Package: nieve (via r-universe)

October 16, 2024

Type Package

Title Miscellaneous Utilities for Extreme Value Analysis

Version 0.1.5

Maintainer Yves Deville <deville.yves@alpestat.com>

Description Provides utility functions and objects for Extreme Value Analysis. These include probability functions with their exact derivatives w.r.t. the parameters that can be used for estimation and inference, even with censored observations. The transformations exchanging the two parameterizations of Peaks Over Threshold (POT) models: Poisson-GP and Point-Process are also provided with their derivatives.

License GPL (>= 2)

Suggests testthat, numDeriv, Renext, knitr, covr

Encoding UTF-8

URL https://github.com/yvesdeville/nieve/

BugReports https://github.com/yvesdeville/nieve/issues/

RoxygenNote 7.2.3

VignetteBuilder knitr

Repository https://yvesdeville.r-universe.dev

RemoteUrl https://github.com/yvesdeville/nieve

RemoteRef HEAD

RemoteSha 93287a258482b1476976ce5196125a87599f8063

Contents

`nieve-package` *Miscellaneous Utilities for Extreme Value Analysis*

Description

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

The **nieve** package provides utility functions for Extreme Value Analysis. It includes the probability functions for the two-parameter Generalized Pareto Distribution (GPD) and for the three-parameter Generalized Extreme Value (GEV) distribution. These functions are vectorized w.r.t. the parameters and optionally provide the exact derivatives w.r.t. the parameters: gradient and Hessian which can be used in optimization e.g., to maximize the log-likelihood. Since the gradient and the Hessian are available for the log-density *and for the distribution function*, the exact gradient and the exact Hessian of the log-likelihood function is available even when censored observations are used.

These functions should behave like the probability functions of the **stats** package. For instance, when a probability `p = 0.0` or `p = 1.0` is given, the quantile functions should return the lower and the upper end-point, be they finite or not. Also when evaluated at `-Inf` and `Inf` the probability functions should return `0.0` and `1.0`. Mind however that the gradient and the Hessian of the upper-end point are not to be trusted for now.

The **nieve** package was partly funded by the French *Institut de Radioprotection et Sûreté Nucléaire (IRSN)* and some of the code formerly was part of R packages owned by the IRSN *Bureau d'Expertise en Hydrogéologie et sur les Risques d'Inondation, météorologiques et Géotechniques* (Behrig).

`Exp1` *Density, Distribution Function, Quantile Function and Random Generation for the One-Parameter Exponential Distribution*

Description

Density, distribution function, quantile function and random generation for the one-parameter Exponential Distribution distribution with scale parameter `scale`.

Usage

```
dexp1(x, scale = 1, log = FALSE, deriv = FALSE, hessian = FALSE)

pexp1(q, scale = 1, lower.tail = TRUE, deriv = FALSE, hessian = FALSE)

qexp1(p, scale = 1, lower.tail = TRUE, deriv = FALSE, hessian = FALSE)

rexp1(n, scale = 1, array)
```

Arguments

`x, q`	Vector of quantiles.
`scale`	Scale parameter. Numeric vector with suitable length, see **Details**.
`log`	Logical; if `TRUE`, densities `p` are returned as `log(p)`.
`deriv`	Logical. If `TRUE`, the gradient of each computed value w.r.t. the parameter vector is computed, and returned as a `"gradient"` attribute of the result. This is a numeric array with dimension `c(n, 1)` where `n` is the length of the first argument, i.e. `x`, `p` or `q`, depending on the function.
`hessian`	Logical. If `TRUE`, the Hessian of each computed value w.r.t. the parameter vector is computed, and returned as a `"hessian"` attribute of the result. This is a numeric array with dimension `c(n, 1, 1)` where `n` is the length of the first argument, i.e. `x`, `p` or depending on the function.
`lower.tail`	Logical; if `TRUE` (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.
`p`	Vector of probabilities.
`n`	Sample size.
`array`	Logical. If `TRUE`, the simulated values form a numeric matrix with `n` columns and `np` rows where `np` is the number of exponential parameter values i.e., the length of `scale`. This option is useful to cope with so-called *non-stationary* models with exponential margins. See **Examples**. The default value is `length(scale) > 1`.

Details

The survival and density functions are given by

$$S(x) = \exp\{-x/\sigma\} \qquad f(x) = \frac{1}{\sigma}\exp\{-x/\sigma\} \qquad (x > 0)$$

where σ is the scale parameter. This distribution is the Generalized Pareto Distribution for a shape $\xi = 0$.

The probability functions `d`, `p` and `q` all allow the parameter `scale` to be a vector. Then the recycling rule is used to get two vectors of the same length, corresponding to the first argument and to the scale parameter. This behaviour is the standard one for the probability functions of the **stats** package but is unusual in R packages devoted to Extreme Value in which the parameters must generally have length one. Note that the provided functions can be used e.g. to evaluate the quantile with a given probability for a large number of values of the parameter vector `shape`. This is frequently required in he Bayesian framework with MCMC inference.

Value

A numeric vector with its length equal to the maximum of the two lengths: that of the first argument and that of the parameter `scale`. When `deriv` is `TRUE`, the returned value has an attribute named `"gradient"` which is a matrix with n lines and 1 column containing the derivative. A row contains the partial derivative of the corresponding element w.r.t. the parameter `"scale"`.

Note

The attributes "gradient" and "hessian" have dimension `c(n, 1)` and `c(n, 1, 1)`, even when `n` equals 1. Use the `drop` method on these objects to drop the extra dimension if wanted i.e. to get a gradient vector and a Hessian matrix.

See Also

The exponential distribution `Exponential` with $rate$ being the inverse scale.

Examples

```
## Illustrate the effect of recycling rule.
pexp1(1.0, scale = 1:4, lower.tail = FALSE) - exp(-1.0 / (1:4))
pexp1(1:4, scale = 1:4, lower.tail = FALSE) - exp(-1.0)

## With gradient and Hessian.
pexp1(c(1.1, 1.7), scale = 1, deriv = TRUE, hessian = TRUE)

ti <- 1:60; names(ti) <- 2000 + ti
sigma <- 1.0 + 0.7 * ti
## simulate 40 paths
y <- rexp1(n = 40, scale = sigma)
matplot(ti, y, type = "l", col = "gray", main = "varying scale")
lines(ti, apply(y, 1, mean))
```

GEV — *Density, Distribution Function, Quantile Function and Random Generation for the Generalized Extreme Value (GEV) Distribution*

Description

Density, distribution function, quantile function and random generation for the Generalized Extreme Value (GEV) distribution with parameters `loc`, `scale` and `shape`. The distribution function $F(x) = \Pr[X \leq x]$ is given by

$$F(x) = \exp\left\{-[1 + \xi z]^{-1/\xi}\right\}$$

when $\xi \neq 0$ and $1 + \xi z > 0$, and by

$$F(x) = \exp\left\{-e^{-z}\right\}$$

for $\xi = 0$ where $z := (x - \mu)/\sigma$ in both cases.

Usage

```
dGEV(
  x,
  loc = 0,
  scale = 1,
  shape = 0,
  log = FALSE,
  deriv = FALSE,
  hessian = FALSE
)

pGEV(
  q,
  loc = 0,
  scale = 1,
  shape = 0,
  lower.tail = TRUE,
  deriv = FALSE,
  hessian = FALSE
)

qGEV(
  p,
  loc = 0,
  scale = 1,
  shape = 0,
  lower.tail = TRUE,
  deriv = FALSE,
  hessian = FALSE
)

rGEV(n, loc = 0, scale = 1, shape = 0, array)
```

Arguments

`x, q` Vector of quantiles.

`loc` Location parameter. Numeric vector with suitable length, see **Details**.

`scale` Scale parameter. Numeric vector with suitable length, see **Details**.

`shape` Shape parameter. Numeric vector with suitable length, see **Details**.

`log` Logical; if `TRUE`, densities `p` are returned as `log(p)`.

`deriv` Logical. If `TRUE`, the gradient of each computed value w.r.t. the parameter vector is computed, and returned as a `"gradient"` attribute of the result. This is a numeric array with dimension `c(n, 3)` where `n` is the length of the first argument, i.e. `x`, `p` or `q` depending on the function.

`hessian` Logical. If `TRUE`, the Hessian of each computed value w.r.t. the parameter vector is computed, and returned as a `"hessian"` attribute of the result. This is

	a numeric array with dimension `c(n, 3, 3)` where `n` is the length of the first argument, i.e. `x`, `p` or depending on the function.
`lower.tail`	Logical; if `TRUE` (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.
`p`	Vector of probabilities.
`n`	Sample size.
`array`	Logical. If `TRUE`, the simulated values form a numeric matrix with `n` columns and `np` rows where `np` is the number of GEV parameter values. This number is obtained by recycling the three GEV parameters vectors to a common length, so `np` is the maximum of the lengths of the parameter vectors `loc`, `scale`, `shape`. This option is useful to cope with so-called *non-stationary* models with GEV margins. See **Examples**. The default value is `TRUE` if any of the vectors `loc`, `scale` and `shape` has length > 1 and `FALSE` otherwise.

Details

Each of the probability function normally requires two formulas: one for the non-zero shape case $\xi \neq 0$ and one for the zero-shape case $\xi = 0$. However the non-zero shape formulas lead to numerical instabilities near $\xi = 0$, especially for the derivatives w.r.t. ξ. This can create problem in optimization tasks. To avoid this, a Taylor expansion w.r.t. ξ is used for $|\xi| < \epsilon$ for a small positive ϵ. The expansion has order 2 for the functions (log-density, distribution and quantile), order 1 for their first-order derivatives and order 0 for the second-order derivatives.

For the `d`, `p` and `q` functions, the GEV parameter arguments `loc`, `scale` and `shape` are recycled in the same fashion as the classical R distribution functions in the **stats** package, see e.g., `Normal`, `GammaDist`, ... Let `n` be the maximum length of the four arguments: `x` `q` or `p` and the GEV parameter arguments, then the four provided vectors are recycled in order to have length `n`. The returned vector has length `n` and the attributes `"gradient"` and `"hessian"`, when computed, are arrays wich dimension: `c(1, 3)` and `c(1, 3, 3)`.

Value

A numeric vector with length `n` as described in the **Details** section. When `deriv` is `TRUE`, the returned value has an attribute named `"gradient"` which is a matrix with n lines and 3 columns containing the derivatives. A row contains the partial derivatives of the corresponding element w.r.t. the three parameters `loc` `scale` and `shape` in that order.

Examples

```
ti <- 1:10; names(ti) <- 2000 + ti
mu <- 1.0 + 0.1 * ti
## simulate 40 paths
y <- rGEV(n = 40, loc = mu, scale = 1, shape = 0.05)
matplot(ti, y, type = "l", col = "gray")
lines(ti, apply(y, 1, mean))
```

GPD2 *Density, Distribution Function, Quantile Function and Random Generation for the Two-Parameter Generalized Pareto Distribution (GPD)*

Description

Density, distribution function, quantile function and random generation for the two-parameter Generalized Pareto Distribution (GPD) distribution with `scale` and `shape`.

Usage

```
dGPD2(x, scale = 1, shape = 0, log = FALSE, deriv = FALSE, hessian = FALSE)

pGPD2(
  q,
  scale = 1,
  shape = 0,
  lower.tail = TRUE,
  deriv = FALSE,
  hessian = FALSE
)

qGPD2(
  p,
  scale = 1,
  shape = 0,
  lower.tail = TRUE,
  deriv = FALSE,
  hessian = FALSE
)

rGPD2(n, scale = 1, shape = 0, array)
```

Arguments

`x, q`	Vector of quantiles.
`scale`	Scale parameter. Numeric vector with suitable length, see **Details**.
`shape`	Shape parameter. Numeric vector with suitable length, see **Details**.
`log`	Logical; if `TRUE`, densities `p` are returned as `log(p)`.
`deriv`	Logical. If `TRUE`, the gradient of each computed value w.r.t. the parameter vector is computed, and returned as a `"gradient"` attribute of the result. This is a numeric array with dimension `c(n, 2)` where `n` is the length of the first argument, i.e. `x`, `p` or `q`, depending on the function.
`hessian`	Logical. If `TRUE`, the Hessian of each computed value w.r.t. the parameter vector is computed, and returned as a `"hessian"` attribute of the result. This is

	a numeric array with dimension `c(n, 2, 2)` where `n` is the length of the first argument, i.e. `x`, `p` or depending on the function.
`lower.tail`	Logical; if `TRUE` (default), probabilities are $\Pr[X \leq x]$, otherwise, $\Pr[X > x]$.
`p`	Vector of probabilities.
`n`	Sample size.
`array`	Logical. If `TRUE`, the simulated values form a numeric matrix with `n` columns and `np` rows where `np` is the number of GPD parameter values. This number is obtained by recycling the two GPD parameters vectors to a common length, so `np` is the maximum of the lengths of the parameter vectors `scale` and `shape`. This option is useful to cope with so-called *non-stationary* models with GPD margins. See **Examples**. The default value is `TRUE` if any of the vectors `scale` and `shape` has length `> 1` and `FALSE` otherwise.

Details

Let $\sigma > 0$ and ξ denote the scale and the shape; the survival function $S(x) := \Pr[X > x]$ is given for $x \geq 0$ by

$$S(x) = [1 + \xi x/\sigma]_+^{-1/\xi}$$

for $\xi \neq 0$ where $v_+ := \max\{v,\, 0\}$ and by

$$S(x) = \exp\{-x/\sigma\}$$

for $\xi = 0$. For $x < 0$ we have $S(x) = 1$: the support of the distribution is $(0,\, \infty($.

The probability functions `d`, `p` and `q` all allow each of the two GP parameters to be a vector. Then the recycling rule is used to get three vectors of the same length, corresponding to the first argument and to the two GP parameters. This behaviour is the standard one for the probability functions of the **stats**. Note that the provided functions can be used e.g. to evaluate the quantile with a given probability for a large number of values of the parameter vector `c(shape, scale)`. This is frequently required in he Bayesian framework with MCMC inference.

Value

A numeric vector with length equal to the maximum of the four lengths: that of the first argument and that of the two parameters `scale` and `shape`. When `deriv` is `TRUE`, the returned value has an attribute named `"gradient"` which is a matrix with n lines and 2 columns containing the derivatives. A row contains the partial derivatives of the corresponding element w.r.t. the two parameters `"scale"` and `"shape"` in that order.

Note

The attributes `"gradient"` and `"hessian"` have dimension `c(n, 2)` and `c(n, 2, 2)`, even when `n` equals `1`. Use the `drop` method on these objects to drop the extra dimension if wanted i.e. to get a gradient vector and a Hessian matrix.

Examples

```
## Illustrate the effect of recycling rule.
pGPD2(1.0, scale = 1:4, shape = 0.0, lower.tail = FALSE) - exp(-1.0 / (1:4))
pGPD2(1:4, scale = 1:4, shape = 0.0, lower.tail = FALSE) - exp(-1.0)

## With gradient and Hessian.
pGPD2(c(1.1, 1.7), scale = 1, shape = 0, deriv = TRUE, hessian = TRUE)

## simulate 40 paths
ti <- 1:20
names(ti) <- 2000 + ti
y <- rGPD2(n = 40, scale = ti, shape = 0.05)
matplot(ti, y, type = "l", col = "gray", main = "varying scale")
lines(ti, apply(y, 1, mean))
```

---

poisGP2PP *Transform Poisson-GP Parameters into Point-Process Parameters*

Description

Transform Poisson-GP parameters into Point-Process (PP) parameters. In the POT Poisson-GP framework the three parameters are the rate `lambda` λ_u of the Poisson process in time and the two GP parameters: `scale` σ_u and `shape` ξ. The vector `loc` contains the fixed threshold u, and `w` the fixed block duration. These parameters are converted into the vector of three parameters of the GEV distribution for the maximum of the marks Y_i on a time interval with duration `w`, the number N of these marks being a r.v. with Poisson distribution. More precisely, the GEV distribution applies when $N > 0$.

Usage

```
poisGP2PP(lambda, loc = 0.0, scale = 1.0, shape = 0.0, w =
    1.0, deriv = FALSE)
```

Arguments

`lambda`	A numeric vector containing the Poisson rate(s).
`loc`	A numeric vector containing the Generalized Pareto location, i.e. the threshold in the POT framework.
`scale, shape`	Numeric vectors containing the Generalized Pareto scale and shape parameters.
`w`	The block duration. Its physical dimension is time and the product $\lambda_u \times w$ is dimensionless.
`deriv`	Logical. If `TRUE` the derivative (Jacobian) of the transformation is computed and returned as an attribute named `"gradient"` of the attribute.

Details

The three PP parameters $\mu^\star_w$, $\sigma^\star_w$ and $\xi^\star$ relate to the Poisson-GP parameters according to

$$\begin{cases} \mu^\star_w &= u + \frac{(\lambda_u w)^\xi - 1}{\xi}\,\sigma_u, \\ \sigma^\star_w &= (\lambda_u w)^\xi\,\sigma_u, \\ \xi^\star &= \xi, \end{cases}$$

the fraction $[(\lambda_u w)^\xi - 1]/\xi$ of the first equation being to be replaced for $\xi = 0$ by its limit $\log(\lambda_u w)$.

Value

A numeric matrix with three columns representing the Point-Process parameters `loc` $\mu^\star_w$, `scale` $\sigma^\star_w$ and `shape` $\xi^\star$.

Note

This function is essentially a re-implementation in C of the function `Ren2gev` of **Renext**. As a major improvement, this function is "vectorized" w.r.t. the parameters so it can transform efficiently a large number of Poisson-GP parameter vectors as can be required e.g. in a MCMC Bayesian inference. Note also that this function copes with values near zero for the shape parameter: it suitably computes then both the function value and its derivatives.

See Also

`PP2poisGP` for the reciprocal transformation.

`PP2poisGP` *Transform Point-Process Parameters into Poisson-GP Parameters*

Description

Transform Point Process (PP) parameters into Poisson-GP parameters. The provided parameters are GEV parameters: location $\mu^\star$, scale $\sigma^\star_w$ and shape $\xi^\star$. They are assumed to describe (the tail of) the distribution for a maximum on a time-interval with given duration w. For a given threshold u chosen to be in the interior of the support of the GEV distribution, there exists a unique vector of three Poisson-GP parameters such that the maximum M of the marks on an interval with duration `w` has the prescribed GEV tail. Remind that the three Poisson-GP parameters are the rate of the Poisson process in time: λ_u, and the two GP parameters: `scale` σ_u and `shape` ξ. The shape parameters $\xi^\star$ and ξ are identical.

Usage

```
PP2poisGP(locStar = 0.0, scaleStar = 1.0, shapeStar = 0.0,
          threshold,
          w = 1.0, deriv = FALSE)
```

Arguments

`locStar`, `scaleStar`, `shapeStar`
: Numeric vectors containing the GEV location, scale and shape parameters.

`threshold`
: Numeric vector containing the thresholds of the Poisson-GP model, i.e. the location of the Generalised Pareto Distribution. *The threshold must be an interior point of the support of the corresponding GEV distribution*.

`w`
: The block duration. Its physical dimension is time and the product $\lambda \times w$ is dimensionless.

`deriv`
: Logical. If `TRUE` the derivative (Jacobian) of the transformation is computed and returned as an attribute named `"gradient"` of the attribute.

Details

The Poisson-GP parameters are obtained by

$$\left\{ \begin{array}{rcl} \sigma_u &=& \sigma_w^\star + \xi^\star \left[u - \mu_w^\star\right], \\ \lambda_u &=& w^{-1} \left[\sigma_u / \sigma_w^\star\right]^{-1/\xi^\star}, \\ \xi &=& \xi^\star, \end{array} \right.$$

the second equation becomes $\lambda_u = w^{-1}$ for $\xi^\star = 0$.

Value

A matrix with three columns representing the Poisson-GP parameters `lambda`, `scale` and `shape`.

Note

This function is essentially a re-implementation in C of the function `gev2Ren` of **Renext**. As a major improvement, this function is "vectorized" w.r.t. the parameters so it can transform efficiently a large number of PP parameter vectors as it can be required e.g. in a MCMC Bayesian inference. Note also that this function copes with values near zero for the shape parameter: it suitably computes then both the function value and its derivatives.

See Also

`poisGP2PP` for the reciprocal transformation.

Index