

The yaev package: Yet Another Extreme Value package?

Yves Deville deville.yves@alpestat.com

September 21, 2022

Contents

1	Probability functions of Extreme-Value distributions	1
2	Deriving the formulas	4
3	Testing the derivatives	4
4	Parameterisations for Peaks Over Threshold models	5
5	EV distributions from other R packages	7

1 Probability functions of Extreme-Value distributions

The probability functions for the Generalised Pareto (GP) and Generalised Extreme Value (GEV) distributions Coles (2001) depend smoothly on the parameters: they are infinitely differentiable functions of the parameters. However, these functions are not *analytic* functions of the parameters and a singularity exists for all of them when the shape parameter, say ξ , is zero. In practice, the functions are given with different formulas depending on whether ξ is zero or not; the formulas for $\xi = 0$ relate to the exponential and Gumbel distributions and correspond to the limit for $\xi \rightarrow 0$ of the functions given by the formulas for $\xi \neq 0$. As an example, consider the quantile function of the Generalised Pareto distribution with shape ξ and unit scale

$$q(p) = \begin{cases} [(1-p)^{-\xi} - 1]/\xi & \xi \neq 0 \\ -\log(1-p) & \xi = 0, \end{cases} \quad 0 < p < 1. \quad (1)$$

It can be shown that for $\xi \approx 0$ whatever be p

$$q \approx -\log(1-p), \quad \frac{\partial q}{\partial \xi} \approx \frac{1}{2} \log^2(1-p), \quad \frac{\partial^2 q}{\partial \xi^2} \approx -\frac{1}{3} \log^2(1-p).$$

It is quite easy to obtain expressions for the derivatives w.r.t. ξ using the definition (1). We can even rely on the symbolic differentiation method `D` available in R which, as opposed to me and many other humans, never makes any mistake when differentiating.

```

qEx <- function(p, xi) ((1 - p)^(-xi) - 1) / xi
dqEx <- D(expression(((1 - p)^(-xi) - 1) / xi), name = "xi")
d2qEx <- D(dqEx, name = "xi")
p <- 0.99; pBar <- 1 - p
xis <- c(1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9)
for (xi in xis) {
  r <- rbind("ord 0" = c("lim" = -log(pBar), "der" = qEx(p = p, xi = xi)),
            "ord 1" = c("lim" = log(pBar)^2 / 2, "der" = eval(dqEx, list(p = p, xi = xi))),
            "ord 2" = c("lim" = -log(pBar)^3 / 3, "der" = eval(d2qEx, list(x = p, xi = xi))))
  cat("xi = ", xi, "\n")
  print(r)
}

## xi = 1e-04
##      lim      der
## ord 0  4.60517  4.606231
## ord 1 10.60380 10.607052
## ord 2 32.55486 32.566137
## xi = 1e-05
##      lim      der
## ord 0  4.60517  4.605276
## ord 1 10.60380 10.604121
## ord 2 32.55486 32.771042
## xi = 1e-06
##      lim      der
## ord 0  4.60517  4.605181
## ord 1 10.60380 10.603811
## ord 2 32.55486 68.213867
## xi = 1e-07
##      lim      der
## ord 0  4.60517    4.605171
## ord 1 10.60380    10.601845
## ord 2 32.55486 39126.437500
## xi = 1e-08
##      lim      der
## ord 0  4.60517  4.605170e+00
## ord 1 10.60380  1.085129e+01
## ord 2 32.55486 -4.949912e+07
## xi = 1e-09
##      lim      der
## ord 0  4.60517  4.605170e+00
## ord 1 10.60380 -3.794649e+01
## ord 2 32.55486  9.710058e+10

```

We see that the formula for the function works fine. However, the formula for the 2-nd order derivative can be completely wrong when ξ is about $1e-6$ and the formula for the 1-st order derivative can also be wrong when ξ is about $1e-8$. Although evaluated at a reasonably small value of ξ , the derivative given in the column `der` differs much from its limit in column `lim`. The reason is that the formulas for the derivatives involve difference and/or fractions or small quantities because ξ or ξ^2 comes at the denominator. As a general rule, the derivatives with higher order are more difficult to evaluate numerically, since they involve more complex expressions. Note that using a shape ξ with $|\xi| \leq 1e-6$ is quite common in EVA because the values of ξ used in practice are often quite small, and moreover very small values of ξ are often used in the initialisation of the Maximum-Likelihood (ML) optimisation.

Let us now see what **yaev** tells about the derivatives. These can be found as attributes of the result returned by the **qGPD2** function, with names **"gradient"** and **"hessian"**. If the formal argument **p** is a vector with length **n**, the attributes are arrays with dimension $c(n, p)$ and $c(n, p, p)$ where **p** stand for the number of parameters, here **p** = 2. The arrays of derivatives have suitable **dimnames** hence can be indexed which characters as in `H[1, "scale", "shape"]` if **H** is the Hessian found in the corresponding attribute.

```
library(yaev)
for (xi in xis) {
  x <- qGPD2(p, shape = xi, deriv = TRUE, hessian = TRUE)
  r <- rbind("ord 0" = c("lim" = -log(pBar), "der" = x),
            "ord 1" = c("lim" = log(pBar)^2 / 2, "der" = attr(x, "gradient")[1, "shape"]),
            "ord 2" = c("lim" = -log(pBar)^3 / 3, "der" = attr(x, "hessian")[1, "shape", "shape"]))
  cat("xi = ", xi, "\n")
  print(r)
}

## xi = 1e-04
##          lim      der
## ord 0  4.60517  4.606231
## ord 1 10.60380 10.607052
## ord 2 32.55486 32.566137
## xi = 1e-05
##          lim      der
## ord 0  4.60517  4.605276
## ord 1 10.60380 10.604122
## ord 2 32.55486 32.554857
## xi = 1e-06
##          lim      der
## ord 0  4.60517  4.605181
## ord 1 10.60380 10.603829
## ord 2 32.55486 32.554857
## xi = 1e-07
##          lim      der
## ord 0  4.60517  4.605171
## ord 1 10.60380 10.603799
## ord 2 32.55486 32.554857
## xi = 1e-08
##          lim      der
## ord 0  4.60517  4.60517
## ord 1 10.60380 10.60380
## ord 2 32.55486 32.55486
## xi = 1e-09
##          lim      der
## ord 0  4.60517  4.60517
## ord 1 10.60380 10.60380
## ord 2 32.55486 32.55486
```

So, no more major departures from the theory can be seen.

Although not yet widespread, the use of the exact formulas for the derivatives w.r.t. the parameters can be of great help in the optimisation tasks required in EVA. These tasks of course involve the ML estimation, but also profile-likelihood inference for models with covariates. Differential equations methods can be also used to derive confidence intervals. Note that the use of formulas for the derivatives is called *symbolic* differentiation and differs from *automatic* differentiation as

increasingly available. However, from the previous analysis it transpires that the cure might be worse than the disease. Unless the derivatives are evaluated with great care, they can merely ruin an optimisation in which they are involved, instead of improving it.

Our strategy consists in fixing a small $\epsilon > 0$ and use the formulas for $\xi \neq 0$ only when $|\xi| > \epsilon$. When $|\xi| \leq \epsilon$, we use a few terms from the Taylor series at $\xi = 0$ e.g.,

$$q(p; \xi) \approx q(p; 0) + \partial_{\xi} q(p; \xi)|_{\xi=0} \times \xi + \frac{1}{2} \partial_{\xi, \xi}^2 q(p; \xi)|_{\xi=0} \times \xi^2 + o(\xi^2).$$

In order to maintain the consistency between the derivatives, it seems good to use the same ϵ for all the derivatives and use consistent Taylor approximations, so use the order 1 for the derivative $\partial_{\xi} q$ and the order order 0 for $\partial_{\xi, \xi}^2 q$ or for a crossed derivative such as $\partial_{\sigma, \xi}^2 q$. Since the 2-nd order derivatives can be required, we must take a value for ϵ which is not too small: 1e-4 or 1e-5, not much smaller. Note that ϵ gives the level of error for the 2-nd order derivative; since the error on the function is $O(\xi^3)$, using $\epsilon = 1e-4$ leads to an error of order 1e-12 on the function, which seems acceptable in practice. This kind of approximation is used in some codes of the **revdbayes** package by Paul Northrop, see the code if the **dgev** and **pgev** functions on GitHub repos.

2 Deriving the formulas

The reports provided with **yaev**: **GEV.pdf**, **GPD2.pdf**, **PoisGP2PP.pdf** and **PP2PoisGP.pdf** give the exact expressions for the first-order and the second-order derivatives of the probability functions w.r.t. the parameters and also provides workable approximations for the case $\xi \approx 0$. We used the Maxima Computer Algebra System along with the **maxiplot** package for L^AT_EX.

- The **raw expressions given by Maxima are reported in green**. The expressions can be regarded as exact, not being influenced by manual computations. However these formulas are usually difficult to use in a compiled code and require some manual transformation for this aim.
- The **simplified expressions are reported in red**. These expressions are derived by us from the raw expressions; they are influenced by manual computations hence could in principle contain errors, although they have been carefully checked. These formulas are used to write the compiled code. They often use auxiliary variables that are shared across several formulas.

3 Testing the derivatives

The **yaev** package comes with a series of tests in the format of the **testthat** package. The **numDeriv** package is used to compute the derivatives by numeric differentiation up to the order 2; these derivatives are compared to those provided by the formulas.

A quite difficult task when checking derivatives is to give a threshold used to decide if the difference between the numeric derivative and the symbolic derivative, say the “error”, is acceptable or not. This error has two sources: one is the numerical evaluation of the symbolic derivative or of its approximation for $\xi \approx 0$ (see example above), and the other is the approximation used in numeric differentiation where the limit defining the derivative is replaced by a finite difference. We should use small values of ξ with $|\xi| < \epsilon$ to check that the approximation for small ξ is correct, although

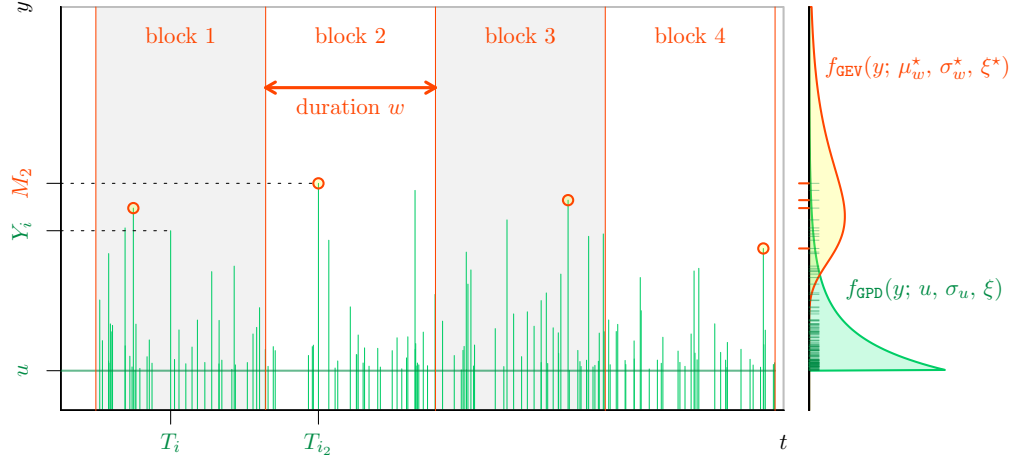


Figure 1: Block maxima by aggregation of the Poisson-GP.

we can only test the approximation at the first order by doing so. Then, with a good choice of ϵ the error should be mainly due to the numeric differentiation. But when the true derivative is small, the relative error may be large (think of a true derivative which is exactly zero). On the other hand, when a derivative is large in absolute value, the absolute error may also be quite large. Mind that the derivatives can in practice be very small or very large, and also that a gradient vector or a Hessian matrix often contain values that are not of the same order of magnitude.

We check that either the *absolute* error or the *relative* error is small. The idea is that none of these two things can come by chance, and if one holds, the formula used must be good even if the other criterion suggests an opposed conclusion. The test is made *elementwise*, meaning that the relative error is computed for each element of a gradient vector or Hessian matrix ignoring the other elements.

4 Parameterisations for Peaks Over Threshold models

Peaks Over Threshold (POT) models are very popular in EVA. These models relate to marked Poisson Process: at each time T_i in a sequence or random times $T_1 < T_2 < \dots$ we observe a random variable Y_i called the *mark*. In applications, the time T_i is typically for the occurrence a storm and the mark Y_i can be an amount of precipitation or a sea levels. The following two frameworks are commonly used, see Northrop et al. (2016) for more details.

- The *Poisson-GP* framework involves a given threshold u having the same physical dimension as the marks Y_i . It assumes that the T_i form an homogeneous Poisson Process with rate λ_u , and that Y_i are i.i.d. with a Generalised Pareto distribution $\text{GPD}(u, \sigma_u, \xi)$ or equivalently that the so-called *excesses* over the threshold $Y_i - u$ follow the two-parameter GP distribution with scale $\sigma > 0$ and shape ξ . The marks Y_i are further assumed to be independent of the event process $\{T_i\}_i$. The parameters form the vector $\theta_u = [\lambda_u, \sigma_u, \xi]$.
- An alternative framework is often named *Point Process* (PP) or *Non-Homogeneous Poisson Process* (NHPP). It involves a reference duration $w > 0$, usually chosen to be one year, and

a vector of parameters $\boldsymbol{\theta}^* := [\mu_w^*, \sigma_w^*, \xi^*]$. The random observations $[T_i, Y_i]$ are given by a Poisson process on the (t, y) -plane with intensity

$$\gamma_w^*(t, y) := \frac{1}{w} \times \frac{1}{\sigma_w^*} \left[1 + \xi^* \frac{y - \mu_w^*}{\sigma_w^*} \right]^{-1/\xi^* - 1} 1_{\mathcal{S}_{\boldsymbol{\theta}_w^*}}(t, y),$$

where $\mathcal{S}_{\boldsymbol{\theta}_w^*}$ is the domain of the plane

$$\mathcal{S}_{\boldsymbol{\theta}_w^*} := \{[t, y] : \sigma_w^* + \xi^*[y - \mu_w^*] > 0\}.$$

This is a half-plane for $\xi^* \neq 0$ and the whole plane for $\xi^* = 0$.

If we take the observation region for the PP model as being the product of the time interval $(0, t^\dagger)$ by the y -interval (u, ∞) where the threshold u is such that $\sigma_w^* + \xi^*[y - \mu_w^*] > 0$, we get the same model as the Poisson-GP on $(0, t^\dagger)$, up to a re-parameterisation. As an interesting feature of the PP parameter $\boldsymbol{\theta}_w^*$, it does not depend on the threshold u . It relates to the distribution of the maximum M of the marks Y_i corresponding to a time interval with duration w , see Figure 4. This distribution is indeed essentially the $\text{GEV}(\mu_w^*, \sigma_w^*, \xi^*)$, up to an atom corresponding to the possibility that no mark is observed during the time interval. We can then define $M := -\infty$ since this is arguably the maximum of the empty set, and the probability of the corresponding event is $\exp\{-\lambda_u w\}$. We may speak of $\text{GEV}(\mu_w^*, \sigma_w^*, \xi^*)$ as the *GEV reference distribution* in relation to w , although this is not exactly the distribution of a maximum M over the reference duration.

For a given threshold u and a given reference duration $w > 0$, the one-to-one correspondence between the vectors $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_w^*$ is given by

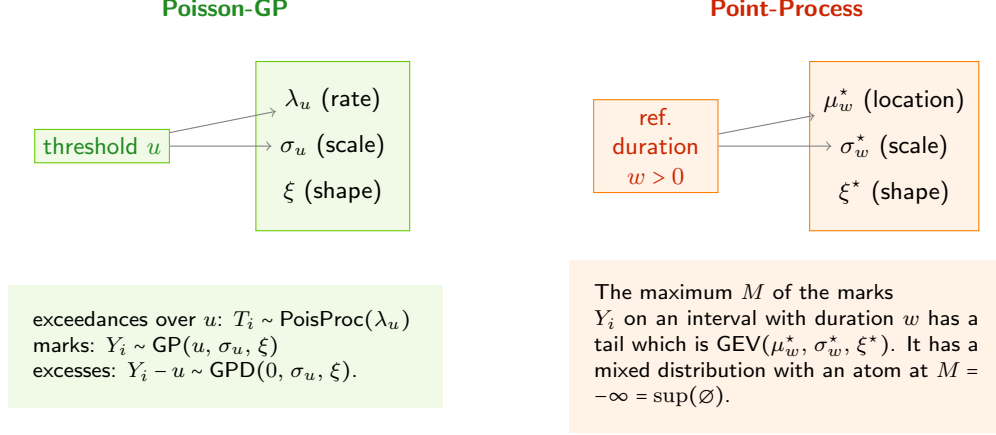
$$\begin{cases} \mu_w^* &= u + \frac{(\lambda_u w)^\xi - 1}{\xi} \sigma_u, \\ \sigma_w^* &= (\lambda_u w)^\xi \sigma_u, \\ \xi^* &= \xi, \end{cases} \quad (2)$$

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

$$\begin{cases} \sigma_u &= \sigma_w^* + \xi^*[u - \mu_w^*], \\ \lambda_u &= w^{-1} [\sigma_u / \sigma_w^*]^{-1/\xi^*}, \\ \xi &= \xi^*, \end{cases} \quad (3)$$

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

When a vector \mathbf{x} of covariates can be used, two different kinds of so-called *non-stationary* POT models can be obtained by relating or “linking” the three parameters to the covariates, either parametrically or non-parametrically. For instance the Poisson-GP scale or its logarithm can be specified as having the parametric form $\mathbf{x}^\top \boldsymbol{\beta}^\sigma$ where $\boldsymbol{\beta}^\sigma$ is a vector of parameters. Different forms of models arise from the two frameworks. Moreover, parametric Poisson-GP models relate to a specific threshold since the same form of link can not persist when the threshold is changed. Remind also that the threshold should in general depend on the covariates. Anyway, for non-stationary models it is often useful to transform one of the two parameterisations into the other. If a Poisson-GP model is used, we may be interested in the GEV reference distribution conditional on a given value



Package	GEV					GPD				
	Lang.	Vec. θ	Grad.	Hess.	$\xi = 0$	Lang.	Vec. θ	Grad.	Hess.	$\xi = 0$
evir	R	no	no	no	NT	R	no	no	no	NT
evd	R	no	no	no	0.0	R	no	no	no	0.0
ismev	R*	no	no	no	NT	R*	no	no	no	NT
Renext						R	no	yes	yes	ϵ/S
POT						R*	yes	no	no	0.0
SpatialExtremes	R	yes	no	no	0.0	R	yes	no	no	0.0
revdbayes	R	yes	no	no	ϵ/AI	R	yes	no	no	ϵ/AI
mev	R	yes	yes*	yes*	NT	R	yes	yes*	yes*	NT
yaev	C	yes	yes	yes	ϵ/AI	C	yes	yes	yes	ϵ/AI

Table 1: Features of some CRAN packages. *Lang.*: the implementation language, *Vec. θ* : vectorised w.r.t. the parameters. The columns *Grad.* and the *Hess.* indicate if the gradient and Hessian are provided, and the columns $\epsilon = 0$ indicate the strategy used to cope with a zero or small shape, as described in the text. A star \star means that the functions are not exported.

2. $[0.0]$ Test the exact equality $\xi = 0$, and if this is true, *switch to the exponential/Gumbel formula*. This helps only when the user gives `xi = 0.0`, but we are essentially doing the same thing as in $[NT]$.
3. $[\epsilon/S]$ Test the equality $|\xi| \leq \epsilon$ where $\epsilon > 0$ is very small, and if this is true, *switch to the exponential/Gumbel formula*. So this produces a (very small) discontinuity. E.g., **Renext** uses $\epsilon \approx 2e-14$.
4. $[\epsilon/AI]$ Test the equality $|\xi| \leq \epsilon$ where $\epsilon > 0$ is very small, and if this is true, use a dedicated *approximation or interpolation*. Several methods can be used including Taylor approximations. The discontinuity should then be undetectable. Mind the probability functions although not being *analytic* functions, are infinitely differentiable C^∞ w.r.t. the parameters.

Note that some of the cited packages are quite old: **evir** (Pfaff and McNeil, 2018), **evd** (Stephenson, 2002), **ismev** (Stephenson and Heffernan, 2018), **Renext** (Deville and IRSN, 2022), **POT** (Ribatet and Dutang, 2022) and **SpatialExtremes** Ribatet (2022). The packages **revdbayes** (Northrop, 2022) and **mev** (Belzile et al., 2022) are more recent. See the CRAN Task View on Extreme Value Analysis (Dutang, 2022) for an extended list of packages devoted to EVA. Also it is worth mentioning that the **extRemes** package (Gilleland and Katz, 2016) optionally uses the exact gradient of the log-likelihood for models with GEV and GP margins but the derivatives are coded (in R) only for internal use in optimisation tasks.

Acknowledgments

We are grateful to the authors and contributors of **Maxima** and to the authors of the **maxiplot** \LaTeX package (J.M. Planas and José Manuel Mira univ. de Murcia, Spain) which helped much for the tedious computations required by the package.

References

- Belzile, L., Wadworth, J. L., Northrop, P. J., Grimshaw, S. D., Jin, Z., A., S. M., Owen, A. B., and Huser, R. (2022). *mev: Modelling Extreme Values*. R package version 1.14.
- Coles, S. (2001). *An Introduction to Statistical Modelling of Extreme Values*. Springer.
- Deville, Y. and IRSN (2022). *Renext: Renewal Method for Extreme Values Extrapolation*. R package version 3.1-3.
- Dutang, C. (2022). CRAN Task View: Extreme Value Analysis. Version 2022-08-04.
- Gilleland, E. and Katz, R. W. (2016). **extRemes** 2.0: An Extreme Value Analysis Package in R. *Journal of Statistical Software*, 72(8):1–39.
- Northrop, P. J. (2022). *revdbayes: Ratio-of-Uniforms Sampling for Bayesian Extreme Value Analysis*. R package version 1.4.9.
- Northrop, P. J., Jonathan, P., and Randell, D. (2016). Threshold Modelling of Non-Stationary Extremes. In Dey, D. and Yan, J., editors, *Extreme Value Modeling and Risk Analysis*. Chapman & Hall.
- Pfaff, B. and McNeil, A. (2018). *evir: Extreme Values in R*. R package version 1.7-4.
- Ribatet, M. (2022). *SpatialExtremes: Modelling Spatial Extremes*. R package version 2.1-0.
- Ribatet, M. and Dutang, C. (2022). *POT: Generalized Pareto Distribution and Peaks Over Threshold*. R package version 1.1-10.
- Stephenson, A. G. (2002). **evd**: Extreme value distributions. *R News*, 2(2):0.
- Stephenson, A. G. and Heffernan, J. E. (2018). *ismev: An Introduction to Statistical Modeling of Extreme Values*. R package version 1.42.