal distribution for the bulk below the threshold and GPD for upper tail.

The user can pre-specify phiu permitting a parameterised value for the tail fraction 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.


```
fnormgpdcon, fnormgpdcon, fnormgpdcon, fnormgpdcon, fnormgpdcon, lnormgpdcon, lnormgpdcon,
lnormgpdcon, lnormgpdcon, lnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon, nlnormgpdcon,
nlnormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon, nlunormgpdcon,
proflunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon, proflunormgpdcon;
fnormgpd, fnormgpd, fnormgpd, fnormgpd, fnormgpd, lnormgpd, lnormgpd, lnormgpd, lnormgpd,
lnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlnormgpd, nlunormgpd, nlunormgpd,
nlunormgpd, nlunormgpd, nlunormgpd, proflunormgpd, proflunormgpd, proflunormgpd, proflunormgpd,
proflunormgpd
```

Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

x = rnormgpd(1000)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx))

# three tail behaviours
plot(xx, pnormgpd(xx), type = "l")
lines(xx, pnormgpd(xx, xi = 0.3), col = "red")
lines(xx, pnormgpd(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

x = rnormgpd(1000, phiu = 0.2)
xx = seq(-4, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-4, 6))
lines(xx, dnormgpd(xx, phiu = 0.2))

plot(xx, dnormgpd(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dnormgpd(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dnormgpd(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

normgpdcon

Normal Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with normal for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the normal mean μ and standard deviation σ , threshold u and GPD shape ξ and tail fraction ϕ .

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 `dgpd(x, u, sigmau, xi)`) respectively. The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u) / [1 - \phi_u] h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)] / h(u)$$

. See [gpd](#) for details of GPD upper tail component and [dnorm](#) for details of normal bulk component.

Value

[dnormgpdcon](#) gives the density, [pnormgpdcon](#) gives the cumulative distribution function, [qnormgpdcon](#) gives the quantile function and [rnormgpdcon](#) gives a random sample.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of [rnormgpdcon](#) any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for [rnormgpdcon](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in `x`, `p` and `q` are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles. The normal mean `nmean` and GPD threshold `u` will also require negation.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Normal_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.


```

lines(xx, dnormgpdcon(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)

```

pickandsplot

Pickands Plot

Description

Produces the Pickand's plot.

Usage

```

pickandsplot(data, orderlim = NULL, tlim = NULL, y.alpha = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "topright",
  try.thresh = quantile(data, 0.9, na.rm = TRUE), main = "Pickand's Plot",
  xlab = "order", ylab = ifelse(y.alpha, "tail index - alpha",
    "shape - xi"), ...)

```

Arguments

data	vector of sample data
orderlim	vector of (lower, upper) limits of order statistics to plot estimator, or NULL to use default values
tlim	vector of (lower, upper) limits of range of threshold to plot estimator, or NULL to use default values
y.alpha	logical, should shape xi (FALSE) or tail index alpha (TRUE) be given on y-axis
alpha	significance level over range (0, 1), or NULL for no CI
ylim	y-axis limits or NULL
legend.loc	location of legend (see legend) or NULL for no legend
try.thresh	vector of thresholds to consider
main	title of plot
xlab	x-axis label
ylab	y-axis label
...	further arguments to be passed to the plotting functions

Details

Produces the Pickand's plot including confidence intervals.

For an ordered iid sequence $X_{(1)} \geq X_{(2)} \geq \dots \geq X_{(n)}$ the Pickand's estimator of the reciprocal of the shape parameter ξ at the k th order statistic is given by

$$\hat{\xi}_{k,n} = \frac{1}{\log(2)} \log \left(\frac{X_{(k)} - X_{(2k)}}{X_{(2k)} - X_{(4k)}} \right).$$

Unlike the Hill estimator it does not assume positive data, is valid for any ξ and is location and scale invariant. The Pickands estimator is defined on orders $k = 1, \dots, \lfloor n/4 \rfloor$.

Once a sufficiently low order statistic is reached the Pickand's estimator will be constant, upto sample uncertainty, for regularly varying tails. Pickand's plot is a plot of

$$\hat{\xi}_{k,n}$$

against the k . Symmetric asymptotic normal confidence intervals assuming Pareto tails are provided.

The Pickand's estimator is for the GPD shape ξ , or the reciprocal of the tail index $\alpha = 1/\xi$. The shape is plotted by default using `y.alpha=FALSE` and the tail index is plotted when `y.alpha=TRUE`.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The estimated parameter (ξ or α) at each threshold are plot by a horizontal solid line for all higher thresholds. The threshold should be set as low as possible, so a dashed line is shown below the pre-chosen threshold. If Pickand's estimator is similar to the dashed line then a lower threshold may be chosen.

If no order statistic (or threshold) limits are provided `orderlim = tlim = NULL` then the lowest order statistic is set to $X_{(1)}$ and highest possible value $X_{[n/4]}$. However, Pickand's estimator is always output for all $k = 1, \dots, [n/4]$.

The missing (NA and NaN) and non-finite values are ignored.

The lower x-axis is the order k . The upper axis is for the corresponding threshold.

Value

`pickandsplot` gives Pickand's plot. It also returns a dataframe containing columns of the order statistics, order, Pickand's estimator, it's standard deviation and $100(1 - \alpha)\%$ confidence interval (when requested).

Acknowledgments

Thanks to Younes Mouatasim, Risk Dynamics, Brussels for reporting various bugs in these functions.

Note

Asymptotic Wald type CI's are estimated for non-NULl significance level α for the shape parameter, assuming exactly GPD tails. When plotting on the tail index scale, then a simple reciprocal transform of the CI is applied which may well be sub-optimal.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

- Pickands III, J.. (1975). Statistical inference using extreme order statistics. Annal of Statistics 3(1), 119-131.
- Dekkers A. and de Haan, S. (1989). On the estimation of the extreme-value index and large quantile estimation. Annals of Statistics 17(4), 1795-1832.
- Resnick, S. (2007). Heavy-Tail Phenomena - Probabilistic and Statistical Modeling. Springer.

See Also[pickands](#)**Examples**

```
## Not run:
par(mfrow = c(2, 1))

# Reproduce graphs from Figure 4.7 of Resnick (2007)
data(danish, package="evir")

# Pickand's plot
pickandsplot(danish, orderlim=c(1, 150), ylim=c(-0.1, 2.2),
  try.thresh=c(), alpha=NULL, legend.loc=NULL)

# Using default settings
pickandsplot(danish)

## End(Not run)
```

psden

*P-Splines probability density function***Description**

Density, cumulative distribution function, quantile function and random number generation for the P-splines density estimate. B-spline coefficients can be result from Poisson regression with log or identity link.

Usage

```
dpsden(x, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, log = FALSE)

ppsdn(q, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, lower.tail = TRUE)

qpsdn(p, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL, lower.tail = TRUE)

rpsdn(n = 1, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, design.knots = NULL)
```

Arguments

x	quantiles
beta	vector of B-spline coefficients (required)
nbinwidth	scaling to convert count frequency into proper density
xrange	vector of minimum and maximum of B-spline (support of density)
nseg	number of segments between knots

degree	degree of B-splines (0 is constant, 1 is linear, etc.)
design.knots	spline knots for splineDesign function
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

P-spline density estimate using B-splines with given coefficients. B-splines knots can be specified using `design.knots` or regularly spaced knots can be specified using `xrange`, `nseg` and `deg`. No default knots are provided.

If regularly spaced knots are specified using `xrange`, `nseg` and `deg`, then B-splines which are shifted/spliced versions of each other are defined (i.e. not natural B-splines) which is consistent with definition of Eilers and Marx, the masters of P-splines.

The `splineDesign` function is used to calculate the B-splines, which intakes knot locations as `design.knots`. As such the `design.knots` are not the knots in their usual sense (e.g. to cover $[0, 100]$ with 10 segments the usual knots would be $0, 10, \dots, 100$). The `design.knots` must be extended by the degree, so for `degree = 2` the `design.knots = seq(-20, 120, 10)`.

Further, if the user wants natural B-splines then these can be specified using the `design.knots`, with replicated knots at each bounday according to the degree. To continue the above example, for `degree = 2` the `design.knots = c(rep(0, 2), seq(0, 100, 10), rep(100, 2))`.

If both the `design.knots` and other knot specification are provided, then the former are used by default. Default values for only the degree and `nseg` are provided, all the other P-spline inputs must be provided. Notice that the order and lambda penalty are not needed as these are encapsulated in the inference for the B-spline coefficients.

Poisson regression is typically used for estimating the B-spline coefficients, using maximum likelihood estimation (via iterative re-weighted least squares). A log-link function is usually used and as such the beta coefficients are on a log-scale, and the density needs to be exponentiated. However, an identity link may be (carefully) used and then these coefficients are on the usual scale.

The beta coefficients are estimated using a particular sample (size) and histogram bin-width, using Poisson regression. Thus to convert the predicted counts into a proper density it needs to be rescaled by dividing by $n * \text{binwidth}$. If `nbinwidth=NULL` is not provided then a crude approximate scaling is used by normalising the density to be proper. The renormalisation requires numerical integration, which is computationally intensive and so best avoided wherever possible.

Checks of the consistency of the `xrange`, degree and `nseg` and `design.knots` are made, with the values implied by the `design.knots` used by default to replace any incorrect values. These replacements are made for notational efficiency for users.

An inversion sampler is used for random number generation which also rather inefficient, as it could be carried out more efficiently using a mixture representation.

The quantile function is rather complicated as there is no closed form solution, so is obtained by numerical approximation of the inverse cumulative distribution function $P(X \leq q) = p$ to find q . The quantile function `qpsden` evaluates the P-splines cumulative distribution function over the `xrange`. A sequence of values of length fifty times the number of knots (with a minimum of 1000) is first calculated. Spline based interpolation using `splinefun`, with default `monoH.FC` method, is then used to approximate the quantile function. This is a similar approach to that taken by Matt Wand in the `qkde` in the `ks` package.

Value

[dpsden](#) gives the density, [ppsden](#) gives the cumulative distribution function, [qpsden](#) gives the quantile function and [rpsden](#) gives a random sample.

Note

Unlike most of the other extreme value mixture model functions the [psden](#) functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length.

Default values are provided for P-spline inputs of degree and nseg only, but all others must be provided by the user. The default sample size for [rpsden](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

References

<http://en.wikipedia.org/wiki/B-spline>

<http://www.stat.lsu.edu/faculty/marx/>

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science* 11(2), 89-121.

See Also

[splineDesign](#).

Other psden fpsden: [cvpsden](#), [cvpsden](#), [cvpsden](#), [cvpsden](#), [cvpsden](#), [fpsden](#), [fpsden](#), [fpsden](#), [fpsden](#), [fpsden](#), [iwlpsden](#), [iwlpsden](#), [iwlpsden](#), [iwlpsden](#), [iwlpsden](#), [lpsden](#), [lpsden](#), [lpsden](#), [lpsden](#), [nlpsden](#), [nlpsden](#), [nlpsden](#), [nlpsden](#), [nlpsden](#)

Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 8 internal segments
# CV search for penalty coefficient.
fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
             xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
psdensity = exp(fit$bsplines %*% fit$mle)
```

```

hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

# P-splines density from dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))

legend("topright", c("True Density", "P-spline density"), col=c("black", "blue"), lty = 1)

# plot B-splines
par(mfrow = c(2, 1))
with(fit, matplot(mids, as.matrix(bsplines), type = "l", lty = 1))

# Natural B-splines
knots = with(fit, seq(xrange[1], xrange[2], length.out = nseg + 1))
natural.knots = with(fit, c(rep(xrange[1], degree), knots, rep(xrange[2], degree)))
naturalb = splineDesign(natural.knots, fit$mids, ord = fit$degree + 1, outer.ok = TRUE)
with(fit, matplot(mids, naturalb, type = "l", lty = 1))

# Compare knot specifications
rbind(fit$design.knots, natural.knots)

# User can use natural B-splines if design.knots are specified manually
natural.fit = fpsden(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
                    design.knots = natural.knots, nseg = 10, degree = 3, ord = 2)
psdensity = with(natural.fit, exp(bsplines %*% mle))

par(mfrow = c(1, 1))
hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))
lines(xx, y, col = "black") # true density

# check density against dpsden function
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))
with(natural.fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots),
                        lwd = 2, col = "red", lty = 2))

legend("topright", c("True Density", "Eilers and Marx B-splines", "Natural B-splines"),
      col=c("black", "blue", "red"), lty = c(1, 1, 2))

## End(Not run)

```

psdengpd

P-Splines Density Estimate and GPD Tail Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with P-splines density estimate for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the B-spline coefficients beta (and associated features), threshold u GPD scale sigma_u and shape xi and tail fraction phi_u.

Usage

```
dpsdengpd(x, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, u = NULL, sigmau = NULL, xi = 0, phiu = TRUE,
  design.knots = NULL, log = FALSE)

ppsdengpd(q, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, u = NULL, sigmau = NULL, xi = 0, phiu = TRUE,
  design.knots = NULL, lower.tail = TRUE)

qpsdengpd(p, beta = NULL, nbinwidth = NULL, xrange = NULL, nseg = 10,
  degree = 3, u = NULL, sigmau = NULL, xi = 0, phiu = TRUE,
  design.knots = NULL, lower.tail = TRUE)

rpsdengpd(n = 1, beta = NULL, nbinwidth = NULL, xrange = NULL,
  nseg = 10, degree = 3, u = NULL, sigmau = NULL, xi = 0,
  phiu = TRUE, design.knots = NULL)
```

Arguments

x	quantiles
beta	vector of B-spline coefficients (required)
nbinwidth	scaling to convert count frequency into proper density
xrange	vector of minimum and maximum of B-spline (support of density)
nseg	number of segments between knots
degree	degree of B-splines (0 is constant, 1 is linear, etc.)
u	threshold
sigmau	scale parameter (positive)
xi	shape parameter
phiu	probability of being above threshold $[0, 1]$ or TRUE
design.knots	spline knots for splineDesign function
log	logical, if TRUE then log density
q	quantiles
lower.tail	logical, if FALSE then upper tail probabilities
p	cumulative probabilities
n	sample size (positive integer)

Details

Extreme value mixture model combining P-splines density estimate for the bulk below the threshold and GPD for upper tail.

The user can pre-specify `phiu` permitting a parameterised value for the tail fraction ϕ_u . Alternatively, when `phiu=TRUE` the tail fraction is estimated as the tail fraction from the KDE bulk model.

The cumulative distribution function with tail fraction ϕ_u defined by the upper tail fraction of the P-splines density estimate (`phiu=TRUE`), upto the threshold $x \leq u$, given by:

$$F(x) = H(x)$$

and above the threshold $x > u$:

$$F(x) = H(u) + [1 - H(u)]G(x)$$

where $H(x)$ and $G(X)$ are the P-splines density estimate and conditional GPD cumulative distribution functions respectively.

The cumulative distribution function for pre-specified 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. The specification of the underlying B-splines and the P-splines density estimator are discussed in the [psden](#) function help.

Value

[dpsdengpd](#) gives the density, [ppsdengpd](#) gives the cumulative distribution function, [qpsdengpd](#) gives the quantile function and [rpsdengpd](#) gives a random sample.

Note

Unlike most of the other extreme value mixture model functions the [psdengpd](#) functions have not been vectorised as this is not appropriate. The main inputs (x, p or q) must be either a scalar or a vector, which also define the output length. The B-splines coefficients beta and knots design.knots are vectors.

Default values are provided for P-spline inputs of degree and nseg only, but all others must be provided by the user. The default sample size for [rpsdengpd](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in x, p and q are passed through as is and infinite values are set to NA. None of these are permitted for the parameters/B-spline criteria.

Due to symmetry, the lower tail can be described by GPD by negating the quantiles.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Alfadino Akbar and Carl Scarrott <carl.scarrott@canterbury.ac.nz>.

References

<http://en.wikipedia.org/wiki/B-spline>

<http://www.stat.lsu.edu/faculty/marx/>

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science 11(2), 89-121.

See Also

[psden](#) and [fpsden](#).

Examples

```
## Not run:
set.seed(1)
par(mfrow = c(1, 1))

x = rnorm(1000)
xx = seq(-6, 6, 0.01)
y = dnorm(xx)

# Plenty of histogram bins (100)
breaks = seq(-4, 4, length.out=101)

# P-spline fitting with cubic B-splines, 2nd order penalty and 8 internal segments
# CV search for penalty coefficient.
fit = fpsdengpd(x, lambdaseq = 10^seq(-5, 5, 0.25), breaks = breaks,
               xrange = c(-4, 4), nseg = 10, degree = 3, ord = 2)
hist(x, freq = FALSE, breaks = seq(-4, 4, length.out=101), xlim = c(-6, 6))

# P-splines only
with(fit, lines(xx, dpsden(xx, beta, nbinwidth, design = design.knots), lwd = 2, col = "blue"))

# P-splines+GPD
with(fit, lines(xx, dpsdengpd(xx, beta, nbinwidth, design = design.knots,
                             u = u, sigmau = sigmau, xi = xi, phiu = phiu), lwd = 2, col = "red"))
abline(v = fit$u, col = "red")

legend("topleft", c("True Density", "P-spline density", "P-spline+GPD"),
      col=c("black", "blue", "red"), lty = 1)

## End(Not run)
```

tcplot

Parameter Threshold Stability Plots

Description

Plots the MLE of the GPD parameters against threshold

Usage

```
tcplot(data, tlim = NULL, nt = min(100, length(data)), p.or.n = FALSE,
       alpha = 0.05, ylim.xi = NULL, ylim.sigmau = NULL,
       legend.loc = "bottomright", try.thresh = quantile(data, 0.9, na.rm =
TRUE), ...)

tshapeplot(data, tlim = NULL, nt = min(100, length(data)), p.or.n = FALSE,
          alpha = 0.05, ylim = NULL, legend.loc = "bottomright",
          try.thresh = quantile(data, 0.9, na.rm = TRUE),
          main = "Shape Threshold Stability Plot", xlab = "Threshold u",
```

```

ylab = "Shape Parameter", ...)

tscaleplot(data, tlim = NULL, nt = min(100, length(data)), p.or.n = FALSE,
  alpha = 0.05, ylim = NULL, legend.loc = "bottomright",
  try.thresh = quantile(data, 0.9, na.rm = TRUE),
  main = "Modified Scale Threshold Stability Plot", xlab = "Threshold u",
  ylab = "Modified Scale Parameter", ...)

```

Arguments

<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>	significance level over range (0, 1), or NULL for no CI
<code>ylim.xi</code>	y-axis limits for shape parameter or NULL
<code>ylim.sigmau</code>	y-axis limits for scale parameter or NULL
<code>legend.loc</code>	location of legend (see legend) or NULL for no legend
<code>try.thresh</code>	vector of thresholds to consider
<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 a set of possible thresholds. If the GPD is a suitable model for a threshold u then for all higher thresholds $v > u$ it will also be suitable, with the shape and modified scale being constant. Known as the threshold stability plots (Coles, 2001). The modified scale parameter is $\sigma_u - u\xi$.

In practice there is sample uncertainty in the parameter estimates, which must be taken into account when choosing a threshold.

The usual asymptotic Wald confidence intervals are shown based on the observed information matrix to measure this uncertainty. The sampling density of the Wald normal approximation is shown by a greyscale image, where lighter greys indicate low density.

A pre-chosen threshold (or more than one) can be given in `try.thresh`. The GPD is fitted to the excesses using maximum likelihood estimation. The estimated parameters are shown as a horizontal line which is solid above this threshold, for which they should be the same if the GPD is a good model (upto sample uncertainty). The threshold should always be chosen to be as low as possible to reduce sample uncertainty. Therefore, below the pre-chosen threshold, where the GPD should not be a good model, the line is dashed and the parameter estimates should now deviate from the dashed line (otherwise a lower threshold could be used). If no threshold limits are provided `tlim = NULL` then the lowest threshold is set to be just below the median data point and the maximum threshold is set to the 11th largest datapoint. This is a slightly lower order statistic compared to that used in

the MRL plot `mrlplot` function to account for the fact the maximum likelihood estimation is likely to be unreliable with 10 or fewer datapoints.

The range of permitted thresholds is just below the minimum datapoint and the second largest value. If there are less unique values of data within the threshold range than the number of threshold 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 if requested. Where the observed information matrix is not obtainable the standard deviation and confidence intervals are NA. For the `tscaleplot` the modified scale quantities are also provided. `tcplot` produces both plots on one graph and outputs a merged dataframe of results.

Acknowledgments

Based on the threshold stability plot function `tcplot` in the `evd` package for which Stuart Coles' and Alec Stephenson's contributions are gratefully acknowledged. They are designed to have similar syntax and functionality to simplify the transition for users of these packages.

Note

If the user specifies the threshold range, the thresholds above the sixth largest are dropped. A warning message is given if any thresholds have at most 10 exceedances, in which case the maximum likelihood estimation is unreliable. If there are less than 10 exceedances of the minimum threshold then the function will stop.

By default, no legend is included when using `tcplot` to get both threshold stability plots.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

- Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. *REVSTAT - Statistical Journal* 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>
- Coles S.G. (2004). *An Introduction to the Statistical Modelling of Extreme Values*. Springer-Verlag: London.

See Also

`mrlplot` and `tcplot` from `evd` library

Examples

```
## Not run:
x = rnorm(1000)
tcplot(x)
tshapeplot(x, tlim = c(0, 2))
tscaleplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))
tcplot(x, tlim = c(0, 2), try.thresh = c(0.5, 1, 1.5))

## End(Not run)
```

weibullgpd

Weibull Bulk and GPD Tail Extreme Value Mixture Model

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold. The parameters are the weibull shape *wshape* and scale *wscale*, threshold *u* GPD scale *sigmau* and shape *xi* and tail fraction *phiu*.

Usage

```
dweibullgpd(x, wshape = 1, wscale = 1, u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
    1/wshape))^2), xi = 0, phiu = TRUE, log = FALSE)

pweibullgpd(q, wshape = 1, wscale = 1, u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
    1/wshape))^2), xi = 0, phiu = TRUE, lower.tail = TRUE)

qweibullgpd(p, wshape = 1, wscale = 1, u = qweibull(0.9, wshape, wscale),
  sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
    1/wshape))^2), xi = 0, phiu = TRUE, lower.tail = TRUE)

rweibullgpd(n = 1, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), sigmau = sqrt(wscale^2 * gamma(1 + 2/wshape) - (wscale * gamma(1 +
    1/wshape))^2), xi = 0, phiu = TRUE)
```

Arguments

<i>x</i>	quantiles
<i>wshape</i>	Weibull shape (positive)
<i>wscale</i>	Weibull scale (positive)
<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

<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 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, 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 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](#), [dweibullgpdcon](#), [dweibullgpdcon](#), [dweibullgpdcon](#), [dweibullgpdcon](#), [pweibullgpdcon](#), [pweibullgpdcon](#), [pweibullgpdcon](#), [pweibullgpdcon](#), [pweibullgpdcon](#), [qweibullgpdcon](#), [qweibullgpdcon](#), [qweibullgpdcon](#), [qweibullgpdcon](#), [qweibullgpdcon](#), [rweibullgpdcon](#), [rweibullgpdcon](#), [rweibullgpdcon](#), [rweibullgpdcon](#), [rweibullgpdcon](#), [rweibullgpdcon](#), [weibullgpdcon](#), [weibullgpdcon](#), [weibullgpdcon](#), [weibullgpdcon](#), [weibullgpdcon](#)

Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

x = rweibullgpd(1000)
xx = seq(-1, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx))

# three tail behaviours
plot(xx, pweibullgpd(xx), type = "l")
lines(xx, pweibullgpd(xx, xi = 0.3), col = "red")
lines(xx, pweibullgpd(xx, xi = -0.3), col = "blue")
legend("topleft", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

x = rweibullgpd(1000, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpd(xx, phiu = 0.2))

plot(xx, dweibullgpd(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dweibullgpd(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dweibullgpd(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topleft", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

weibullgpdcon	<i>Weibull Bulk and GPD Tail Extreme Value Mixture Model with Single Continuity Constraint</i>
---------------	--

Description

Density, cumulative distribution function, quantile function and random number generation for the extreme value mixture model with Weibull for bulk distribution upto the threshold and conditional GPD above threshold with continuity at threshold. The parameters are the weibull shape *wshape* and scale *wscale*, threshold *u* GPD shape *xi* and tail fraction *phiu*.

Usage

```
dweibullgpdcon(x, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), xi = 0, phiu = TRUE, log = FALSE)

pweibullgpdcon(q, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), xi = 0, phiu = TRUE, lower.tail = TRUE)

qweibullgpdcon(p, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), xi = 0, phiu = TRUE, lower.tail = TRUE)

rweibullgpdcon(n = 1, wshape = 1, wscale = 1, u = qweibull(0.9, wshape,
  wscale), xi = 0, phiu = TRUE)
```

Arguments

<i>x</i>	quantiles
<i>wshape</i>	Weibull shape (positive)
<i>wscale</i>	Weibull scale (positive)
<i>u</i>	threshold
<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 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, 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 Weibull and conditional GPD density functions (i.e. `dweibull(x, wshape, wscale)` and `dgp(x, u, sigmau, xi)`) respectively. The resulting GPD scale parameter is then:

$$\sigma_u = \phi_u H(u) / [1 - \phi_u] h(u)$$

. In the special case of where the tail fraction is defined by the bulk model this reduces to

$$\sigma_u = [1 - H(u)] / h(u)$$

.

The Weibull is defined on the non-negative reals, so the threshold must be positive.

See [gpd](#) for details of GPD upper tail component and [dweibull](#) for details of weibull bulk component.

Value

[dweibullgpdcon](#) gives the density, [pweibullgpdcon](#) gives the cumulative distribution function, [qweibullgpdcon](#) gives the quantile function and [rweibullgpdcon](#) gives a random sample.

Acknowledgments

Thanks to Ben Youngman, Exeter University, UK for reporting a bug in the [rweibullgpdcon](#) function.

Note

All inputs are vectorised except `log` and `lower.tail`. The main inputs (`x`, `p` or `q`) and parameters must be either a scalar or a vector. If vectors are provided they must all be of the same length, and the function will be evaluated for each element of vector. In the case of [rweibullgpdcon](#) any input vector must be of length `n`.

Default values are provided for all inputs, except for the fundamentals `x`, `q` and `p`. The default sample size for [rweibullgpdcon](#) is 1.

Missing (NA) and Not-a-Number (NaN) values in `x`, `p` and `q` are passed through as is and infinite values are set to NA. None of these are not permitted for the parameters.

Error checking of the inputs (e.g. invalid probabilities) is carried out and will either stop or give warning message as appropriate.

Author(s)

Yang Hu and Carl Scarrott <carl.scarrott@canterbury.ac.nz>

References

http://en.wikipedia.org/wiki/Weibull_distribution

http://en.wikipedia.org/wiki/Generalized_Pareto_distribution

Scarrott, C.J. and MacDonald, A. (2012). A review of extreme value threshold estimation and uncertainty quantification. REVSTAT - Statistical Journal 10(1), 33-59. Available from <http://www.ine.pt/revstat/pdf/rs120102.pdf>

Behrens, C.N., Lopes, H.F. and Gamerman, D. (2004). Bayesian analysis of extreme events with threshold estimation. Statistical Modelling. 4(3), 227-244.

See Also

[gpd](#) and [dweibull](#)

Other weibullgpd weibullgpdcon fweibullgpd fweibullgpdcon: [dweibullgpd](#), [dweibullgpd](#), [dweibullgpd](#), [dweibullgpd](#), [dweibullgpd](#), [pweibullgpd](#), [pweibullgpd](#), [pweibullgpd](#), [pweibullgpd](#), [pweibullgpd](#), [qweibullgpd](#), [qweibullgpd](#), [qweibullgpd](#), [qweibullgpd](#), [qweibullgpd](#), [rweibullgpd](#), [rweibullgpd](#), [rweibullgpd](#), [rweibullgpd](#), [weibullgpd](#), [weibullgpd](#), [weibullgpd](#), [weibullgpd](#), [weibullgpd](#)

Examples

```
## Not run:
set.seed(1)
par(mfrow = c(2, 2))

x = rweibullgpdcon(1000)
xx = seq(-0.1, 6, 0.01)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpdcon(xx))

# three tail behaviours
plot(xx, pweibullgpdcon(xx), type = "l")
lines(xx, pweibullgpdcon(xx, xi = 0.3), col = "red")
lines(xx, pweibullgpdcon(xx, xi = -0.3), col = "blue")
legend("bottomright", paste("xi =", c(0, 0.3, -0.3)),
      col=c("black", "red", "blue"), lty = 1)

x = rweibullgpdcon(1000, phiu = 0.2)
hist(x, breaks = 100, freq = FALSE, xlim = c(-1, 6))
lines(xx, dweibullgpdcon(xx, phiu = 0.2))

plot(xx, dweibullgpdcon(xx, xi=0, phiu = 0.2), type = "l")
lines(xx, dweibullgpdcon(xx, xi=-0.2, phiu = 0.2), col = "red")
lines(xx, dweibullgpdcon(xx, xi=0.2, phiu = 0.2), col = "blue")
legend("topright", c("xi = 0", "xi = 0.2", "xi = -0.2"),
      col=c("black", "red", "blue"), lty = 1)

## End(Not run)
```

Index

bckden, [4](#), [7](#), [13](#), [17](#), [35](#), [106](#), [196](#), [208](#), [212](#),
[216](#), [219](#), [221](#)

bckdengpd, [8](#), [9](#), [12](#), [17](#), [35](#), [106](#), [208](#), [212](#), [216](#)

bckdengpdcon, [8](#), [13](#), [14](#), [16](#), [35](#), [106](#), [208](#),
[212](#), [216](#)

bckdenxbeta1 (internal), [194](#)

bckdenxbeta2 (internal), [194](#)

bckdenxcopula (internal), [194](#)

bckdenxcutnorm (internal), [194](#)

bckdenxgamma1 (internal), [194](#)

bckdenxgamma2 (internal), [194](#)

bckdenxreflect (internal), [194](#)

bckdenxrenorm (internal), [194](#)

bckdenxsimple (internal), [194](#)

betagpd, [18](#), [23](#)

betagpdcon, [20](#), [21](#)

bw.nrd0, [8](#), [13](#), [17](#), [35](#), [40](#), [45](#), [66](#), [67](#), [71](#), [72](#),
[105](#), [106](#), [110](#), [114](#), [170](#), [171](#), [174](#),
[175](#), [207](#), [208](#), [211](#), [212](#), [215](#), [216](#)

- check.bcmethod (checking), 24
- check.control (checking), 24
- check.design.knots (checking), 24
- check.inputn, 26
- check.inputn (checking), 24
- check.kbw, 219
- check.kbw (kfun), 220
- check.kernel, 219, 221
- check.kernel (kfun), 220
- check.kinputs, 219, 221
- check.kinputs (kfun), 220
- check.klambda, 221
- check.logic (checking), 24
- check.n (checking), 24
- check.nn (checking), 24
- check.nparam (checking), 24
- check.offset (checking), 24
- check.optim (checking), 24
- check.param, 24
- check.param (checking), 24
- check.phiu (checking), 24
- check.posparam, 24
- check.posparam (checking), 24
- check.prob (checking), 24

check.quant (checking), 24
check.text (checking), 24
checking, 24
condmixt, 4, 87, 91, 191, 194
cvpsden, 148, 253
cvpsden (fpsden), 146

dbckden, 6, 11–13, 16, 17, 35, 39, 43, 44, 106, 208, 212, 216
dbckden (bckden), 4
dbckdengpd, 8, 12, 17, 35, 106, 208, 212, 216
dbckdengpd (bckdengpd), 9
dbckdengpdcon, 8, 13, 16, 35, 42, 106, 208, 212, 216
dbckdengpdcon (bckdengpdcon), 14
dbeta, 20, 22, 23, 48, 52
dbetagpd, 20, 23
dbetagpd (betagpd), 18
dbetagpdcon, 20, 23, 50
dbetagpdcon (betagpdcon), 21
dcauchy, 28
ddwm, 27
ddwm (dwm), 26
density, 5, 8, 11, 13, 15, 17, 30, 32, 35, 38, 40, 43, 45, 65–67, 70–72, 103, 106, 109, 110, 113, 114, 169–171, 173–175, 196, 207, 208, 211, 212, 214–216, 218, 219, 221
densplot, 30
densplot (evmix.diag), 29
dgamma, 58, 62, 125, 130, 136, 162, 163, 166, 167, 231, 235, 239
dgamma_{gpd}, 58, 62, 125, 130, 136, 162, 167, 231, 235, 239
dgamma_{gpd} (gamma_{gpd}), 161
dgamma_{gpd}con, 58, 60, 62, 125, 130, 136, 163, 166, 231, 235, 239
dgamma_{gpd}con (gamma_{gpd}con), 164
dgkg, 170, 175
dgkg (gkg), 168
dgkgcon, 70, 171, 174
dgkgcon (gkgcon), 172
dgng, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248

- dgng (gng), 176
- dgngcon, 76, 79, 81, 95, 142, 145, 178, 182, 199, 244, 248
- dgngcon (gngcon), 180
- dgpd, 85, 185
- dgpd (gpd), 184
- dhpd, 190, 194
- dhpd (hpd), 189
- dhpdcon, 89, 191, 193
- dhpdcon (hpdcon), 192
- ditmgng, 76, 81, 93, 95, 142, 145, 178, 182, 198, 244, 248
- ditmgng (itmngng), 196
- ditmnormgpd, 97, 198, 201
- ditmnormgpd (itmnormgpd), 200
- ditmweibullgpd, 100, 204
- ditmweibullgpd (itmweibullgpd), 203
- dkde, 8, 13, 17, 40, 45, 67, 72, 110, 114, 171, 175, 208, 212, 216
- dkden, 8, 13, 17, 35, 106, 170, 174, 207, 211, 212, 215, 216
- dkden (kden), 206
- dkdengpd, 8, 13, 17, 35, 106, 208, 211, 216
- dkdengpd (kdengpd), 210
- dkdengpdcon, 8, 13, 17, 35, 106, 113, 208, 212, 215
- dkdengpdcon (kdengpdcon), 213
- dlnorm, 118, 121, 224, 225, 227, 228
- dlognormgpd, 224, 228
- dlognormgpd (lognormgpd), 223
- dlognormgpdcon, 120, 225, 227
- dlognormgpdcon (lognormgpdcon), 226
- dmgamma, 58, 62, 126, 131, 136, 163, 167, 230, 235, 239
- dmgamma (mgamma), 229
- dmgammagpd, 58, 62, 125, 130, 136, 163, 167, 231, 234, 239
- dmgammagpd (mgammagpd), 232
- dmgammagpdcon, 58, 62, 125, 130, 136, 163, 167, 231, 235, 238
- dmgammagpdcon (mgammagpdcon), 236
- dnorm, 76, 81, 91, 95, 98, 142, 145, 178, 182, 190, 191, 193, 194, 199, 202, 243, 244, 247, 248
- dnormgpd, 76, 81, 95, 142, 145, 178, 179, 182, 183, 199, 243, 248
- dnormgpd (normgpd), 242
- dnormgpdcon, 76, 81, 95, 142, 144, 145, 179, 183, 199, 244, 247
- dnormgpdcon (normgpdcon), 245
- dpsden, 148, 149, 253
- dpsden (psden), 251
- dpsdengpd, 256
- dpsdengpd (psdengpd), 254
- dweibull, 28, 101, 156, 160, 205, 261, 262, 264, 265
- dweibullgpd, 261, 265
- dweibullgpd (weibullgpd), 260
- dweibullgpdcon, 158, 262, 264
- dweibullgpdcon (weibullgpdcon), 263
- dwm, 26
- evd, 4, 29, 30, 38, 43, 47, 51, 53, 54, 57, 61, 66, 71, 75, 79, 83, 84, 86, 90, 104, 109, 113, 117, 120, 129, 134, 140, 141, 144, 152, 155, 159, 184–186, 241, 242, 259
- evmix, 29
- evmix (evmix-package), 2
- evmix-package, 2
- evmix.diag, 29, 30
- fbckden, 6, 9, 11, 13, 16, 18, 31, 33, 106, 208, 212, 216
- fbckdengpd, 11, 34, 36, 38, 45, 110, 115
- fbckdengpdcon, 16, 40, 41, 43, 110, 114
- fbetagpd, 46, 47, 52
- fbetagpdcon, 48, 49, 50
- fdwm, 52, 54
- fgammagpd, 55, 57, 62, 126, 131, 136, 163, 167, 231, 235, 239
- fgammagpdcon, 58, 59, 61, 126, 131, 136, 163, 167, 231, 235, 239
- fgkg, 63, 65, 72, 105
- fgkgcon, 67, 68, 70
- fgng, 30, 64, 69, 70, 73, 73, 74, 75, 78, 79, 81, 95, 142, 145, 179, 183, 199, 244, 248
- fgngcon, 76, 78, 79, 95, 142, 145, 179, 183, 199, 244, 248
- fgpd, 29, 30, 40, 45, 48, 52, 55, 58, 62, 67, 72, 76, 81, 82, 83, 87, 91, 95, 98, 101, 110, 114, 118, 121, 130, 136, 142, 145, 153, 156, 160, 186
- fhpdp, 85, 86, 91
- fhpdpcon, 87, 88, 89
- fitdistr, 85
- fitmgng, 76, 81, 92, 93, 142, 146, 179, 183, 199, 244, 248
- fitmnormgpd, 96, 97
- fitmweibullgpd, 99, 100
- fkden, 9, 13, 18, 35, 38, 39, 43, 44, 65, 66, 70, 71, 102, 104, 109, 113, 209, 212, 216
- fkdengpd, 40, 45, 105, 107, 109, 115
- fkdengpdcon, 40, 45, 110, 111, 113
- flognormgpd, 115, 117, 121

- flognormgpdcon, [118](#), [119](#), [120](#)
- fmgamma, [58](#), [62](#), [122](#), [124](#), [131](#), [136](#), [164](#), [167](#), [231](#), [232](#), [235](#), [239](#)
- fmgammagpd, [58](#), [62](#), [124](#), [126](#), [126](#), [129](#), [136](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- fmgammagpdcon, [58](#), [62](#), [126](#), [131](#), [132](#), [134](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- fnormgpd, [29](#), [30](#), [37–39](#), [42](#), [44](#), [46](#), [47](#), [50](#), [51](#), [54](#), [56](#), [57](#), [60](#), [61](#), [65](#), [66](#), [70](#), [71](#), [74](#), [75](#), [77](#), [79–81](#), [89](#), [90](#), [93–95](#), [97](#), [98](#), [100](#), [101](#), [105](#), [108](#), [109](#), [112–114](#), [116](#), [117](#), [119–121](#), [127–129](#), [133](#), [135](#), [137](#), [138–140](#), [143–146](#), [151–156](#), [158](#), [159](#), [179](#), [183](#), [199](#), [245](#), [248](#)
- fnormgpdcon, [76](#), [77](#), [81](#), [95](#), [142](#), [143](#), [144](#), [179](#), [183](#), [199](#), [245](#), [248](#)
- fpgd, [84](#)
- fpot, [34](#), [54](#), [84](#), [85](#), [87](#), [140](#), [186](#)
- fpsden, [146](#), [148](#), [151–153](#), [253](#), [257](#)
- fpsdengpd, [150](#), [152](#)
- fweibullgpd, [154](#), [155](#), [160](#)
- fweibullgpdcon, [156](#), [157](#), [159](#)
- gammagpd, [58](#), [62](#), [125](#), [130](#), [136](#), [161](#), [167](#), [231](#), [234](#), [235](#), [238](#), [239](#)
- gammagpdcon, [58](#), [62](#), [125](#), [130](#), [136](#), [163](#), [164](#), [231](#), [235](#), [239](#)
- gammamixEM, [125](#)
- gkg, [168](#), [170](#), [175](#)
- gkgcon, [171](#), [172](#), [174](#)
- gng, [76](#), [81](#), [95](#), [142](#), [145](#), [176](#), [182](#), [199](#), [244](#), [248](#)
- gngcon, [76](#), [81](#), [95](#), [142](#), [145](#), [178](#), [180](#), [199](#), [244](#), [248](#)
- gpd, [11](#), [13](#), [16](#), [17](#), [19](#), [20](#), [22](#), [23](#), [28](#), [40](#), [45](#), [48](#), [52](#), [55](#), [58](#), [62](#), [67](#), [72](#), [76](#), [81](#), [85](#), [87](#), [91](#), [95](#), [98](#), [101](#), [110](#), [114](#), [118](#), [121](#), [130](#), [136](#), [142](#), [145](#), [153](#), [156](#), [160](#), [162](#), [163](#), [166](#), [167](#), [170](#), [174](#), [178](#), [182](#), [184](#), [185](#), [190](#), [191](#), [193](#), [194](#), [199](#), [202](#), [205](#), [211](#), [215](#), [224](#), [225](#), [227](#), [228](#), [231](#), [235](#), [239](#), [242–244](#), [247](#), [248](#), [256](#), [261](#), [262](#), [264](#), [265](#)
- gpd.diag, [31](#)
- gpd.fit, [84](#)
- hill, [189](#)
- hillplot, [186](#), [188](#)
- hist, [147](#), [151](#)
- hpareto, [191](#), [194](#)
- hpareto.fit, [87](#), [91](#)
- hpareto.negloglike, [87](#), [91](#)
- hparetomixt, [191](#), [194](#)
- hpd, [189](#), [194](#)
- hpdcon, [191](#), [192](#)
- integrate, [7](#), [12](#), [16](#)
- internal, [194](#)
- ismev, [4](#), [84](#), [141](#)
- itmgng, [76](#), [81](#), [95](#), [142](#), [145](#), [178](#), [182](#), [196](#), [244](#), [248](#)
- itmnormgpd, [200](#)
- itmweibullgpd, [203](#)
- iwlspdsden, [148](#), [253](#)
- iwlspdsden (fpsden), [146](#)
- jitter, [32](#), [34](#), [35](#), [37](#), [42](#), [64](#), [69](#), [102](#), [105](#), [106](#), [108](#), [112](#)
- ka0, [219](#), [221](#)
- ka0 (kfun), [220](#)
- ka1, [219](#), [221](#)
- ka1 (kfun), [220](#)
- ka2, [219](#), [221](#)
- ka2 (kfun), [220](#)
- kbw, [219](#), [221](#)
- kbw (kfun), [220](#)
- kd*, [219](#)
- kdbiweight, [221](#)
- kdbiweight (kernels), [217](#)
- kdcosine, [221](#)
- kdcosine (kernels), [217](#)
- kden, [5](#), [8](#), [13](#), [17](#), [35](#), [106](#), [149](#), [196](#), [206](#), [207](#), [212](#), [216](#), [219](#), [221](#)
- kdengpd, [8](#), [13](#), [17](#), [35](#), [106](#), [208](#), [210](#), [211](#), [216](#)
- kdengpdcon, [8](#), [13](#), [17](#), [35](#), [106](#), [208](#), [212](#), [213](#), [215](#)
- kdenx (internal), [194](#)
- kdepanechnikov, [221](#)
- kdepanechnikov (kernels), [217](#)
- kdgaussian, [221](#)
- kdgaussian (kernels), [217](#)
- kdoptcosine, [221](#), [222](#)
- kdoptcosine (kernels), [217](#)
- kdparzen, [222](#)
- kdparzen (kernels), [217](#)
- kdtriangular, [222](#)
- kdtriangular (kernels), [217](#)
- kdtricube, [222](#)
- kdtricube (kernels), [217](#)
- kdtriweight, [222](#)
- kdtriweight (kernels), [217](#)
- kduniform, [222](#)
- kduniform (kernels), [217](#)

- kdz, [219](#), [222](#)
- kdz (kernels), [217](#)
- kernels, [5](#), [6](#), [8](#), [11](#), [13](#), [15](#), [17](#), [32](#), [35](#), [38](#), [40](#), [43](#), [45](#), [65](#), [67](#), [70](#), [72](#), [103](#), [106](#), [109](#), [110](#), [113](#), [114](#), [169](#), [171](#), [173](#), [175](#), [207](#), [208](#), [211](#), [212](#), [214](#), [216](#), [217](#), [221](#), [222](#)
- kfun, [8](#), [13](#), [17](#), [35](#), [40](#), [45](#), [67](#), [72](#), [106](#), [110](#), [114](#), [171](#), [175](#), [208](#), [212](#), [216](#), [219](#), [220](#)
- klambda, [105](#), [219](#), [221](#)
- klambda (kfun), [220](#)
- kp*, [219](#)
- kpbiweight, [222](#)
- kpbiweight (kernels), [217](#)
- kpcosine, [222](#)
- kpcosine (kernels), [217](#)
- kpepanechnikov, [222](#)
- kpepanechnikov (kernels), [217](#)
- kpgaussian, [222](#)
- kpgaussian (kernels), [217](#)
- kpoptcosine, [222](#)
- kpoptcosine (kernels), [217](#)
- kparzen, [222](#)
- kparzen (kernels), [217](#)
- kptriangular, [222](#)
- kptriangular (kernels), [217](#)
- kptricube, [222](#)
- kptricube (kernels), [217](#)
- kptriweight, [222](#)
- kptriweight (kernels), [217](#)
- kpu, [221](#)
- kpuniform, [222](#), [223](#)
- kpuniform (kernels), [217](#)
- kpz, [219](#), [223](#)
- kpz (kernels), [217](#)
- ks, [6–8](#), [13](#), [17](#), [40](#), [45](#), [67](#), [72](#), [110](#), [114](#), [171](#), [175](#), [207](#), [208](#), [212](#), [216](#), [252](#)
- lbckden, [9](#), [13](#), [18](#), [32](#), [33](#), [106](#), [209](#), [212](#), [216](#)
- lbckden (fbckden), [31](#)
- lbckdengpd, [38](#), [45](#), [110](#), [115](#)
- lbckdengpd (fbckdengpd), [36](#)
- lbckdengpdcon, [40](#), [43](#), [110](#), [114](#)
- lbckdengpdcon (fbckdengpdcon), [41](#)
- lbetagpd, [47](#), [52](#)
- lbetagpd (fbetagpd), [46](#)
- lbetagpdcon, [48](#), [50](#)
- lbetagpdcon (fbetagpdcon), [49](#)
- ldwm, [53](#), [54](#)
- ldwm (fdwm), [52](#)
- legend, [187](#), [240](#), [249](#), [258](#)
- lgammagpd, [57](#), [62](#), [126](#), [131](#), [136](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- lgammagpd (fgammagpd), [55](#)
- lgammagpdcon, [58](#), [61](#), [126](#), [131](#), [136](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- lgammagpdcon (fgammagpdcon), [59](#)
- lgkg, [65](#), [72](#)
- lgkg (fgkg), [63](#)
- lgkgcon, [67](#), [70](#)
- lgkgcon (fgkgcon), [68](#)
- lng, [75](#), [81](#), [95](#), [142](#), [145](#), [179](#), [183](#), [199](#), [244](#), [248](#)
- lng (fgng), [73](#)
- lngcon, [76](#), [79](#), [95](#), [142](#), [145](#), [179](#), [183](#), [199](#), [244](#), [248](#)
- lngcon (fgngcon), [78](#)
- lgpd, [83](#), [84](#), [186](#)
- lgpd (fgpd), [82](#)
- lhp, [86](#), [87](#), [91](#)
- lhp (fhpd), [85](#)
- lhpdcon, [87](#), [89](#)
- lhpdcon (fhpdcon), [88](#)
- litmgng, [76](#), [81](#), [93](#), [142](#), [146](#), [179](#), [183](#), [199](#), [244](#), [248](#)
- litmgng (fitmgng), [92](#)
- litnormgpd, [97](#)
- litnormgpd (fitnormgpd), [96](#)
- litmweibullgpd, [100](#)
- litmweibullgpd (fitmweibullgpd), [99](#)
- lkden, [9](#), [13](#), [18](#), [35](#), [104](#), [209](#), [212](#), [216](#)
- lkden (fkden), [102](#)
- lkdengpd, [40](#), [45](#), [109](#), [115](#)
- lkdengpd (fkden), [107](#)
- lkdengpdcon, [40](#), [45](#), [110](#), [113](#)
- lkdengpdcon (fkden), [111](#)
- llognormgpd, [117](#), [121](#)
- llognormgpd (flognormgpd), [115](#)
- llognormgpdcon, [118](#), [120](#)
- llognormgpdcon (flognormgpdcon), [119](#)
- lmgamma, [58](#), [62](#), [123–125](#), [131](#), [136](#), [164](#), [167](#), [232](#), [235](#), [239](#)
- lmgamma (fmgamma), [122](#)
- lmgammagpd, [58](#), [62](#), [126](#), [128–130](#), [136](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- lmgammagpd (fmammagpd), [126](#)
- lmgammagpdcon, [58](#), [62](#), [126](#), [131](#), [133–135](#), [163](#), [167](#), [231](#), [235](#), [239](#)
- lmgammagpdcon (fmammagpdcon), [132](#)
- lnormgpd, [77](#), [81](#), [95](#), [104](#), [123](#), [139–141](#), [146](#), [179](#), [183](#), [199](#), [245](#), [248](#)
- lnormgpd (fnormgpd), [137](#)
- lnormgpdcon, [77](#), [81](#), [95](#), [142](#), [144](#), [179](#), [183](#),

- 199, 245, 248
 lnormgpdcon (fnormgpdcon), 143
 lognormgpd, 223, 228
 lognormgpdcon, 225, 226
 lpgd, 84
 lpsden, 148, 253
 lpsden (fpsden), 146
 lpsdengpd, 152
 lpsdengpd (fpsdengpd), 150
 lweibullgpd, 155, 160
 lweibullgpd (fweibullgpd), 154
 lweibullgpdcon, 156, 159
 lweibullgpdcon (fweibullgpdcon), 157

 mgamma, 58, 62, 126, 131, 136, 163, 167, 229, 235, 239
 mgammagpd, 58, 62, 125, 130, 136, 163, 167, 231, 232, 239
 mgammagpdcon, 58, 62, 125, 130, 136, 163, 167, 231, 235, 236
 mrlplot, 240, 241, 242, 259

 nlbckden, 9, 13, 18, 32, 33, 106, 209, 212, 216
 nlbckden (fbckden), 31
 nlbckdengpd, 38, 45, 110, 115
 nlbckdengpd (fbckdengpd), 36
 nlbckdengpdcon, 40, 43, 110, 114
 nlbckdengpdcon (fbckdengpdcon), 41
 nlbetagpd, 47, 52
 nlbetagpd (fbetagpd), 46
 nlbetagpdcon, 48, 50
 nlbetagpdcon (fbetagpdcon), 49
 nldwm, 53, 54
 nldwm (fdwm), 52
 nlEMgamma, 58, 62, 123–125, 131, 136, 164, 167, 232, 235, 239
 nlEMgamma (fmgamma), 122
 nlEMgammagpd, 58, 62, 126, 128–130, 136, 163, 167, 231, 235, 239
 nlEMgammagpd (fmgammagpd), 126
 nlEMgammagpdcon, 58, 62, 126, 131, 133–135, 163, 167, 231, 235, 239
 nlEMgammagpdcon (fmgammagpdcon), 132
 nleuitmgng, 76, 81, 142, 146, 179, 183, 199, 244, 248
 nleuitmgng (fitmgng), 92
 nleuitnormgpd (fitnormgpd), 96
 nleuitweibullgpd (fitweibullgpd), 99
 nlgamma, 57, 62, 126, 131, 136, 163, 167, 231, 235, 239
 nlgamma (fgamma), 55
 nlgamma, 58, 61, 126, 131, 136, 163, 167, 231, 235, 239
 nlgamma (fgamma), 59
 nlkg, 65, 72
 nlkg (fgkg), 63
 nlkgcon, 67, 70
 nlkgcon (fgkgcon), 68
 nlng, 75, 81, 95, 142, 145, 179, 183, 199, 244, 248
 nlng (fgng), 73
 nlngcon, 76, 79, 95, 142, 145, 179, 183, 199, 244, 248
 nlngcon (fgngcon), 78
 nlpgd, 83, 84, 186
 nlpgd (fgpd), 82
 nlhpd, 86, 87, 91
 nlhpd (fhpd), 85
 nlhpdcon, 87, 89
 nlhpdcon (fhpdcon), 88
 nlitmng, 76, 81, 93, 142, 146, 179, 183, 199, 244, 248
 nlitmng (fitmgng), 92
 nlitmnormgpd, 97
 nlitmnormgpd (fitnormgpd), 96
 nlitmweibullgpd, 100
 nlitmweibullgpd (fitweibullgpd), 99
 nlkden, 9, 13, 18, 35, 104, 209, 212, 216
 nlkden (fkden), 102
 nlkdengpd, 40, 45, 109, 115
 nlkdengpd (fkden), 107
 nlkdengpdcon, 40, 45, 110, 113
 nlkdengpdcon (fkden), 111
 nllognormgpd, 117, 121
 nllognormgpd (flognormgpd), 115
 nllognormgpdcon, 118, 120
 nllognormgpdcon (flognormgpdcon), 119
 nlmgamma, 58, 62, 123–125, 131, 136, 164, 167, 232, 235, 239
 nlmgamma (fmgamma), 122
 nlmgammagpd, 58, 62, 126, 128–130, 136, 163, 167, 231, 235, 239
 nlmgammagpd (fmgammagpd), 126
 nlmgammagpdcon, 58, 62, 126, 131, 133–135, 163, 167, 231, 235, 239
 nlmgammagpdcon (fmgammagpdcon), 132
 nlnormgpd, 77, 81, 95, 139–141, 146, 179, 183, 199, 245, 248
 nlnormgpd (fnormgpd), 137
 nlnormgpdcon, 77, 81, 95, 142, 144, 179, 183, 199, 245, 248
 nlnormgpdcon (fnormgpdcon), 143
 nlornlkdenmgpd, 104
 nlpgd, 84
 nlpsden, 148, 253

- nlpsden (fpsden), 146
- nlpsdengpd, 152
- nlpsdengpd (fpsdengpd), 150
- nlubckdengpd, 38, 45, 110, 115
- nlubckdengpd (fbckdengpd), 36
- nlubckdengpdcon, 40, 43, 110, 114
- nlubckdengpdcon (fbckdengpdcon), 41
- nlubetapd, 47, 52
- nlubetapd (fbetapd), 46
- nlubetapdcon, 48, 50
- nlubetapdcon (fbetapdcon), 49
- nlumgammagpd, 58, 62, 126, 128–130, 136, 163, 167, 231, 235, 239
- nlumgammagpd (fmgammagpd), 126
- nlumgammagpdcon, 58, 62, 126, 131, 134, 135, 163, 167, 231, 235, 239
- nlumgammagpdcon (fmgammagpdcon), 132
- nlugammagpd, 57, 62, 126, 131, 136, 163, 167, 231, 235, 239
- nlugammagpd (fgammagpd), 55
- nlugammagpdcon, 58, 61, 126, 131, 136, 163, 167, 231, 235, 239
- nlugammagpdcon (fgammagpdcon), 59
- nlugkg, 65, 72
- nlugkg (fgkg), 63
- nlugkgcon, 67, 70
- nlugkgcon (fgkgcon), 68
- nlugng, 75, 81, 95, 142, 145, 179, 183, 199, 244, 248
- nlugng (fgng), 73
- nlugngcon, 76, 79, 95, 142, 145, 179, 183, 199, 244, 248
- nlugngcon (fgngcon), 78
- nluhpdcon, 87, 89
- nluhpdcon (fhpdcon), 88
- nluitmgng, 76, 81, 93, 142, 146, 179, 183, 199, 244, 248
- nluitmgng (fitmgng), 92
- nluitmnormgpd, 97
- nluitmnormgpd (fitmnormgpd), 96
- nluitmweibullgpd, 100
- nluitmweibullgpd (fitmweibullgpd), 99
- nlukdengpd, 40, 45, 109, 115
- nlukdengpd (fkdgengpd), 107
- nlukdengpdcon, 40, 45, 110, 113
- nlukdengpdcon (fkdgengpdcon), 111
- nlulognormgpd, 117, 121
- nlulognormgpd (flognormgpd), 115
- nlulognormgpdcon, 118, 120
- nlulognormgpdcon (flognormgpdcon), 119
- nlumgammagpd, 58, 62, 126, 129, 130, 136, 163, 164, 167, 231, 235, 239
- nlumgammagpd (fmgammagpd), 126
- nlumgammagpdcon, 58, 62, 126, 131, 134, 135, 163, 167, 231, 235, 239
- nlumgammagpdcon (fmgammagpdcon), 132
- nlunormgpd, 77, 81, 95, 139–141, 146, 179, 183, 199, 245, 248
- nlunormgpd (fnormgpd), 137
- nlunormgpdcon, 77, 81, 95, 142, 144, 179, 183, 199, 245, 248
- nlunormgpdcon (fnormgpdcon), 143
- nlupsdengpd, 152
- nlupsdengpd (fpsdengpd), 150
- nluweibullgpd, 155, 160
- nluweibullgpd (fweibullgpd), 154
- nluweibullgpdcon, 156, 159
- nluweibullgpdcon (fweibullgpdcon), 157
- nlweibullgpd, 155, 160
- nlweibullgpd (fweibullgpd), 154
- nlweibullgpdcon, 156, 159
- nlweibullgpdcon (fweibullgpdcon), 157
- normgpd, 76, 81, 95, 142, 145, 179, 183, 199, 202, 242, 248
- normgpdcon, 76, 81, 95, 142, 145, 179, 183, 199, 244, 245
- optim, 25, 32, 33, 37, 42, 46, 50, 53, 54, 56, 60, 64, 69, 74, 78, 82, 83, 85, 86, 89, 93, 96, 99, 100, 103, 104, 108, 112, 116, 120, 123, 124, 127, 133, 138, 140, 144, 151, 155, 158
- pbckden, 6, 13, 17, 35, 106, 208, 212, 216
- pbckden (bckden), 4
- pbckdengpd, 8, 12, 17, 35, 106, 208, 212, 216
- pbckdengpd (bckdengpd), 9
- pbckdengpdcon, 8, 13, 16, 35, 106, 208, 212, 216
- pbckdengpdcon (bckdengpdcon), 14
- pbckdenxbeta1 (internal), 194
- pbckdenxbeta2 (internal), 194
- pbckdenxcopula (internal), 194
- pbckdenxcutnorm (internal), 194
- pbckdenxgamma1 (internal), 194
- pbckdenxgamma2 (internal), 194
- pbckdenxlog (internal), 194
- pbckdenxnn (internal), 194
- pbckdenxreflect (internal), 194
- pbckdenxrenorm (internal), 194
- pbckdenxsimple (internal), 194
- pbetapd, 20, 23
- pbetapd (betapd), 18
- pbetapdcon, 20, 23
- pbetapdcon (betapdcon), 21

- pdwm, 27
- pdwm (dwm), 26
- pgammagpd, 58, 62, 125, 130, 136, 162, 167, 231, 235, 239
- pgammagpd (gammagpd), 161
- pgammagpdcon, 58, 62, 125, 130, 136, 163, 166, 231, 235, 239
- pgammagpdcon (gammagpdcon), 164
- pgkg, 170, 175
- pgkg (gkg), 168
- pgkgcon, 171, 174
- pgkgcon (gkgcon), 172
- pgng, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- pgng (gng), 176
- pgngcon, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- pgngcon (gngcon), 180
- pgpd, 85, 185
- pgpd (gpd), 184
- phpd, 190, 194
- phpd (hpd), 189
- phpdcon, 191, 193
- phpdcon (hpdcon), 192
- pickands, 251
- pickandsplot, 249, 250
- pitmgng, 76, 81, 95, 142, 145, 178, 182, 198, 244, 248
- pitmgng (itmgng), 196
- pitmnormgpd, 201
- pitmnormgpd (itmnormgpd), 200
- pitmweibullgpd, 204
- pitmweibullgpd (itmweibullgpd), 203
- pkden, 8, 13, 17, 35, 106, 207, 212, 216
- pkden (kden), 206
- pkdengpd, 8, 13, 17, 35, 106, 208, 211, 216
- pkdengpd (kdengpd), 210
- pkdengpdcon, 8, 13, 17, 35, 106, 208, 212, 215
- pkdengpdcon (kdengpdcon), 213
- pkdenx (internal), 194
- plognormgpd, 224, 228
- plognormgpd (lognormgpd), 223
- plognormgpdcon, 225, 227
- plognormgpdcon (lognormgpdcon), 226
- plot, 29
- plot.uvevd, 30, 31
- pmgamma, 58, 62, 126, 131, 136, 163, 167, 230, 235, 239
- pmgamma (mgamma), 229
- pmgammagpd, 58, 62, 126, 130, 136, 163, 167, 231, 234, 239
- pmgammagpd (mgammagpd), 232
- pmgammagpdcon, 58, 62, 125, 130, 136, 163, 167, 231, 235, 238
- pmgammagpdcon (mgammagpdcon), 236
- pnorm, 221
- pnormgpd, 76, 81, 95, 142, 145, 179, 183, 199, 243, 248
- pnormgpd (normgpd), 242
- pnormgpdcon, 76, 81, 95, 142, 145, 179, 183, 199, 244, 247
- pnormgpdcon (normgpdcon), 245
- pplot, 30
- pplot (evmix.diag), 29
- ppoints, 29–31
- ppsden, 149, 253
- ppsden (psden), 251
- ppsdengpd, 256
- ppsdengpd (psdengpd), 254
- profluitmgng, 76, 81, 142, 146, 179, 183, 199, 244, 248
- profluitmgng (fitmgng), 92
- profluitmnormgpd (fitmnormgpd), 96
- profluitmweibullgpd (fitmweibullgpd), 99
- proflubckdengpd, 38, 45, 110, 115
- proflubckdengpd (fbckdengpd), 36
- proflubckdengpdcon, 40, 43, 110, 115
- proflubckdengpdcon (fbckdengpdcon), 41
- proflubetagpd, 47, 52
- proflubetagpd (fbetagpd), 46
- proflubetagpdcon, 48, 50
- proflubetagpdcon (fbetagpdcon), 49
- proflugammagpd, 57, 62, 126, 131, 136, 163, 167, 231, 235, 239
- proflugammagpd (fgammagpd), 55
- proflugammagpdcon, 58, 61, 126, 131, 136, 163, 167, 231, 235, 239
- proflugammagpdcon (fgammagpdcon), 59
- proflugkg, 65, 72
- proflugkg (fgkg), 63
- proflugkgcon, 67, 70
- proflugkgcon (fgkgcon), 68
- proflugng, 74, 75, 81, 95, 142, 146, 179, 183, 199, 244, 248
- proflugng (fgng), 73
- proflugngcon, 76, 79, 95, 142, 145, 179, 183, 199, 244, 248
- proflugngcon (fgngcon), 78
- profluhpdcon, 87, 89
- profluhpdcon (fhpdcon), 88
- profluitmgng, 76, 81, 93, 142, 146, 179, 183, 199, 244, 248
- profluitmgng (fitmgng), 92

- profluitmnormgpd, 97
- profluitmnormgpd (fitmnormgpd), 96
- profluitmweibullgpd, 100
- profluitmweibullgpd (fitmweibullgpd), 99
- proflukdengpd, 40, 45, 109, 115
- proflukdengpd (fkdengpd), 107
- proflukdengpdcon, 40, 45, 110, 113
- proflukdengpdcon (fkdengpdcon), 111
- proflulognormgpd, 117, 121
- proflulognormgpd (flognormgpd), 115
- proflulognormgpdcon, 118, 120
- proflulognormgpdcon (flognormgpdcon), 119
- proflumgammagpd, 58, 62, 126, 128, 129, 136, 164, 167, 231, 235, 239
- proflumgammagpd (fmgammagpd), 126
- proflumgammagpdcon, 58, 62, 126, 131, 134, 163, 167, 231, 235, 239
- proflumgammagpdcon (fmgammagpdcon), 132
- proflunormgpd, 77, 81, 95, 139, 140, 146, 179, 183, 199, 245, 248
- proflunormgpd (fnormgpd), 137
- proflunormgpdcon, 77, 81, 95, 142, 144, 179, 183, 199, 245, 248
- proflunormgpdcon (fnormgpdcon), 143
- proflupsdengpd, 152
- proflupsdengpd (fpsdengpd), 150
- profluweibullgpd, 155, 160
- profluweibullgpd (fweibullgpd), 154
- profluweibullgpdcon, 156, 159
- profluweibullgpdcon (fweibullgpdcon), 157
- pscounts (internal), 194
- psden, 149, 251, 253, 256, 257
- psdengpd, 254, 256
- pweibullgpd, 261, 265
- pweibullgpd (weibullgpd), 260
- pweibullgpdcon, 262, 264
- pweibullgpdcon (weibullgpdcon), 263
- pxb (internal), 194
- qbckden, 6, 13, 17, 35, 106, 208, 212, 216
- qbckden (bckden), 4
- qbckdengpd, 8, 12, 17, 35, 106, 208, 212, 216
- qbckdengpd (bckdengpd), 9
- qbckdengpdcon, 8, 13, 16, 35, 106, 208, 212, 216
- qbckdengpdcon (bckdengpdcon), 14
- qbetagpd, 20, 23
- qbetagpd (betagpd), 18
- qbetagpdcon, 20, 23
- qbetagpdcon (betagpdcon), 21
- qdwm, 27
- qdwm (dwm), 26
- qgammagpd, 58, 62, 125, 130, 136, 162, 167, 231, 235, 239
- qgammagpd (gammagpd), 161
- qgammagpdcon, 58, 62, 125, 130, 136, 163, 166, 231, 235, 239
- qgammagpdcon (gammagpdcon), 164
- qgbgmix, 198
- qgbgmix (internal), 194
- qgbgmixprime (internal), 194
- qgkg, 170, 175
- qgkg (gkg), 168
- qgkgcon, 171, 174
- qgkgcon (gkgcon), 172
- qng, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- qng (gng), 176
- qngcon, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- qngcon (gngcon), 180
- qgpd, 85, 185
- qgpd (gpd), 184
- qhpd, 190, 194
- qhpd (hpd), 189
- qhpdcon, 191, 193
- qhpdcon (hpdcon), 192
- qitmng, 76, 81, 95, 142, 145, 178, 179, 182, 183, 198, 244, 248
- qitmng (itmng), 196
- qitmnormgpd, 201
- qitmnormgpd (itmnormgpd), 200
- qitmweibullgpd, 204
- qitmweibullgpd (itmweibullgpd), 203
- qkde, 6, 207, 252
- qkden, 8, 13, 17, 18, 35, 106, 207, 212, 216
- qkden (kden), 206
- qkdengpd, 8, 13, 17, 35, 106, 208, 211, 216
- qkdengpd (kdengpd), 210
- qkdengpdcon, 8, 13, 17, 35, 106, 208, 212, 215
- qkdengpdcon (kdengpdcon), 213
- qlognormgpd, 224, 228
- qlognormgpd (lognormgpd), 223
- qlognormgpdcon, 225, 227
- qlognormgpdcon (lognormgpdcon), 226
- qmgamma, 58, 62, 126, 131, 136, 163, 167, 230, 235, 239
- qmgamma (mgamma), 229
- qmgammagpd, 58, 62, 126, 130, 131, 136, 163, 167, 231, 234, 239
- qmgammagpd (mgammagpd), 232
- qmgammagpdcon, 58, 62, 125, 130, 136, 163, 167, 231, 235, 238

- qmgammagpdcon (mgammagpdcon), 236
- qmix, 198, 201, 204
- qmix (internal), 194
- qmixprime (internal), 194
- qmixxprime, 198, 201, 204
- qnormgpd, 76, 81, 95, 142, 145, 179, 183, 199, 243, 248
- qnormgpd (normgpd), 242
- qnormgpdcon, 76, 81, 95, 142, 145, 179, 183, 199, 244, 247
- qnormgpdcon (normgpdcon), 245
- qplot, 30
- qplot (evmix.diag), 29
- qpsden, 149, 252, 253
- qpsden (psden), 251
- qpsdengpd, 256
- qpsdengpd (psdengpd), 254
- qweibullgpd, 261, 265
- qweibullgpd (weibullgpd), 260
- qweibullgpdcon, 262, 264
- qweibullgpdcon (weibullgpdcon), 263
- rbckden, 6, 7, 13, 17, 35, 106, 208, 212, 216
- rbckden (bckden), 4
- rbckdengpd, 8, 12, 17, 35, 106, 208, 212, 216
- rbckdengpd (bckdengpd), 9
- rbckdengpdcon, 8, 13, 16, 17, 35, 106, 208, 212, 216
- rbckdengpdcon (bckdengpdcon), 14
- rbetagpd, 20, 23
- rbetagpd (betagpd), 18
- rbetagpdcon, 20, 23
- rbetagpdcon (betagpdcon), 21
- rdwm, 27
- rdwm (dwm), 26
- rgammagpd, 58, 62, 125, 130, 136, 162, 167, 231, 235, 239
- rgammagpd (gammagpd), 161
- rgammagpdcon, 58, 62, 125, 130, 136, 163, 166, 231, 235, 239
- rgammagpdcon (gammagpdcon), 164
- rgkg, 170, 175
- rgkg (gkg), 168
- rgkgcon, 171, 174, 175
- rgkgcon (gkgcon), 172
- rgng, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- rgng (gng), 176
- rgngcon, 76, 81, 95, 142, 145, 178, 182, 199, 244, 248
- rgngcon (gngcon), 180
- rgpd, 85, 185
- rgpd (gpd), 184
- rhpdp, 190, 194
- rhpdp (hpd), 189
- rhpdpcon, 191, 193
- rhpdpcon (hpdcon), 192
- ritmgng, 76, 81, 95, 142, 145, 179, 183, 198, 244, 248
- ritmgng (itmng), 196
- ritmnormgpd, 201, 202
- ritmnormgpd (itmnormgpd), 200
- ritmweibullgpd, 204, 205
- ritmweibullgpd (itmweibullgpd), 203
- rkden, 8, 9, 13, 18, 35, 106, 207, 208, 212, 216
- rkden (kden), 206
- rkdengpd, 8, 13, 17, 35, 106, 208, 211, 216
- rkdengpd (kdengpd), 210
- rkdengpdcon, 8, 13, 17, 35, 106, 208, 212, 215
- rkdengpdcon (kdengpdcon), 213
- rlognormgpd, 224, 225, 228
- rlognormgpd (lognormgpd), 223
- rlognormgpdcon, 225, 227, 228
- rlognormgpdcon (lognormgpdcon), 226
- rlplot, 30
- rlplot (evmix.diag), 29
- rmgamma, 58, 62, 126, 131, 136, 163, 167, 230, 235, 239
- rmgamma (mgamma), 229
- rmgammagpd, 58, 62, 126, 131, 136, 163, 167, 231, 234, 239
- rmgammagpd (mgammagpd), 232
- rmgammagpdcon, 58, 62, 125, 130, 136, 163, 167, 231, 235, 238
- rmgammagpdcon (mgammagpdcon), 236
- rnormgpd, 76, 81, 95, 142, 145, 179, 183, 199, 243, 244, 248
- rnormgpd (normgpd), 242
- rnormgpdcon, 76, 81, 95, 142, 145, 179, 183, 199, 244, 247
- rnormgpdcon (normgpdcon), 245
- rpsden, 149, 253
- rpsden (psden), 251
- rpsdengpd, 256
- rpsdengpd (psdengpd), 254
- rweibullgpd, 261, 265
- rweibullgpd (weibullgpd), 260
- rweibullgpdcon, 262, 264
- rweibullgpdcon (weibullgpdcon), 263
- sapply, 139, 218
- splineDesign, 148, 152, 252, 253
- splinefun, 6, 207, 252
- tcplot, 257, 259
- tscaleplot, 259

tscaleplot (tcplot), [257](#)

tshapeplot, [259](#)

tshapeplot (tcplot), [257](#)

weibullgpd, [205](#), [260](#), [265](#)

weibullgpdcon, [262](#), [263](#)