




## Extremes.jl: Extreme Value Analysis in Julia

**Jonathan Jalbert**   
Polytechnique Montréal

**Marilou Farmer**  
Polytechnique Montréal  
Clinia

**Gabriel Gobeil**  
Polytechnique Montréal  
Environment and Climate Change Canada

**Philippe Roy**   
Institut de Recherche  
d'Hydro-Québec

---

### Abstract

The **Extremes.jl** package provides exhaustive, high-performance functions by leveraging the multiple-dispatch capabilities in Julia for the analysis of extreme values. In particular, the package implements statistical models for both block maxima and peaks-over-threshold methods, along with several methods for the generalized extreme value and generalized Pareto distributions used in extreme value theory. Additionally, the package offers various parameter estimation methods, such as probability-weighted moments, maximum likelihood, and Bayesian estimation. It also includes tools for handling dependence in excesses over a threshold and methods for managing nonstationary models. Inference for extreme quantiles is available for both stationary and nonstationary models, along with diagnostic figures to assess the goodness of fit of the model to the data.

*Keywords:* extreme value analysis, generalized extreme value distribution, generalized Pareto distribution, block maxima, peaks-over-threshold, nonstationary extremes, Julia.

---

## 1. Introduction

Julia (Bezanson, Karpinski, Shah, and Edelman 2012; Bezanson, Edelman, Karpinski, and Shah 2017) is a high-performance language that was built to be remarkably easy to work with. In the context of the climate sciences, Julia offers a high-level alternative to Fortran, which is the language used by most physicists for climate modeling, without sacrificing the computational performance (Edelman 2019). With the massive volume of data that is generated by climate simulations, a fast and concise language is clearly advantageous over other

mainstream analysis languages such as MATLAB (The Math Works Inc. 2022), R (R Core Team 2024) and Python (Python Core Team 2024).

An important aspect of climate science is the risk assessment and impact analysis of extreme values. Recently, the Intergovernmental Panel on Climate Change (IPCC) indicated that extreme events related to sea level, precipitation, temperature, etc., are expected to increase in frequency and intensity with climate change, leading to important impacts on many sectors of activities (IPCC 2021). The only statistical discipline in which a rigorous framework for the study of extreme events has been developed is *extreme value theory* (EVT, e.g., Coles 2001; Beirlant, Goegebeur, Teugels, and Segers 2004). However, unlike other programming languages used by statisticians, only a few packages for the analysis of extreme values exist in Julia despite its growing popularity among the scientific community.

The goal of this paper is to present a new Julia package, **Extremes.jl**, which provides exhaustive high-performance functions for the statistical analysis of extreme values. In particular, methods for block maxima and the above-threshold peak model are implemented. Model parameter estimation can be achieved by using the probability-weighted moments, the maximum likelihood and the Bayesian paradigm. Nonstationary models as well as diagnostic plots are implemented to assess the model accuracy and high quantile estimation. This package is largely inspired by two books. The first one is *An Introduction to Statistical Modeling of Extreme Values* by Coles (2001) and the second one is *Statistics of Extremes: Theory and Applications* by Beirlant *et al.* (2004). The API for **Extremes.jl** is designed to provide a clear approach, leveraging the multiple-dispatch capabilities of Julia. The proposed package is designed to be used by statisticians, climate scientists and engineers. Moreover, this package has already been used in several scientific publications (e.g., Jalbert, Genest, and Perreault 2022). The package is complete and well documented, and there are many tests in place for continuous integration. The tutorial section illustrates the package functionalities by reproducing many of the results obtained by Coles (2001). Extreme value theory is widely used in climate science, but it also plays an important role in insurance, reinsurance and finance (see McNeil, Frey, and Embrechts 2015, for example). The developed package also allows us to use extreme value theory in these contexts.

Probability distribution objects that are commonly used in extreme value analysis, such as the generalized extreme value (GEV) distribution and the generalized Pareto (GP) distribution, already exist in the package **Distributions.jl** (Besançon *et al.* 2021). However, no parameter estimation procedure is available for these distributions. To the best of our knowledge, only one other extreme value analysis software exists in Julia, **ExtremeStats.jl** (J. Hoffmann 2018), but its features are limited. For example, Bayesian inference, diagnostic plots and nonstationary models are not implemented. The implementation of diagnostic plots and nonstationary models is of particular importance for the analysis of environmental extremes due to climate change.

In R, at least thirteen packages exist for extreme value analysis, including **extRemes** (Gilleland and Katz 2016) for univariate extreme value analysis (see Gilleland, Ribatet, and Stephenson 2013, for a short review of these packages). **Extremes.jl** is similar to **extRemes** in R, although the latter has more features, such as point process representation and extremal index estimation. However, **Extremes.jl** is not a mere refactoring of **extRemes**. The package is specifically developed using Julia specific features such as multiple dispatch, which makes it a particularly powerful package. Moreover, the Bayesian functionality of **Extremes.jl** is more efficient because it uses the Hamiltonian Monte Carlo method. This method requires fewer iterations

for the exploration of the posterior distribution than the Gibbs sampling implemented in **extRemes**. For univariate extreme value analysis, **Extremes.jl** has more features than the other packages in R. The **texmex** (Southworth, Heffernan, and Metcalfe 2024) package in R concerns only models for threshold exceedances. The **climextRemes** (Paciorek 2023) package is a specialized package for extreme value analysis in climate science, but it uses **extRemes** for the extreme value analysis. The **evir** (Pfaff and McNeil 2018) package is a useful package for exploratory analysis of extremes, but it does not contain advanced features for parameter estimation, such as the method of moments or the Bayesian method, and it does not support nonstationary modeling.

In Python, libraries for extreme value analysis are very basic. The **ExtremeLy** (Lamichaney 2022) package gathers the existing libraries in Python for extreme value analysis. It allows the estimation of stationary GEV and GP distributions by the method of moments and the method of maximum likelihood only. **pyextremes** (Bocharov 2021) is another package in Python for extreme value analysis. It supports the estimation of stationary GEV and GP distributions using the maximum likelihood method and the Bayesian method but does not support the estimation of nonstationary models.

The remainder of this paper is as follows. In Section 2, the main statistical models used in univariate extreme value analysis are introduced. In Section 3, some of the package functionalities are presented. The parameter estimation techniques and inference methods for high quantiles that are currently supported by **Extremes.jl** are presented in Section 4. In Section 5, the main package types and their hierarchy are presented. Additional features, such as visualization tools and declustering procedures for threshold exceedance clusters, are presented in Section 6. In Section 7, several functionalities are illustrated by reproducing many of the results presented by Coles (2001). Finally, in Section 8, the presented work is concluded, and future development for **Extremes.jl** and extreme value analysis in Julia is presented.

## 2. Theoretical framework

Extreme value analysis is used to estimate extreme events by extrapolating beyond the data range using an asymptotically justified limit model. The block maxima and the peaks-over-threshold models are two classic approaches used to model the extreme values within the extreme value theory framework.

### 2.1. Block maxima

Let  $Y_1, Y_2, \dots$  be a sequence of independently and identically distributed random variables with distribution function  $F$ . Let  $M_n = \max\{Y_1, \dots, Y_n\}$  be the sample maximum of the first  $n$  random variables. The Fisher–Tippett–Gnedenko theorem (Fisher and Tippett 1928; Gnedenko 1943) states that if there exist two sequences of constants  $a_n > 0$  and  $b_n$  such that the normalized maximum  $(M_n - b_n)/a_n$  converges in distribution to a nondegenerate limit  $G_\xi$  as  $n \rightarrow \infty$ , *viz.*

$$\lim_{n \rightarrow \infty} \mathbb{P} \left( \frac{M_n - b_n}{a_n} \leq y \right) = G_\xi(y),$$

then  $G_\xi$  is necessarily of the following form

$$G_\xi(y) = \begin{cases} \exp \left\{ -(1 + \xi y)_+^{-1/\xi} \right\} & \text{if } \xi \neq 0, \\ \exp \{-\exp(y)\} & \text{if } \xi = 0; \end{cases}$$

where  $a_+ = \max\{0, a\}$ . The tail index  $\xi$  describes the tail behavior of  $F$ . If  $\xi$  is negative, the upper tail is bounded,  $F$  is referred to as short-tailed and belongs in the domain of attraction of the Weibull distribution. If  $\xi$  is zero,  $F$  is referred to as light-tailed and belongs in the domain of attraction of the Gumbel distribution. If  $\xi$  is positive, the moments eventually become infinite,  $F$  is referred to as heavy-tailed and belongs to the domain of attraction of the Fréchet distribution.

In practice, if  $n$  is sufficiently large, the distribution of the unnormalized maximum  $M_n$  can be approximated as follows:

$$\mathbb{P}(M_n \leq y) \approx G_\xi \left( \frac{y - b_n}{a_n} \right) = \begin{cases} \exp \left[ - \left\{ 1 + \xi \left( \frac{y - \mu_n}{\sigma_n} \right)_+ \right\}^{-1/\xi} \right] & \text{if } \xi \neq 0; \\ \exp \left[ - \exp \left\{ - \left( \frac{y - \mu_n}{\sigma_n} \right) \right\} \right] & \text{if } \xi = 0. \end{cases} \quad (1)$$

The normalization constants  $a_n$  and  $b_n$  are absorbed into the location and scale parameters  $\mu_n$ , and  $\sigma_n > 0$ . This limiting distribution is referred to as the generalized extreme value (GEV) distribution, and the parameters  $\mu_n$ ,  $\sigma_n$  and  $\xi$  are the location, scale and shape parameters, respectively. When  $\xi = 0$ , the GEV distribution reduces to the Gumbel distribution.

Equation 1 suggests the following statistical model for modeling extreme values. The series of variables  $Y_1, Y_2, \dots$  is partitioned into blocks of  $n$  observations, and the maximum of each of these blocks is computed. Let  $M_{n,i}$  denote the maximum of the  $n$  observations of block  $i$ . If  $n$  is sufficiently large, then the GEV distribution is a reasonable approximation of the distribution of  $M_{n,i}$ , and its parameters can be estimated by using the series of maxima  $M_{n,1}, M_{n,2}, \dots$ . This model is referred to as the block maxima approach. The subscript  $n$  denotes the block size in the maximum expression. The location  $\mu_n$  and scale  $\sigma_n$  depend on the block size  $n$ , but the index  $n$  is usually dropped in the literature when the block size is fixed, and this is how these parameters are presented in the rest of the present paper.

In environmental applications, a block size of one year is often used. A risk measure that is commonly used in such applications is the  $T$ -year return level  $r_T$ , where  $T > 0$ ; the measure is defined as the value that is expected to be exceeded once every  $T$  years. The  $T$ -year return level corresponds to the quantile  $G(r_T) = 1 - 1/T$  of the GEV distribution  $G$  that is used to model the annual maxima. By inverting the cumulative distribution function expressed in Equation 1, the expression for  $r_T$  is as follows:

$$r_T = \mu - \frac{\sigma}{\xi} \left[ 1 - \left\{ -\log \left( 1 - \frac{1}{T} \right) \right\}^{-\xi} \right].$$

In financial applications, value-at-risk is commonly used as a risk measure. It is also a high quantile and it can be estimated by inverting the GEV distribution.

## 2.2. Exceedances above a high threshold

Another popular approach to studying extreme values within the extreme value theory framework consists of modeling the exceedances above a high threshold. Such an approach relies

on the *Pickands-Balkema-de Haan* theorem (Balkema and De Haan 1974; Pickands 1975). Let  $y_* = \sup\{y \in \mathbb{R} : F(y) < 1\}$  be the upper end point of the support of  $F$ , which is the common distribution function of  $Y_1, Y_2, \dots$  that was introduced in the previous section. The theorem states that if there exists a positive scaling function  $c(u)$  such that the scaled excess random variable  $\{(Y - u)/c(u)\} | Y > u$  converges in distribution to a nondegenerate limit  $H_\xi$  as  $u \rightarrow y_*$ , *viz.*

$$\lim_{u \uparrow y_*} \mathbb{P} \left\{ \frac{Y - u}{c(u)} \leq z \mid Y > u \right\} = H_\xi(z), \text{ for } z \in \mathbb{R};$$

Then,  $H_\xi$  is necessarily of the following form:

$$H_\xi(z) = 1 - (1 + \xi z)_+^{-1/\xi}, \text{ with } \xi \in \mathbb{R}.$$

In practice, if the threshold  $u$  is sufficiently high, the unnormalized conditional tail distribution can be approximated as follows:

$$\mathbb{P}(Y - u \leq z | Y > u) \approx H_\xi \{c(u) z\} = \begin{cases} 1 - \left(1 + \xi \frac{z}{\sigma_u}\right)_+^{-1/\xi} & \text{if } \xi \neq 0, \\ 1 - \exp\left(-\frac{z}{\sigma_u}\right) & \text{if } \xi = 0; \end{cases} \quad (2)$$

The normalization function  $c(u)$  is absorbed into the scale parameter  $\sigma_u > 0$ . This limiting distribution is referred to as the generalized Pareto (GP) distribution, and the parameters  $\sigma_u$  and  $\xi$  are the scale parameter and the tail index, respectively. This last parameter is the same as the one corresponding to the GEV distribution and its interpretation remains the same. The subscript  $u$  is commonly dropped in the scale parameter notation and is also dropped in the remainder of this paper.

Based on Equation 2, Davison and Smith (1990) proposed the following model to characterize the exceedances above a high threshold  $u$ :

$$\mathbb{P}(Y > u + z) = \mathbb{P}(Y > u + z | Y > u) \times \mathbb{P}(Y > u) \text{ for } z \geq 0. \quad (3)$$

If  $u$  is large enough, the conditional distribution on the right-hand side can be approximated with the GP distribution expressed in Equation 2, which is referred to as the peaks-over-threshold (POT) method. Often,  $\zeta_u$  denotes the probability  $\mathbb{P}(Y > u)$  of exceeding the threshold.

Notably, Coles (2001, Chapter 4) mentioned that the choice of threshold  $u$  is critical for using the last approximation. As the GP distribution is only asymptotically justified, a threshold that is too low leads to biased estimates, and a threshold that is too high leaves very few data for inference, leading to a large estimation variance. In practice, parameter stability plots (Coles 2001, Chapter 4) and mean residual life plots (Davison and Smith 1990) are utilized to select a suitable threshold. In a stability plot, the estimates of both  $\sigma$  and  $\xi$  in the GP distribution for a range of thresholds are shown. An appropriate threshold can then be defined by the lowest threshold in which both the shape and scale estimates remain constant. If  $\xi = 0$ , Coles (2001) proposed a reparametrization of the scale parameter which remains constant in the range of appropriate thresholds. Additionally, an appropriate threshold can be defined as one in which the mean of the exceedances (i.e., the mean residual life) varies linearly with the threshold value.

In environmental applications, the POT model is often used to estimate the  $T$ -year return level. Suppose that the data consist of  $m$  observations per year; then, the  $T$ -year return level can be obtained by inverting Equation 3:

$$r_T = u + \frac{\sigma}{\xi} \left\{ (Tm\zeta_u)^\xi - 1 \right\}, \quad (4)$$

where  $\zeta_u = \mathbb{P}(Y > u)$ .

### 2.3. Dependent sequences

When the stationary series  $Y_1, Y_2, \dots$  exhibits dependence, the abovementioned theorems still hold if the dependence is of short range. Leadbetter, Lindgren, and Rootzén (1983) provided a precise mathematical definition for this condition on dependence. In the block maxima model, the distribution of the block maximum still converges in distribution to the GEV distribution as  $n \rightarrow \infty$ . The threshold exceedance also still converges in distribution to the GP distribution as  $u \uparrow y^*$ . However, clustering tends to occur when the series exhibits dependence.

To avoid modeling the dependence of threshold exceedances of the same cluster in the POT model, a procedure called declustering is often used to retain only the maximum of the cluster. The different cluster maxima can then be assumed to be independent and identically distributed according to the GP distribution. Beirlant *et al.* (2004) described several methods to identify these clusters.

### 2.4. Nonstationary sequences

In the case where the sequence of random variables  $Y_1, Y_2, \dots$  is nonstationary, there is unfortunately no general formal framework for the analysis of extreme values. Generally, methods qualified as pragmatic by Coles (2001, Chapter 6) are used. Let  $M_1, M_2, \dots$  denote the sequence of block maxima. Since every  $M_i$ , where  $i \in \{1, 2, \dots\}$ , corresponds to a maximum, it is natural to assume that its distribution can be well approximated by the GEV distribution, provided that the block size is large enough. However, the parameters of the GEV distribution for each of the maxima may vary because of the nonstationarity of the underlying series. The usual nonstationary model for block maxima is expressed as follows:

$$M_i \approx \mathcal{GEV}(\mu_i, \sigma_i, \xi_i).$$

From a statistical perspective, identifying one or more explanatory variables that can model the nonstationarity of the parameters is important.

Similarly, let  $Z_i$  be the  $i$ th declustered excess above the threshold, i.e., the threshold is subtracted from the exceedance. The generalized Pareto distribution is a reasonable choice to approximate the distribution of this excess, provided that the threshold is sufficiently high.

$$Z_i \approx \mathcal{GP}(\sigma_i, \xi_i).$$

The GP parameters can vary, and a set of explanatory variables are needed to model this nonstationarity.

For a nonstationary model, the notion of a return period is inappropriate since the probability of exceeding a certain value changes during the period studied. According to Katz, Parlange,

and Naveau (2002), a simple generalization of the return levels are the effective return levels. The effective return level is defined by the return level corresponding to particular parameter values. For example, for the block maxima model, a return level associated with  $M_i$  can be calculated using the triplet of parameters  $(\mu_i, \sigma_i, \xi_i)$ . This is the return level that would be obtained if the conditions of the maximum  $M_i$  continued indefinitely. The return levels for the set of indices  $i$  correspond to the effective return levels. Their analogs can be obtained using the POT model.

### 3. Package overview

Before describing the package architecture in detail, some of its functionalities are first presented to facilitate their justification in the next sections. The analysis of extreme sea level values at Port Pirie, Australia, is performed with **Extremes.jl**. The data provided by Coles (2001) consist of the annual sea level maxima recorded at Port Pirie from 1923 to 1987. In this section, only the block maxima model and the maximum likelihood parameter estimation method are used, as in Coles (2001, Chapter 3 and Chapter 6). In Section 7, other extreme value models and other parameter estimation methods are presented.

#### 3.1. Loading the Port Pirie data

All the datasets analyzed by Coles (2001) are available in **Extremes.jl**, and they can be loaded with the function `dataset`. The following command loads the Port Pirie data:

```
julia> data = Extremes.dataset("portpirie")
```

This command returns a `DataFrame`, where each row consists of the year and the corresponding sea level maximum. The data are illustrated in Figure 1.

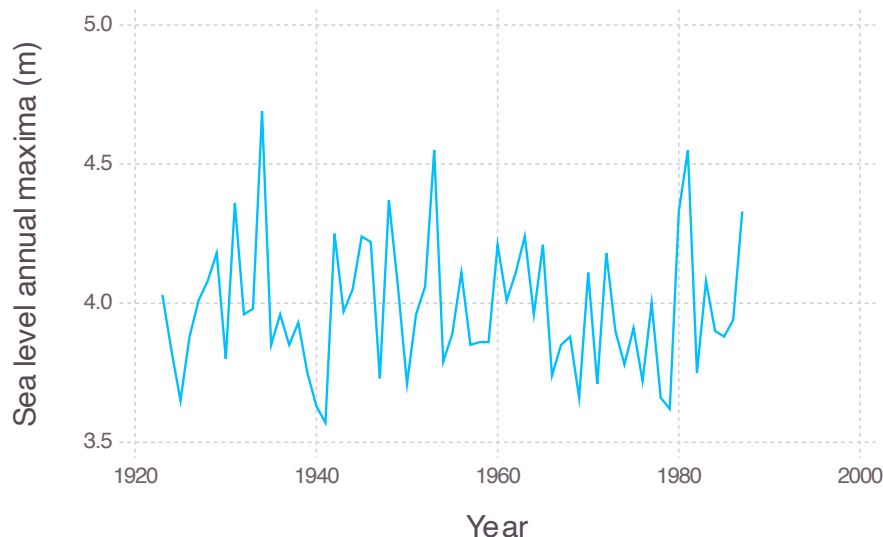


Figure 1: Annual maximum sea levels at Port Pirie.



### 3.2. Block maxima model

#### Parameters estimation

As mentioned in Section 2.1, the GEV is a reasonable approximation for the distribution of the annual maxima. The GEV parameter estimation using the maximum likelihood is performed with the function `gevfit`. When the data are stored in a `DataFrame`, the name of the `DataFrame` and the data column identifier can be directly passed to the function as follows:

```
julia> fm = gevfit(data, :SeaLevel)
```

```
MaximumLikelihoodAbstractExtremeValueModel
model :
```

```
BlockMaxima{GeneralizedExtremeValue}
data: Vector{Float64}[65]
location:  $\mu \sim 1$ 
logscale:  $\phi \sim 1$ 
shape:  $\xi \sim 1$ 
```

```
 $\hat{\theta}$ : [3.874750223091266, -1.6192723640210762, -0.05010719929448139]
```

The function returns `MaximumLikelihoodAbstractExtremeValueModel`  $<$ : `AbstractFittedExtremeValueModel`, which contains the extreme value model type `BlockMaxima`  $<$ : `AbstractExtremeValueModel` and the parameter estimates  $\hat{\theta}$ . These structures are detailed in Section 5. Note that by default, the logarithm of the scale parameter  $\phi = \log \sigma$  is returned. From this output, the GEV parameter estimates are

$$\hat{\mu} \approx 3.875, \quad \hat{\sigma} \approx 0.198 \quad \text{and} \quad \hat{\xi} \approx -0.050,$$

which are the same estimations as those obtained by Coles (2001, Chapter 3).

The approximate covariance matrix of the parameter estimates can be obtained using the function `parametervar`, which takes the output of `gevfit` as input:

```
julia> V = parametervar(fm)
```

```
3×3 Matrix{Float64}:
 0.000780204  0.000995016  -0.0010741
 0.000995016  0.0104541    -0.00392576
-0.0010741   -0.00392576    0.00965404
```

The corresponding approximate 95% confidence intervals for the parameters can be obtained using the function `cint`, which also takes as input the output of `gevfit`:

```
julia> cint(fm, 0.95)
```

```
3-element Vector{Vector{Float64}}:
 [3.820004234825991, 3.929496211356541]
 [-1.8196698585895985, -1.418874869452554]
 [-0.24268345866324603, 0.14246906007428323]
```



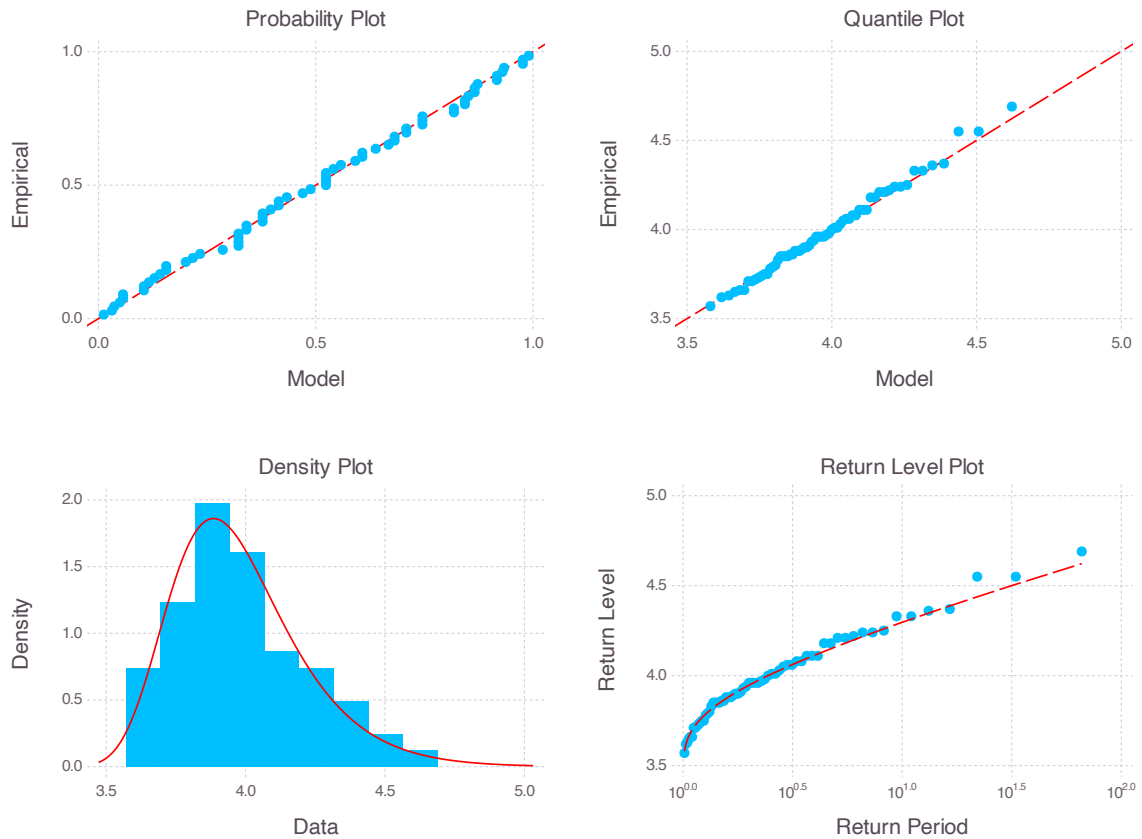


Figure 2: Diagnostic plots for GEV fit of the Port Pirie sea level data.

These are the confidence intervals for the location, log scale and shape parameters.

Note that inference based on the Gumbel distribution is also provided in the package. All the methods presented are also applicable to the Gumbel case.

### *Goodness of fit*

Diagnostic plots to evaluate the goodness of fit of the model can be displayed using the function `diagnosticplots`, which takes as argument the type `AbstractFittedExtremeValueModel`:

```
julia> diagnosticplots(fm)
```

The corresponding plots are shown in Figure 2.

### *Return level*

The  $T$ -year return level estimate can be obtained using the function `ReturnLevel`. For example, the estimated 100-year return level for the Port Pirie block maxima model is computed as follows:

```
julia> r = returnlevel(fm, 100)
```

```
ReturnLevel
returnperiod: 100
value: Vector{Float64}[1]
```

This function returns a `ReturnLevel` that contains the return period and the return level. In this example, the return level can be extracted using `r.value`, and a value of [4.69] is returned. The approximate 95% confidence interval for the 100-year return level can be obtained using

```
julia> c = cint(r, 0.95)
```

```
1-element Vector{Vector{Real}}:
 [4.377121171613508, 4.999685549252193]
```

### 3.3. Nonstationary block maxima model

Using Julia's multiple-dispatch capability, the same functions used in the previous section can be applied to the nonstationary model for parameter estimation, display of the diagnostic figures and estimation of effective return levels. For the Port Pirie example, the year can be used as the covariate.

#### *Parameter estimation*

Coles (2001, Chapter 6) proposed modeling the GEV location parameter as a linear function of the year. This model has 4 parameters. In `Extremes.jl`, the maximum likelihood parameter estimation of this type of model is obtained using the function `gevfit`, where the covariate identifiers for the location parameter are provided by the optional argument `locationcovid`:

```
julia> fm1 = gevfit(data, :SeaLevel, locationcovid = [:Year])
```

```
MaximumLikelihoodAbstractExtremeValueModel
model :
  BlockMaxima{GeneralizedExtremeValue}
  data: Vector{Float64}[65]
  location:  $\mu \sim 1 + \text{Year}$ 
  logscale:  $\phi \sim 1$ 
  shape:  $\xi \sim 1$ 
```

```
 $\hat{\theta}$ : [4.568356786108704, -0.0003547310197546491,
      -1.6196104648710474, -0.050460448403627546]
```

The approximate covariance matrix of the parameter estimates can be obtained using the function `parametervar`, and the corresponding confidence intervals can be obtained using the function `cint`.

#### *Goodness of fit*

The goodness of fit can be assessed based on the diagnostic plots of the standardized maxima conditional on the estimated parameters (see Coles 2001, Chapter 6). These plots are obtained using the function

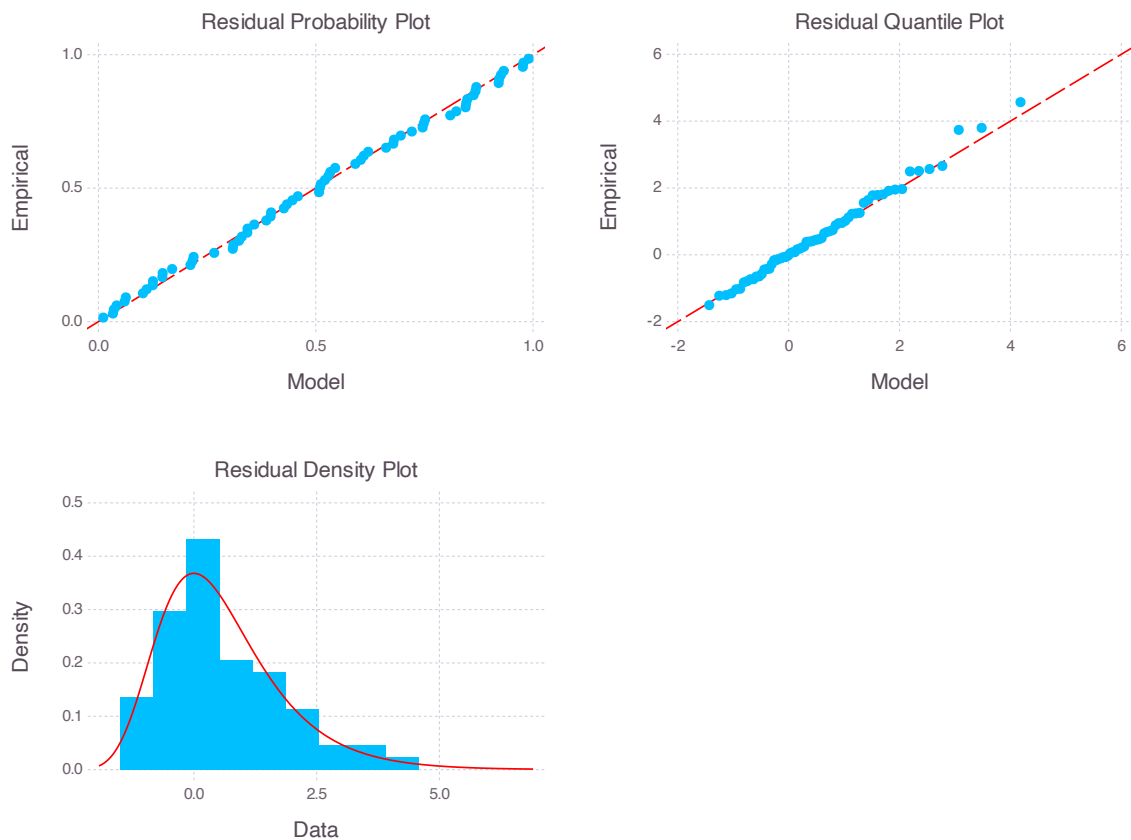


Figure 3: Diagnostic plots for GEV fit of the Port Pirie sea level data.

```
julia> diagnosticplots(fm1)
```

and are illustrated in Figure 3.

### *Effective return levels*

The  $T$ -year effective return level estimate can be obtained using the function `ReturnLevel`. For example, the estimated 100-year effective return level for the Port Pirie block maxima model is computed as follows:

```
julia> r = returnlevel(fm1, 100)
```

```
ReturnLevel
```

```
returnperiod: 100
```

```
value: Vector{Float64}[65]
```

The function returns a `ReturnLevel` containing the return period and 65 effective return levels. The effective return level vector can be extracted using `r.value`, and  $[4.70, \dots, 4.68]^T$  is returned. The 95% confidence interval for the 100-year effective return levels can be obtained using

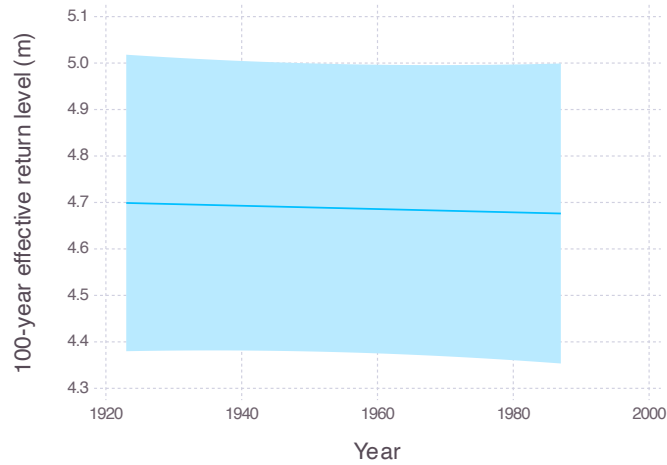


Figure 4: The 100-year effective return levels of the sea level at Port Pirie. The solid line corresponds to the maximum likelihood estimate, and the 95% confidence interval is illustrated by the ribbon.

```
julia> cint(r, 0.95)
```

The effective return levels and their corresponding 95% confidence intervals are shown in Figure 4. In this Port Pirie example, the applied non-stationary methods indicated no evidence of non-stationarity. Conversely, several examples in Section 7 demonstrate instances of non-stationarity.

## 4. Estimation

### 4.1. Parameter estimation

Many techniques have been proposed for parameter estimation in extreme value models. The **Extremes.jl** package currently supports three of these techniques: maximum likelihood inference, Bayesian inference and inference based on the probability-weighted moments. In the case of a nonstationary extreme value model, the covariates are standardized before the parameters are estimated to help fit the model. The model parameters and the standardized variables are transformed back to their original scales before the fitted model is returned.

#### *Maximum likelihood*

The functions `gevfit` and `gpfit` are provided for parameter estimation by maximum likelihood for the block maxima model and the threshold exceedances model, respectively. These functions can be called with the data vector as an input or with a `DataFrame` containing the data. The Nelder-Mead solver (Nelder and Mead 1965; Gao and Han 2012) is implemented in the package **Optim.jl** (Mogensen and Riseth 2018) to find the point estimate that maximizes the log-likelihood.

The approximate covariance matrix of the parameter estimates can be obtained with the function `parametervar`. This matrix is based on the inverse of the observed Fisher information matrix. The corresponding Wald confidence interval on parameter estimates can be obtained using the function `cint`.

### *Bayesian paradigm*

In **Extremes.jl**, Bayesian parameter estimation is performed through the functions `gevfitbayes` and `gpfitbayes` for the block maxima model and the threshold exceedances model, respectively. These functions simulate an autocorrelated sample from the parameter posterior distribution using the No-U-Turn Sampler extension (Hoffman and Gelman 2014) on the Hamiltonian Monte Carlo (Neal 2011) implemented in the package **Mamba.jl** (Smith 2014).

Currently, only the improper flat prior is implemented. It yields a proper posterior as long as the sample size is larger than 3 (Northrop and Attalides 2016). A functionality allowing the use of a prior defined by the user will be added in the next version of the package.

Credible intervals on the parameter are obtained using the function `cint` with the appropriate empirical quantiles of the simulated posterior distribution sample. The parameter covariance matrix is estimated using the empirical covariance of the sample in the function `parametervar`.

### *Probability-weighted moments*

Parameter estimates based on the probability-weighted moments (Landwehr, Matalas, and Wallis 1979) are provided through the functions `gevfitpwm` and `gpfitpwm` for the block maxima and the threshold exceedances model, respectively. Unlike the two other implemented estimation procedures in **Extremes.jl**, the parameter estimation based on the probability-weighted moments is only possible for the stationary extreme value model.

The covariance matrix of the parameter estimates obtained using the probability-weighted moments is estimated by nonparametric bootstrapping in the function `parametervar`. Confidence intervals on the parameter estimates are estimated by nonparametric bootstrapping in the function `cint`.

## 4.2. Return levels

Inference for the return levels can be obtained using the function `ReturnLevel`, which is applied to a `AbstractFittedExtremeValueModel` structure. Confidence intervals on the return level can be obtained using the function `cint`. When the estimation procedure is the maximum likelihood approach, the Wald confidence interval using the variance estimated by the delta method is returned. When the probability-weighted moments are used, the confidence interval is computed using nonparametric bootstrapping. Under the Bayesian paradigm, the empirical quantile of the return level posterior distribution is returned as the credible interval.

These functions return a unit dimension vector for the return level and one vector for the confidence interval. The reason is that the function always returns the same type in the stationary and nonstationary cases. The function is therefore *type-stable*, allowing better code execution performance.

For the nonstationary models, since the model parameters vary over time, the quantiles also vary over time. Therefore, a  $T$ -year return level can be estimated for each year. This set of return levels are referred to as effective return levels, as proposed by Katz *et al.* (2002).

## 5. Type hierarchy

**Extremes.jl** leverages the multiple-dispatch capabilities of Julia (see Zappa Nardelli, Belyakova, Pelenitsyn, Chung, Bezanson, and Vitek 2018). Several methods have been developed depending on the type of input argument. There are four main abstract types defined in **Extremes.jl**: `DataItem`, `AbstractExtremeValueModel`, `AbstractFittedExtremeValueModel` and `ReturnLevel`. When a parameter estimation function is called to estimate the parameters of an extreme value model, the data and the explanatory variables are first encapsulated in objects of type `Variable`, and the extreme value model is built in the `AbstractExtremeValueModel` structure. The appropriate method of parameter estimation is performed, and an object of type `AbstractFittedExtremeValueModel`, which contains the results of the estimation procedure, is returned. The function `ReturnLevel` used to estimate the quantiles that can be applied to an `AbstractFittedExtremeValueModel` object, and an object of type `ReturnLevel` is returned. The relation between these types is illustrated in Figure 5. This section describes these main abstract types and their utility. Figure 6 illustrates the complete hierarchy.

### 5.1. DataItem

`DataItem` is an abstract type that encapsulates the data and its label. Several methods are defined for this type of data manipulation, for example, data standardization. It contains the composite type `Variable` that holds the name (`String`) and values (`Vector{<:Real}`) of a variable. It also contains the composite type `VariableStd` that holds the name (`String`), the standardized values (`Vector{<:Real}`), the offset (`Real`), and the scale offset (`Real`) values of the transformation.

This type is useful when explanatory variables are introduced to model nonstationarity. These variables are encapsulated in `Variable`. In the parameter estimation procedure, the variables are standardized and stored in `VariableStd`.

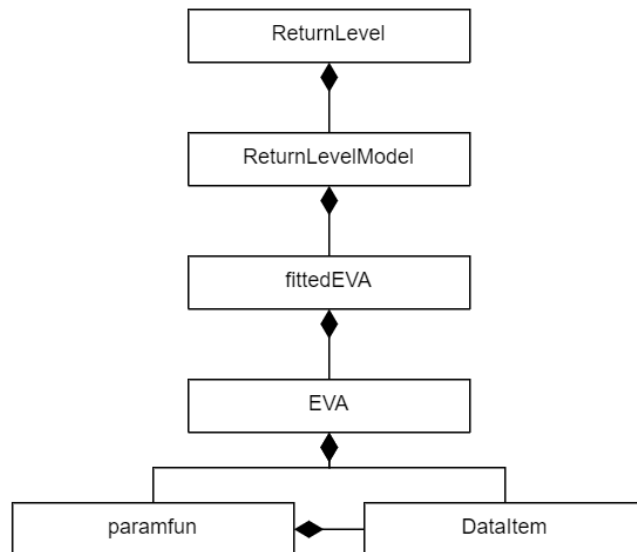


Figure 5: Abstract type hierarchy in **Extremes.jl**.

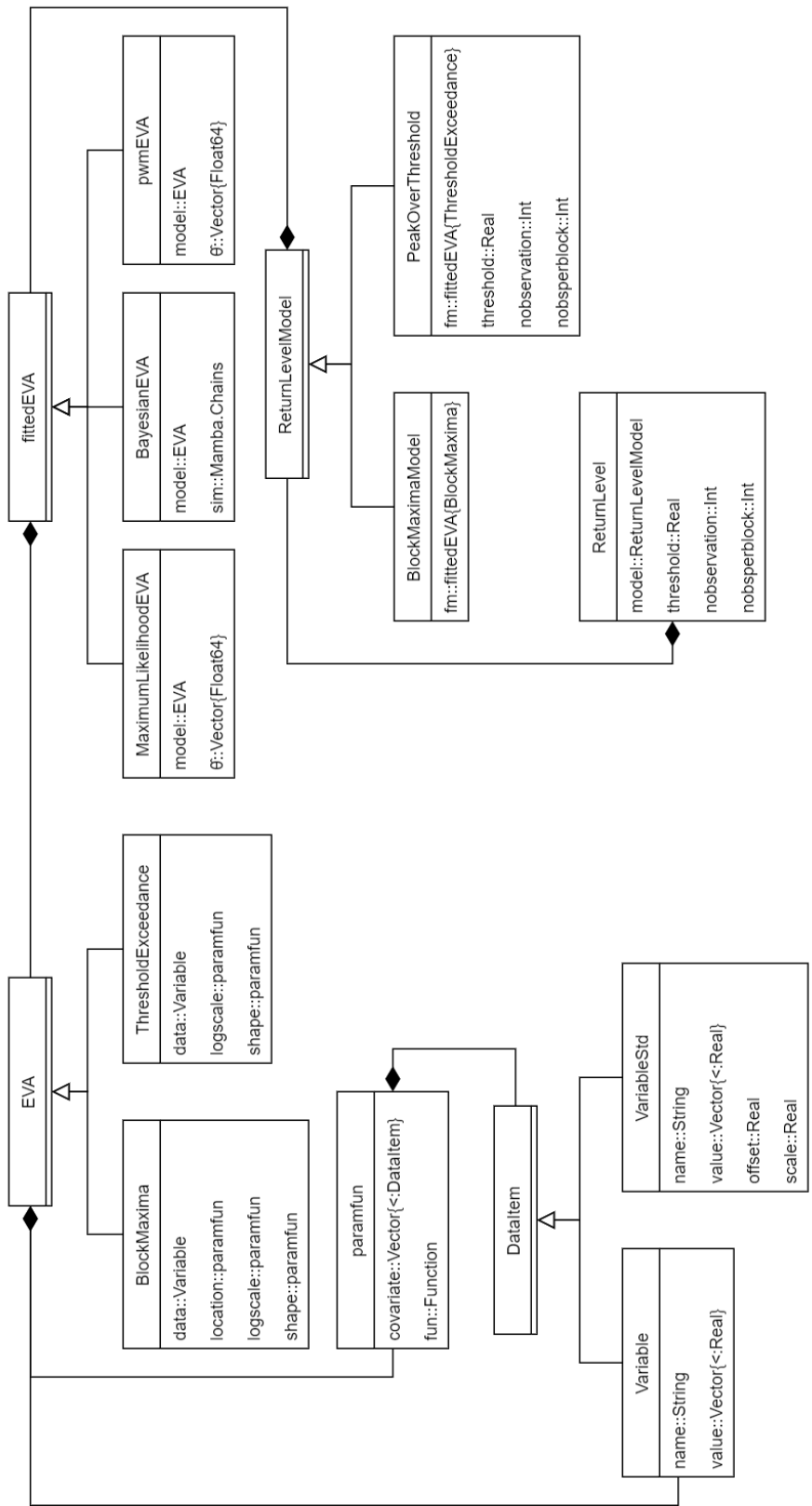


Figure 6: Complete type hierarchy in **Extremes.jl**.



This standardization allows for greater robustness of the estimation procedure. The variables are then transformed back to their original scale using the offset and scale as well as the extreme value model parameters before returning the results of the parameter estimation procedure.

## 5.2. AbstractExtremeValueModel

`AbstractExtremeValueModel` is an abstract type for the two classical extreme value models, the block maxima and the peaks-over-threshold models. For each of these models, two composite subtypes of `AbstractExtremeValueModel`, `BlockMaxima` and `ThresholdExceedance`, are defined to encapsulate the statistical model.

The `BlockMaxima` subtype is defined as follows:

```
julia> struct BlockMaxima <: AbstractExtremeValueModel
    data::Variable
    location::paramfun
    logscale::paramfun
    shape::paramfun
end
```

where `data` corresponds to the block maxima data encapsulated in `Variable` and `location`, `logscale` and `shape` correspond to the GEV parameters as a function of the given covariates. The parameter function can be the identity function in the case of a constant parameter. This allows **Extremes.jl** to hold nonstationary models within the same structures as stationary models.

The `ThresholdExceedance` subtype is defined as follows:

```
julia> struct ThresholdExceedance <: AbstractExtremeValueModel
    data::Variable
    logscale::paramfun
    shape::paramfun
end
```

where the exceedances above the threshold are encapsulated in `data`, and the `logscale` and `shape` represent the GP distribution parameters as a function of the given covariates. Similar to `BlockMaxima`, the `ThresholdExceedance` structure can contain both stationary and nonstationary extreme value models for exceedances above a high threshold.

## 5.3. AbstractFittedExtremeValueModel

`AbstractFittedExtremeValueModel` is a composite subtype of `AbstractExtremeValueModel` that is defined to contain the data, the extreme value model and the parameter estimates. This structure is flexible enough to contain parameter point estimates when the estimation method is based on maximum likelihood or probability-weighted moments or a sample of the parameters posterior distribution when the estimation is performed under the Bayesian paradigm.

The composite type depends on the estimation procedure, and different methods apply to each of them:

```

julia> struct MaximumLikelihoodAbstractExtremeValueModel{T} <:
AbstractFittedExtremeValueModel{T}
    "Extreme value model definition"
    model::T
    "Maximum likelihood estimate"
     $\hat{\theta}$ ::Vector{Float64}
end

julia> struct BayesianAbstractExtremeValueModel{T} <:
AbstractFittedExtremeValueModel{T}
    "Extreme value model definition"
    model::T
    "MCMC outputs"
    sim::Mamba.Chains
end

julia> struct pwmAbstractExtremeValueModel{T} <:
AbstractFittedExtremeValueModel{T}
    "Extreme value model definition"
    model::T
    "Probability-weighted moments estimate"
     $\hat{\theta}$ ::Vector{Float64}
end

```

The extreme value model (`AbstractExtremeValueModel`) is stored in the `model`, and the parameter point estimates are stored in  $\hat{\theta}$ . In the Bayesian model case, the parameter posterior sample `sim` is stored in an object of type `Chains` from the package `Mamba.jl` (Smith 2014).

## 5.4. ReturnLevel

`ReturnLevel` is a composite type that is dependent on the estimation procedure. It has three attributes, the return level model `model`, the `returnperiod` and `value`, that are used to store the type of `AbstractExtremeValueModel` model, the return period and the estimated value, respectively. A `ReturnLevelModel` type depends on the extreme value model, either a block maxima or a peaks-over-threshold model.

```

julia> struct ReturnLevel{T} <: AbstractFittedExtremeValueModel{T}
    model::ReturnLevelModel{T}
    returnperiod::Real
    value::Array{<:Real}
end

```

## 6. Additional features

### 6.1. Visualization tools

Several validation plots for assessing the accuracy of the fitted models are implemented in `Extremes.jl` using the function `plot` from the package `Gadfly.jl` (Jones *et al.* 2021).

*Diagnostic plots*

Diagnostic plots for `AbstractFittedExtremeValueModel` can be built using the function `diagnosticplots`. It consists of a four-panel plot (see Figure 2, Section 3.2) that includes a probability plot (upper left panel), a quantile plot (upper right panel), a density plot (lower left panel) and a return level plot (lower right panel).

Each of these subplots can be built with its own function, which is also defined for `AbstractFittedExtremeValueModel`: `probplot`, `qqplot`, `histplot` and `returnlevelplot`. These functions are specifically designed for extreme values and follow the definitions presented by Coles (2001, Section 2.6.7). In the Bayesian fitted model case, these plots are constructed by taking the mean of the statistics computed over each MCMC iteration.

Moreover, standardized versions of these plots, `diagnosticplots_std`, `probplot_std` and `qqplot_std`, are defined for residuals in the case of nonstationary models.

*Mean residual life plot*

The mean residual life plot can be plotted using the function `mrlplot` to help users choose a suitable threshold for the peaks-over-threshold model (see Figure 8a, Section 7.1).

*Stability plots*

**Extremes.jl** does not currently provide a function to illustrate stability graphs for threshold selection in the peaks-over-threshold model. The user can plot these graphs by plotting the parameter estimates as a function of the threshold.

**6.2. Declustering**

A composite type `Cluster` is also defined for the threshold exceedance clusters over which declustering methods (see Section 6) are defined.

```
julia> struct Cluster
    "first threshold"
    u1::Real
    "second threshold"
    u2::Real
    "positions of clustered exceedances"
    position::Vector{<:Int}
    "values of clustered exceedances"
    value::Vector{<:Real}
end
```

When the data are dependent, the exceedances tend to cluster. It is common to assume that the marginal distribution of each exceedance is a generalized Pareto (see, for example, Coles 2001, Chapter 5). Various methods have been proposed for dealing with the problem of dependent exceedances. The most widely adopted method consists of declustering the exceedances by filtering the exceedances to obtain a set that is approximately independent. As summarized by Coles (2001), the declustering approach consists of the following:

- identifying clusters of extreme values;

- retrieving the maxima within each cluster;
- assuming that each cluster maxima are independent; and
- fitting the GP distribution to the cluster maxima.

In **Extremes.jl**, two methods for identifying clusters of extreme values are implemented in the function `getcluster`: *the runs method* and *the two thresholds method*. This function extracts the clusters from a vector of values and returns a vector of type `Cluster`.

#### *The two thresholds method*

The two threshold declustering method is used when three arguments are given to the function `getcluster`: the vector of values to decluster and threshold values  $u_1$  and  $u_2$ , where  $u_1 > u_2$ . A vector of clusters is then returned, where each cluster is defined as a sequence of values higher than threshold  $u_2$  with at least a value higher than threshold  $u_1$ .

#### *The runs method*

The runs declustering method (see Coles 2001, Section 5.3.2) is used when two arguments are given to the function `getcluster`: the vector of values to decluster and one threshold value  $u$ . An additional keyword argument `runlength` can be used to specify the runlength parameter  $r$ . Exceedances separated by fewer than  $r$  nonexceedances are assumed to belong to the same cluster (see Beirlant *et al.* 2004, Chapter 10). When  $r = 0$ , each exceedance forms a separate cluster. By default,  $r = 1$  in the function `getcluster` when using the runs declustering method.

## 7. Examples

Four additional examples from Coles (2001) are developed in this section to show the functionalities of **Extremes.jl**. In the first example, the Bayesian stationary threshold exceedances model is illustrated in Section 7.1 using rainfall data (Coles 2001, Chapter 4). Declustering the threshold exceedances is performed in Section 7.2 using the Wooster temperatures dataset (Coles 2001, Chapter 5). The nonstationary block maxima model is illustrated in Section 7.3 using the annual sea level maxima at Fremantle (Coles 2001, Chapter 6) and the nonstationary threshold exceedances model with rainfall in South England (Coles 2001, Chapter 6).

As mentioned in Section 4.1, stationary extreme value models can be fitted by either the probability-weighted moments, maximum likelihood or Bayesian methods, while the probability-weighted moments cannot be applied to nonstationary models. For illustration in this section, the block maxima models are fitted by the maximum likelihood method, and the threshold exceedance models are fitted by the Bayesian method.

In addition to **Extremes.jl**, the **DataFrames.jl**, **Distributions.jl**, and **Gadfly.jl** packages need to be installed to reproduce these examples. These packages can be loaded as follows:

```
julia> using Extremes, Dates, DataFrames, Distributions, Gadfly
```

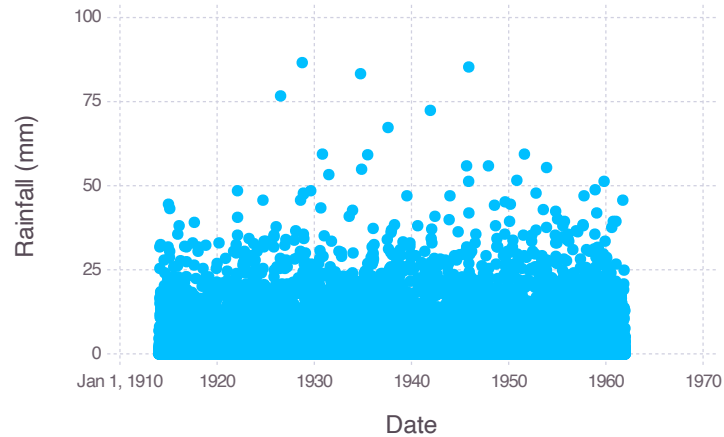


Figure 7: Daily rainfall recorded in *mm* at a location in southwest England.

### 7.1. Threshold exceedances approach

The stationary `ThresholdExceedance` model is illustrated using the daily rainfall accumulations at a location in southwest England from 1914 to 1962, as shown in Figure 7. This dataset was studied by Coles (2001, Chapter 4). The daily rainfall is assumed to be independent and identically distributed. This dataset can be loaded using the following lines of code:

```
julia> data = Extremes.dataset("rain")
```

#### *Threshold selection and exceedances extraction*

Parameter estimation of the generalized Pareto distribution in **Extremes.jl** is performed using the threshold exceedances in  $(0, \infty)$ . To extract the threshold exceedances, a suitable threshold for the peaks-over-threshold model can be chosen by examining the mean residual life plot. The mean residual life plot can be constructed using the function `mrlplot`.

Figure 8a shows the mean residual life plot for daily rainfall accumulations in southwest England. When the threshold is high enough, the mean residual life is expected to be a linear function of the threshold. Here, a reasonable threshold is 30 *mm*, as concluded by Coles (2001, Chapter 4). Once the threshold is defined, the exceedances above this threshold can be retrieved as follows:

```
julia> df_exceedance = filter(row -> row.Rainfall > threshold, data)
julia> df_exceedance.Exceedance = df_exceedance.Rainfall .- threshold
julia> select!(df_exceedance, :Date, :Exceedance)
```

and are shown in Figure 8b.

#### *Bayesian inference*

The generalized Pareto distribution can be fitted to threshold exceedances using the Bayesian method using the function `gpfitbayes`:

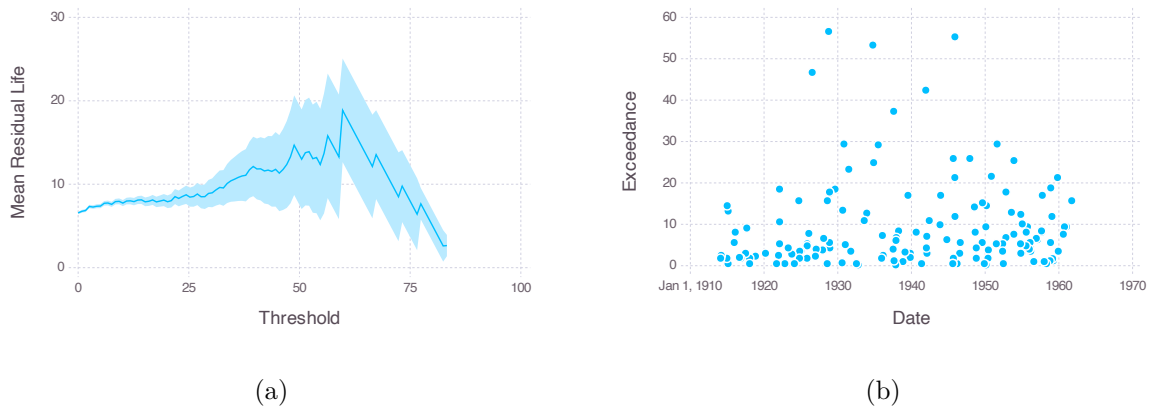


Figure 8: Mean residual life plot of the daily rainfall data (a), and exceedances above the threshold of 30 mm (b).

```
julia> import Random
julia> Random.seed!(4786)
julia> nothing #hide
julia> fm = gpfitybayes(df_exceedance, :Exceedance)
```

```
BayesianAbstractExtremeValueModel
model :
  ThresholdExceedance
  data: Vector{Float64}[152]
  logscale:  $\phi \sim 1$ 
  shape:  $\xi \sim 1$ 

sim :
  Mamba.Chains
  Iterations: 2001:5000
  Thinning interval: 1
  Chains: 1
  Samples per chain: 3000
  Value: Array{Float64, 3}[3000,2,1]
```

By default, 5000 iterations of the Hamiltonian Monte Carlo algorithm are performed. The first 2000 iterations are considered the warmup period, and there is no chain thinning. The result is a sample of size 3000 with a parameter posterior distribution in the format of `Chains` in the package `Mamba.jl`. Figure 9 illustrates the chain traces. The diagnostic plots built with the function `diagnosticplots` are shown Figure 10.

The empirical covariance matrix of the parameters and the credible intervals can be obtained with the functions `parametervar` and `cint`, respectively:

```
julia> parametervar(fm)
```

```
2×2 Matrix{Float64}:
```

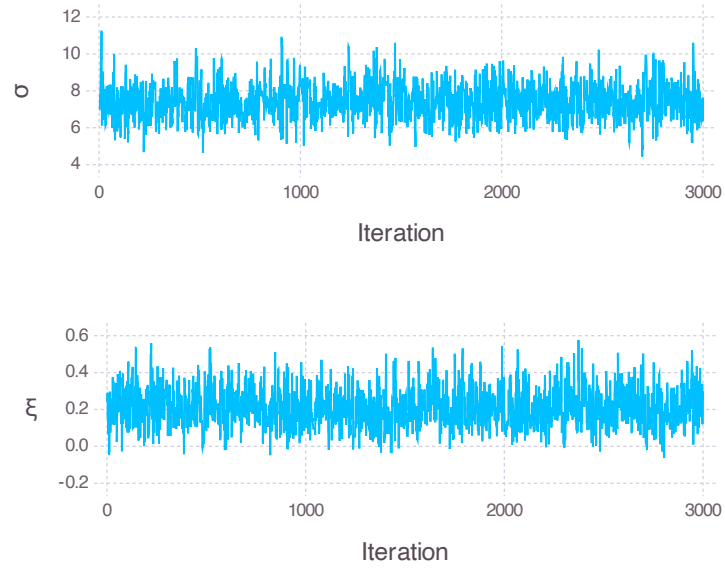


Figure 9: MCMC realizations of the generalized Pareto parameter posterior distribution.

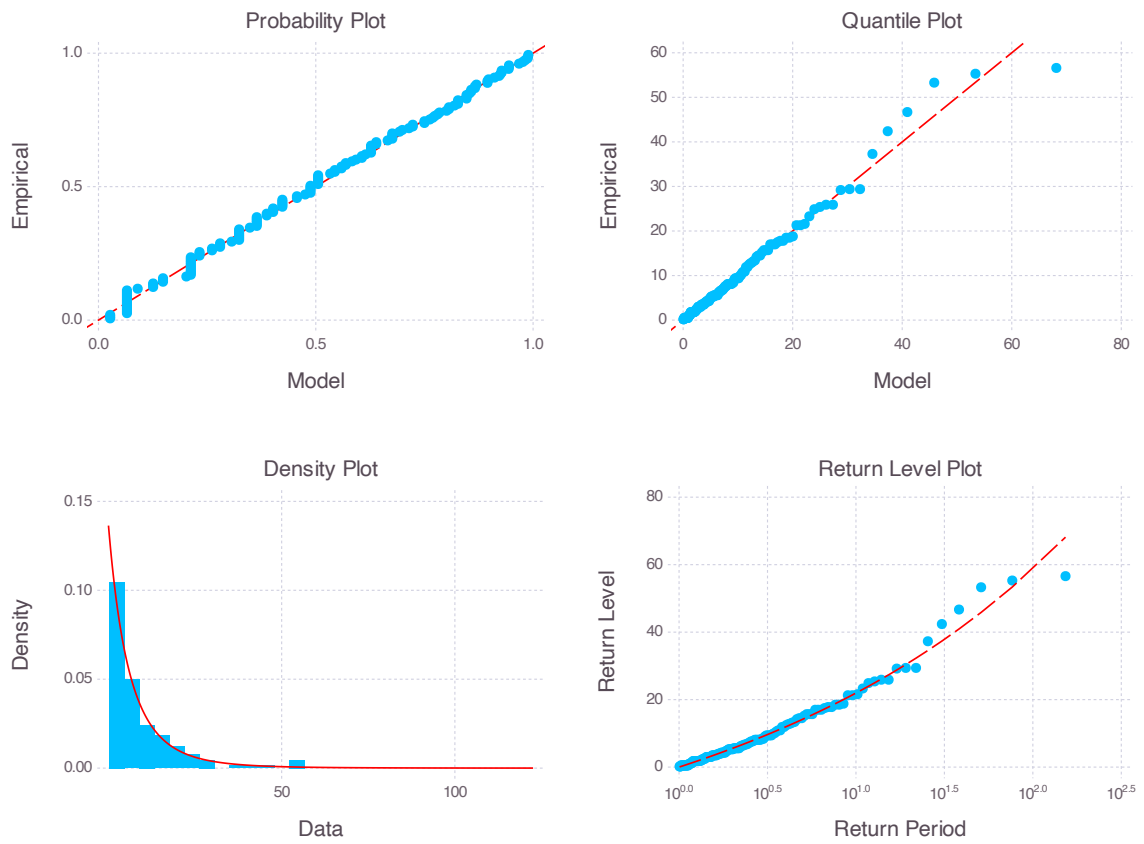


Figure 10: Diagnostic plots for the threshold exceedances model fitted to the daily rainfall data.



```

0.0157694 -0.0083161
-0.0083161 0.0106421

```

```
julia> cint(fm, 0.95)
```

```

2-element Vector{Vector{Float64}}:
 [1.7565728861081145, 2.2442609341726394]
 [-0.002598987690899647, 0.4029866604225447]

```

### *Return level estimation*

The T-year return level estimate can be obtained using the function `returnlevel`. Along with the fitted generalized Pareto distribution for the threshold exceedances, the following information is also needed for estimating the T-year return level using the POT model: the threshold, the number of total observations and the number of observations per year. For example, the 100-year return level for the rainfall POT model is computed as follows:

```

julia> nobs = size(data,1)
julia> nobsperblock = 365
julia> r = returnlevel(fm, threshold, nobs, nobsperblock, 100)

```

```

ReturnLevel
returnperiod: 100
value: Matrix{Float64}[3000]

```

The function returns a `ReturnLevel` structure containing the return period and the return level for each MCMC iteration, 3000 in this case. The corresponding 95% credible interval can be computed using the function `cint`:

```
julia> c = cint(r, 0.95)
```

```

1-element Vector{Vector{Real}}:
 [74.13350590381265, 176.0166014288908]

```

## 7.2. Declustering threshold exceedances

Both methods for identifying the clusters are illustrated with the Wooster dataset studied by Coles (2001, Chapter 5). This dataset consists of the daily temperature minimum recorded in Wooster, Ohio, from 1983 to 1988. The attention is restricted to the winter months (November to February) and to the negated series for using the model defined for maxima. This dataset can be loaded as follows:

```
julia> data = Extremes.dataset("wooster")
```

The opposite temperature of the winter months can be extracted as follows:

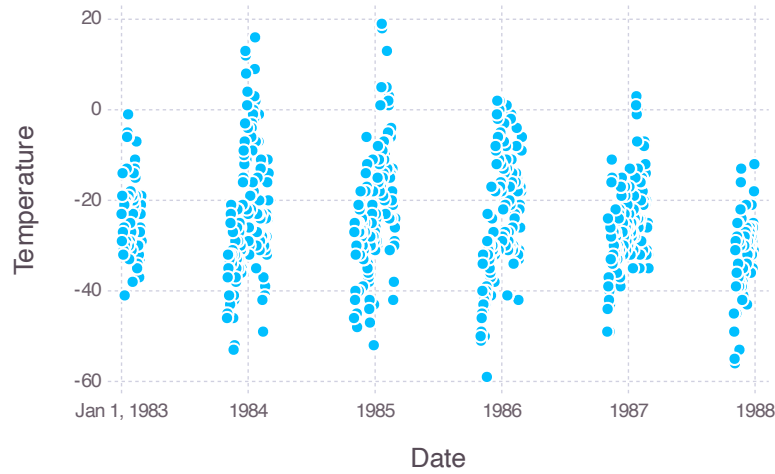


Figure 11: Opposite daily minimum temperatures recorded in winter in Wooster.

```
julia> df = copy(data)
julia> df[!,:Temperature] = -data[:,:Temperature]
julia> filter!(row -> month(row.Date) ∈ (1,2,11,12), df)
```

Figure 11 shows the opposite winter daily minimum temperatures.

#### *The runs method*

When using a single threshold of  $-10^{\circ}F$ , the runs declustering method is used. Setting the runlength to 4, 17 clusters are identified using the function `getcluster`:

```
julia> threshold = -10.0
julia> cluster = getcluster(df[:,:Temperature], threshold, runlength = 4)
```

The first cluster, *viz.*,

```
julia> first(cluster)
```

```
Cluster
u1: -10.0
u2: -10.0
position: [17, 18, 19, 20]
value: [-6, -5, -6, -1]
```

is in positions 17 to 20 of the series, and the opposite of the associated temperatures are  $[-6, -5, -6, -1]$ . These characteristics can be extracted for each cluster. The function `maximum` allows us to extract the maximum value of each cluster:

```
julia> z = maximum.(cluster)
```

```
17-element Vector{Int64}:
-1
-7
13
⋮
-6
 3
-7
```

This function is useful for retrieving the data on which the generalized Pareto distribution can be fitted.

### *The two thresholds method*

In the two threshold method, a cluster of threshold exceedances is defined as a streak of data higher than the second threshold  $u_2$  with at least one data point higher than the first threshold  $u_1$ , where  $u_1 \geq u_2$ . When using  $u_1 = -10^\circ F$  as the first threshold and  $u_2 = -15^\circ F$  as the second threshold, 27 clusters are identified:

```
julia> u1 = -10.0
julia> u2 = -15.0
julia> cluster = getcluster(df[:, :Temperature], u1, u2)
```

The first cluster, *viz.*,

```
julia> first(cluster)
```

```
Cluster
u1: -10
u2: -15
position: [16, 17, 18, 19, 20]
value: [-13, -6, -5, -6, -1]
```

is in positions 16 to 20 of the series, and the opposite of the associated temperatures are  $[-13, -6, -5, -6, -1]$ . These characteristics can be extracted for each cluster. The function `maximum` allows us to extract the maximum value of each cluster:

```
julia> z = maximum.(cluster)
```

```
27-element Vector{Int64}:
-1
-7
13
⋮
-6
 3
-7
```

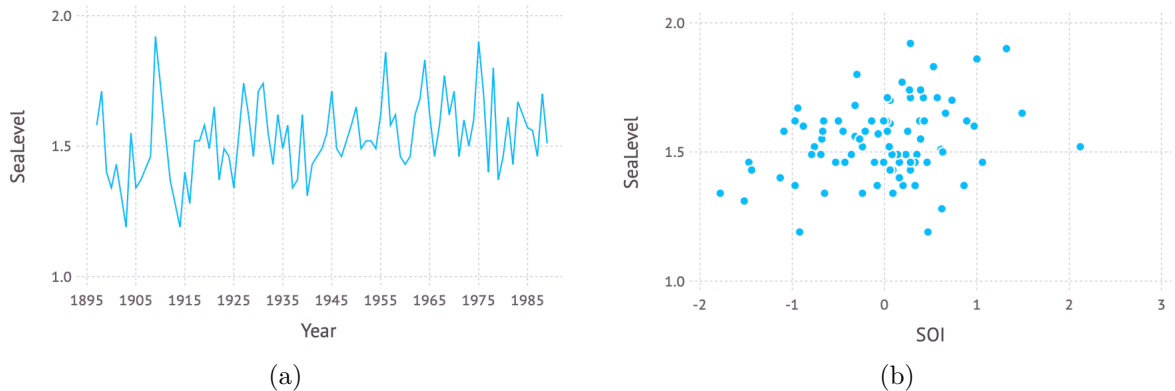


Figure 12: Annual maximum sea levels at Fremantle (a) as a function of the year and (b) as a function of the Southern Oscillation Index (SOI).

### 7.3. Nonstationary block maxima approach

The nonstationary `BlockMaxima` model is illustrated using the annual maximum sea levels recorded at Fremantle in West Australia from 1897 to 1989, as studied by Coles (2001, Chapter 6). Again, this dataset can be loaded as follows:

```
julia> data = Extremes.dataset("fremantle")
```

Figure 12 shows the annual maxima as a function of the year and as a function of the Southern Oscillation Index (SOI).

In the nonstationary block maxima model, each GEV parameter is allowed to be a function of multiple covariates:

$$\begin{aligned}\mu &= X_1 \times \beta_1 \\ \phi &= X_2 \times \beta_2 \\ \xi &= X_3 \times \beta_3\end{aligned}$$

where  $\{X_1, \beta_1\}$ ,  $\{X_2, \beta_2\}$  and  $\{X_3, \beta_3\}$  are the design matrix and the corresponding coefficient parameter vector of  $\mu$ ,  $\phi$  and  $\xi$ , respectively.

#### *Maximum likelihood parameter estimation*

Several nonstationary GEV models using the year and the SOI as covariates can be fitted with the function `gevfit`.

- The stationary model:

```
julia> fm = gevfit(data, :SeaLevel)
```

- The nonstationary model using the year as the covariate of the location parameter:

```
julia> fm1 = gevfit(data, :SeaLevel, locationcovid = [:Year])
```

- The nonstationary model using the year and the SOI as the covariates of the location parameter:

```
julia> fm2 = gevfit(data, :SeaLevel, locationcovid = [:Year, :SOI])
```

- The nonstationary model using the year and the SOI as the covariates of both the location and the log-scale parameters:

```
julia> fm3 = gevfit(data, :SeaLevel, locationcovid = [:Year, :SOI],
                    logscalecovid = [:Year, :SOI])
```

As shown by [Coles \(2001, Chapter 6\)](#), the best model is the one where the location parameter varies as a linear function of the year and the SOI. This model is `fm2` in the present section. Several diagnostic plots for assessing the accuracy of the GEV model fitted to the Fremantle data can be shown with the function `diagnosticplots` and are illustrated in [Figure 13](#). The approximate covariance matrix of the parameter estimates for this model can be obtained using the function `parametervar`:

```
julia> parametervar(fm2)
```

```
5×5 Matrix{Float64}:
```

```
 1.02181      -0.000524501  ...  0.0127779      -0.0209508
-0.000524501  2.69285e-7      -6.4457e-6      1.05896e-5
 6.71532e-5   -1.61708e-8      0.000239274    -0.000355337
 0.0127779    -6.4457e-6      0.00703707     -0.00256783
-0.0209508    1.05896e-5      -0.00256783    0.00444377
```

The corresponding Wald 95% confidence intervals of the parameters are computed with the function `cint`:

```
julia> cint(fm2, 0.95)
```

```
5-element Vector{Vector{Float64}}:
```

```
[-4.607209165941878, -0.6447744599886223]
[0.0010969546332656244, 0.0031311107401392115]
[0.01603794386329234, 0.09299743628807014]
[-2.27859801793459, -1.9497660238399186]
[-0.2806398353854561, -0.0193312124712762]
```

### *Return level estimation*

Since the model parameters vary over time, the quantiles also vary over time. Therefore, the 100-year effective return levels ([Katz et al. 2002](#)) can be estimated for each year as follows:

```
julia> r = returnlevel(fm2, 100)
```

```
ReturnLevel
returnperiod : 100
value :      Vector{Float64}[86]
```

and the corresponding 95% confidence intervals are given as:

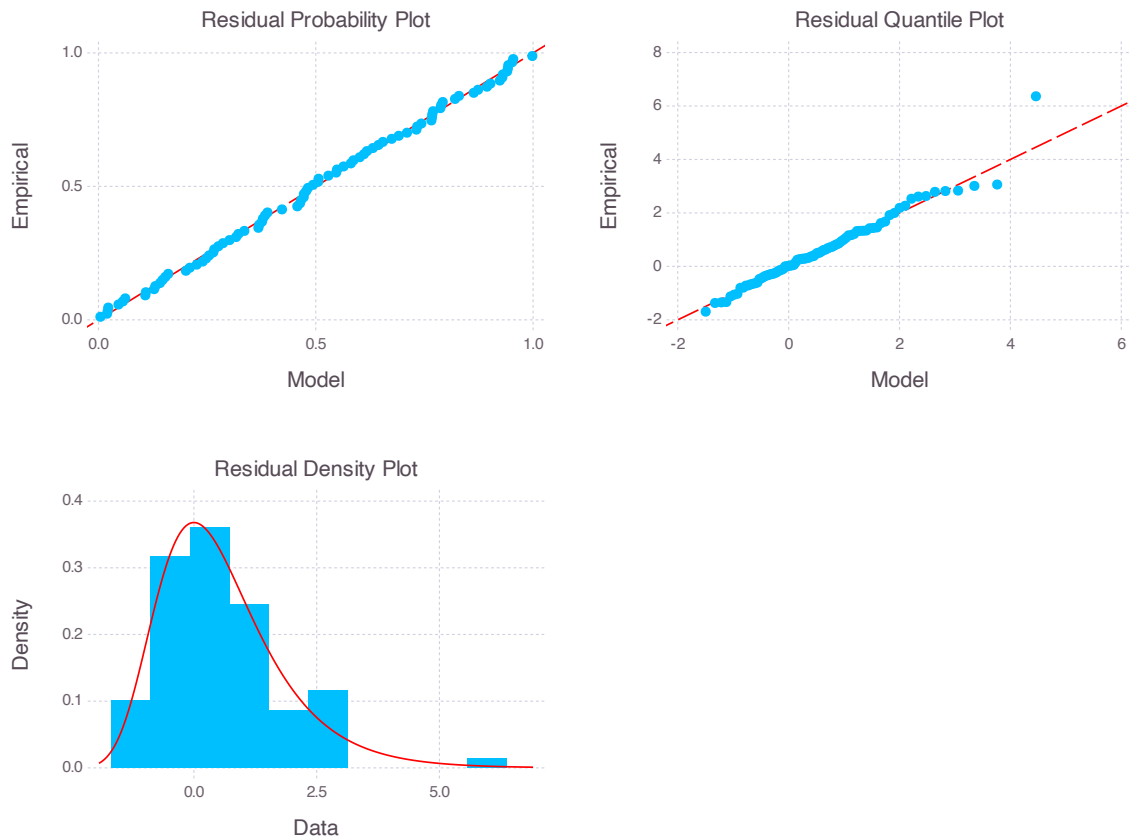


Figure 13: Residual diagnostic plots in the linear trend GEV model of the Fremantle annual maximum sea level series.

```
julia> c = cint(r, 0.95)
```

```
86-element Vector{Vector{Real}}:
 [1.6464829066065432, 1.8515044599835624]
 [1.7251036134261084, 1.9123156899250917]
 [1.7048688798346419, 1.8920739830279043]
 ⋮
 [1.7646548289730073, 2.0266301171024437]
 [1.9073298034621462, 2.128061044319707]
 [1.9027214984142944, 2.1238131691228004]
```

Figure 14 shows the estimated 100-year return levels along with their 95% confidence intervals estimated with the  $\text{fm}_2$  model.

#### 7.4. Nonstationary threshold exceedances approach

The nonstationary threshold exceedances model is illustrated again using the daily rainfall at a location in southwest England from 1914 to 1962 (Coles 2001, Chapter 6), as shown in Figure 7.

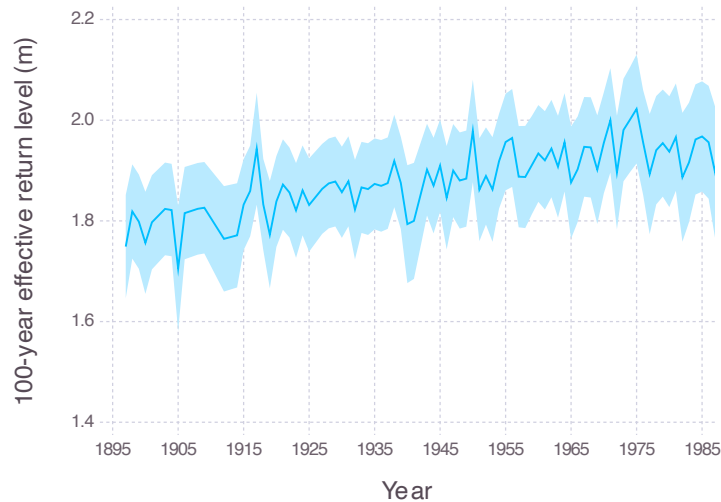


Figure 14: The 100-year effective return level of the sea level at Fremantle.

In the nonstationary threshold exceedances model, the GP parameters are allowed to be functions of multiple covariates:

$$\begin{aligned}\phi &= X_2 \times \beta_2 \\ \xi &= X_3 \times \beta_3\end{aligned}$$

where  $\{X_2, \beta_2\}$  and  $\{X_3, \beta_3\}$  are the design matrix and the corresponding coefficient parameter vector of  $\phi$  and  $\xi$ , respectively.

### *Bayesian inference*

The Bayesian estimation of the parameters of the GP distribution is performed with the function `gpfitbayes`. In addition to the stationary model, the nonstationary model using the year as the covariate for the log-scale GP parameter can be fitted as follows:

```
julia> import Random
julia> Random.seed!(4786)
julia> nothing #hide
julia> fm1 = gpfitbayes(df, :Exceedance, logscalecovid = [:Year])
```

The model fit can be assessed with the diagnostic plots shown in Figure 15.

The empirical covariance matrix of the parameters and the credible intervals can be obtained with the functions `parametervar` and `cint`, respectively:

```
julia> parametervar(fm1)
julia> cint(fm1, 0.95)
```

### *Return level estimation*

The 100-year effective return level estimates can be obtained using the function `returnlevel`.



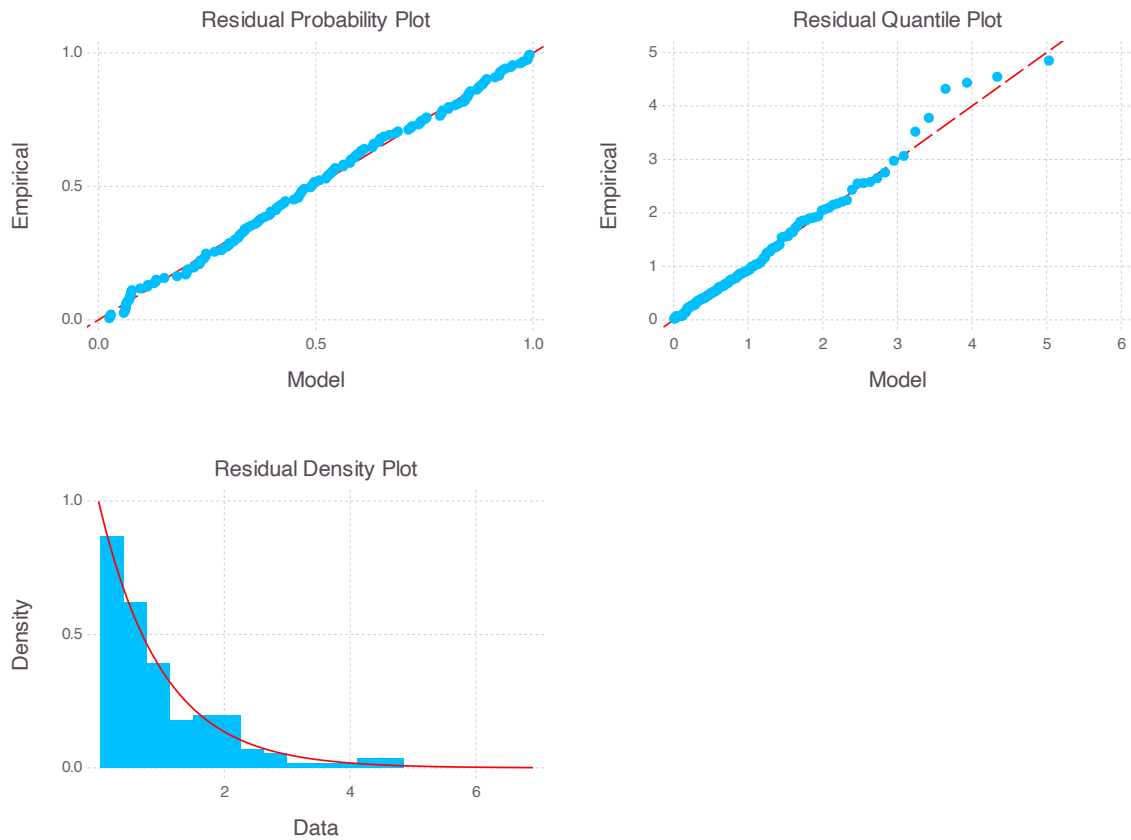


Figure 15: Residual diagnostic plots with a linear trend with the threshold excess model fitted to daily rainfall exceedances data at a location in southwest England.

```
julia> nobs = size(data,1)
julia> nobsperblock = 365
julia> r = returnlevel(fm1, threshold, nobs, nobsperblock, 100)
```

```
ReturnLevel
returnperiod: 100
value: Matrix{Float64}[456000]
```

The corresponding 95% credible interval can be computed using the function `cint`.

```
julia> c = cint(r, 0.95)
```

```
152-element Vector{Vector{Real}}:
 [63.68524916513855, 173.95767630079843]
 [63.68524916513855, 173.95767630079843]
 [63.68524916513855, 173.95767630079843]
 ⋮
 [78.00472366762, 237.2212316732687]
 [76.94872130273704, 238.893208610095]
 [76.94872130273704, 238.893208610095]
```

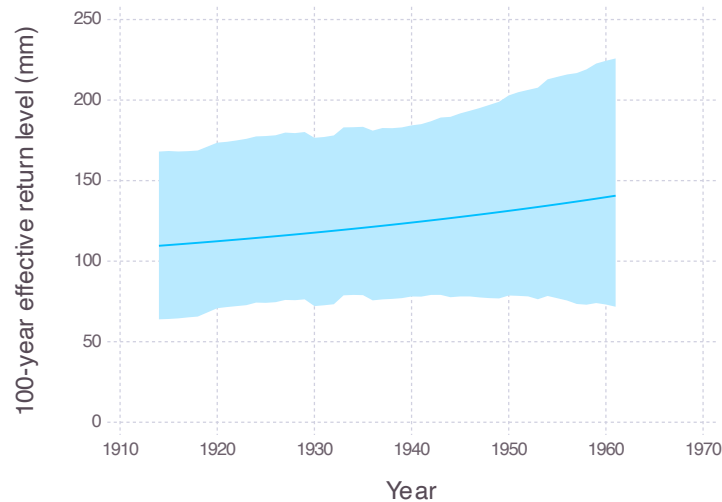


Figure 16: The 100-year effective return level of the daily rainfall in southwest England.

The 100-year effective return levels along with their 95% credible intervals are illustrated in Figure 16.

## 8. Conclusion and future work

The **Extremes.jl** package provides a collection of methods for the analysis of extreme values in Julia. In this article, we have illustrated the use of the **Extremes.jl** package with many examples using datasets from Coles (2001). **Extremes.jl** is designed to be easy to use by researchers, students in courses on extreme values and engineers needing robust and powerful estimations of extremes. This package has a concise and robust API, which exploits Julia’s multidispatch features, and the functionality is tested against the known results by Coles (2001). An often overlooked aspect of Julia is that since the code is entirely in Julia, contributions from graduate students and scientists are easier compared to other high-level languages that need low-level codes for performance reasons.

Future work on this package will include the implementation of the computation of extremal index (e.g., Laurini and Tawn 2003; Ferro and Segers 2003) to measure the degree of clustering of extremes in stationary processes. Moreover, other inference methods, such as point processes or the use of  $r$  order statistics, could be implemented. More direct methods for model selection will potentially be implemented in the next major version. **Extremes.jl** is not intended for multivariate extremes, but the development of other packages for the analysis of multivariate extremes in Julia is encouraged. **Extremes.jl** could potentially be used as a building block for a multivariate extremes package.

## References

- Balkema AA, De Haan L (1974). “Residual Life Time at Great Age.” *The Annals of Probability*, **2**(5), 792–804. doi:10.1214/aop/1176996548.

- Beirlant J, Goegebeur Y, Teugels J, Segers J (2004). *Statistics of Extremes*. John Wiley & Sons. doi:10.1002/0470012382.
- Besançon M, Papamarkou T, Anthoff D, Arslan A, Byrne S, Lin D, Pearson J (2021). “**Distributions.jl**: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem.” *Journal of Statistical Software*, **98**(16), 1–30. doi:10.18637/jss.v098.i16.
- Bezanson J, Edelman A, Karpinski S, Shah VB (2017). “Julia: A Fresh Approach to Numerical Computing.” *SIAM Review*, **59**(1), 65–98. doi:10.1137/141000671.
- Bezanson J, Karpinski S, Shah VB, Edelman A (2012). “Julia: A Fast Dynamic Language for Technical Computing.” *arXiv 1209.5145*, arXiv.org E-Print Archive. doi:10.48550/arxiv:1209.5145.
- Bocharov G (2021). “**pyextremes**: Extreme Value Analysis (EVA) in Python.” URL <https://github.com/georgebv/pyextremes>.
- Coles S (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag. doi:10.1007/978-1-4471-3675-0.
- Davison AC, Smith RL (1990). “Models for Exceedances over High Thresholds.” *Journal of the Royal Statistical Society B*, **52**(3), 393–442. doi:10.1111/j.2517-6161.1990.tb01796.x.
- Edelman A (2019). “A Programming Language to Heal the Planet Together: Julia.” URL [https://www.ted.com/talks/alan\\_edelman\\_a\\_programming\\_language\\_to\\_heal\\_the\\_planet\\_together\\_julia](https://www.ted.com/talks/alan_edelman_a_programming_language_to_heal_the_planet_together_julia).
- Ferro CAT, Segers J (2003). “Inference for Clusters of Extreme Values.” *Journal of the Royal Statistical Society B*, **65**(2), 545–556. doi:10.1111/1467-9868.00401.
- Fisher RA, Tippett LHC (1928). “Limiting Form of the Frequency Distribution of the Largest or Smallest Member of a Sample.” *Mathematical Proceedings of the Cambridge Philosophical Society*, **24**(2), 180–190. doi:10.1017/s0305004100015681.
- Gao F, Han L (2012). “Implementing the Nelder-Mead Simplex Algorithm with Adaptive Parameters.” *Computational Optimization and Applications*, **51**(1), 259–277. doi:10.1007/s10589-010-9329-3.
- Gilleland E, Katz RW (2016). “**extRemes 2.0**: An Extreme Value Analysis Package in R.” *Journal of Statistical Software*, **72**(8), 1–39. doi:10.18637/jss.v072.i08.
- Gilleland E, Ribatet M, Stephenson AG (2013). “A Software Review for Extreme Value Analysis.” *Extremes*, **16**(1), 103–119. doi:10.1007/s10687-012-0155-0.
- Gnedenko B (1943). “Sur La Distribution Limite Du Terme Maximum D’une Série Aléatoire.” *Annals of Mathematics*, pp. 423–453. doi:10.2307/1968974.
- Hoffman MD, Gelman A (2014). “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *Journal of Machine Learning Research*, **15**(47), 1593–1623.
- IPCC (2021). *Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge. doi:10.1017/9781009157896.

- J Hoffmann (2018). “**ExtremeStats**: Extreme Value Statistics in Julia.” URL <https://github.com/JuliaEarth/ExtremeStats.jl>.
- Jalbert J, Genest C, Perreault L (2022). “Interpolation of Precipitation Extremes on a Large Domain Toward IDF Curve Construction at Unmonitored Locations.” *Journal of Agricultural, Biological and Environmental Statistics*, **27**, 461–486. doi:10.1007/s13253-022-00491-5.
- Jones DC, Arthur B, Nagy T, Matriks, Gowda S, Godisemo, Holy T, Noack A, Sengupta A, Darakananda D, B A, Dunning I, Leblanc S, Huijzer R, Fischer K, Chudzicki D, Piibelet M, Mellnik A, Kleinschmidt D, Breloff T, Yu Y, Huchette J, Innes MJ, Inkyu, Verzani J, Pelenitsyn A, Coalson C, O’Mara C, Saba E (2021). **GiovineItalia/Gadfly.jl**: v1.3.4. doi:10.5281/zenodo.5559613.
- Katz RW, Parlange MB, Naveau P (2002). “Statistics of Extremes in Hydrology.” *Advances in Water Resources*, **25**(8), 1287–1304. doi:10.1016/s0309-1708(02)00056-8.
- Lamichaney S (2022). “**ExtremeLy**: Extreme Value Analysis in Python.” URL <https://github.com/SURYA-LAMICHANEY/ExtremeLy>.
- Landwehr JM, Matalas NC, Wallis JR (1979). “Probability Weighted Moments Compared with Some Traditional Techniques in Estimating Gumbel Parameters and Quantiles.” *Water Resources Research*, **15**(5), 1055–1064. doi:10.1029/wr015i005p01055.
- Laurini F, Tawn JA (2003). “New Estimators for the Extremal Index and Other Cluster Characteristics.” *Extremes*, **6**(3), 189–211. doi:10.1023/b:extr.0000031179.49454.90.
- Leadbetter MR, Lindgren G, Rootzén H (1983). *Extremes and Related Properties of Random Sequences and Processes*. Springer-Verlag. doi:10.1007/978-1-4612-5449-2.
- McNeil AJ, Frey R, Embrechts P (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Revised edition. Princeton University Press.
- Mogensen PK, Riseth AN (2018). “**Optim**: A Mathematical Optimization Package for Julia.” *Journal of Open Source Software*, **3**(24). doi:10.21105/joss.00615.
- Neal RM (2011). “MCMC Using Hamiltonian Dynamics.” In S Brooks, A Gelman, G Jones, XL Meng (eds.), *Handbook of Markov Chain Monte Carlo*, chapter 5. Chapman & Hall/CRC. doi:10.1201/b10905.
- Nelder JA, Mead R (1965). “A Simplex Method for Function Minimization.” *The Computer Journal*, **7**(4), 308–313. doi:10.1093/comjnl/7.4.308.
- Northrop PJ, Attalides N (2016). “Posterior Propriety in Bayesian Extreme Value Analyses Using Reference Priors.” *Statistica Sinica*, **26**(2). doi:10.5705/ss.2014.034.
- Paciorek CJ (2023). **climextRemes**: Tools for Analyzing Climate Extremes. R package version 0.3.1, URL <https://CRAN.R-project.org/package=climextRemes>.
- Pfaff B, McNeil A (2018). **evir**: Extreme Values in R. R package version 1.7-4, URL <https://CRAN.R-project.org/package=evir>.

- Pickands J (1975). “Statistical Inference Using Extreme Order Statistics.” *The Annals of Statistics*, **3**(1), 119–131. doi:10.1214/aos/1176343003.
- Python Core Team (2024). *Python: A Dynamic, Open Source Programming Language*. Python Software Foundation. URL <https://www.python.org/>.
- R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Smith BJ (2014). *Mamba: Markov Chain Monte Carlo for Bayesian Analysis in Julia*. URL <https://github.com/brian-j-smith/Mamba.jl>.
- Southworth H, Heffernan JE, Metcalfe PD (2024). *texmex: Statistical Modelling of Extreme Values*. R package version 2.4.9, URL <https://CRAN.R-project.org/package=texmex>.
- The Math Works Inc (2022). *MATLAB: Version 7.10.0 (R2022b)*. Natick, Massachusetts. URL <https://www.mathworks.com/>.
- Zappa Nardelli F, Belyakova J, Pelenitsyn A, Chung B, Bezanson J, Vitek J (2018). “Julia Subtyping: A Rational Reconstruction.” *Proceedings of the ACM on Programming Languages*, **2**(OOPSLA). doi:10.1145/3276483.

**Affiliation:**

Jonathan Jalbert  
Department of Mathematics and Industrial Engineering  
Polytechnique Montréal  
Montréal, QC, Canada  
E-mail: [jonathan.jalbert@polymtl.ca](mailto:jonathan.jalbert@polymtl.ca)  
URL: <https://www.polymtl.ca/expertises/jalbert-jonathan>

Marilou Farmer  
Computer Engineering and Software Engineering  
Polytechnique Montréal  
Montréal, QC, Canada  
*and*  
Clinia  
Montréal, QC, Canada

Gabriel Gobeil  
Department of Mathematics and Industrial Engineering  
Polytechnique Montréal  
Montréal, QC, Canada  
*and*  
Environment and Climate Change Canada  
Québec City, QC, Canada

Philippe Roy  
Institut de Recherche d'Hydro-Québec  
Varenes, QC, Canada