GAM model for EGPD distributions

Noémie Le Carrer, Carlo Gaetan *

Dipartimento di Scienze Ambientali, Informatica e Statistica
Università Ca' Foscari di Venezia, Venice, Italy

January 31, 2022

Abstract

In this document, we present a code making use of the R-package `GAMLSS`, to model non-stationary fields whose margins are EGPD distributed, in the sense of Naveau et al. [4]. The `GAMLSS` package provides univariate distributional regression models, where parameters of the assumed distribution for the response can be modeled as additive functions of the explanatory variables. It is flexible enough to allow the encoding of novel families of distribution, as long as their density, cumulative, quantile and random generative function and, optionally, first and second (cross-) derivatives of the likelihood can be computed. We seize this potential to implement the parametric EGPD family in a generic way, allowing to test any new parametric form, up to 4 parameters so far, satisfying the requirement of the EGPD family.

1 Introduction

We consider a field of space-time data, $X(\boldsymbol{s}, t)$ whose marginal distribution can be modelled by means of a parametric EGPD distribution. The EGPD family [4, 9] accounts simulatenously and in a parsimonous way for both extreme and non extreme values. Its cumulative distribution function (CDF) reads:

$$F(x) = G\{H_\mu(x/\sigma)\}\,,\ \forall x > 0 \tag{1}$$

where H_μ represents the CDF of the Generalized Pareto distribution (GPD) of shape and scale parameters μ and σ, namely:

$$H_\mu(z) = \begin{cases} 1 - (1+\mu z)_+^{-1/\mu} & \text{for } \mu \neq 0 \\ 1 - \exp\{-z\} & \text{for } \mu = 0 \end{cases} \tag{2}$$

where the values x are scaled into $z = x/\sigma$ and $a_+ = \max(a, 0)$. The shape parameter μ is assumed to be non-negative (common use e.g. for daily rainfall [2]). In our add-on to the `GAMLSS` package, we implemented a more generic version where μ can take any value.

G is a continuous CDF on the unit interval. It is constrained by several assumptions: 1) we want to ensure that the upper-tail behavior of F follows a GPD; 2) the low rainfalls, seen as the upper tail of $-X$ when $x \to 0$, follows a GPD as well.

This leads to the following consequences:

1. the upper-tail behavior of $1 - F(x)$ is equivalent to the original GPD tail used to build $F(x)$;

*We thank Nicolas Berthier for helping us to develop some parts of the R code.

2. when $x \to 0$, $F(x) \sim \frac{c}{\sigma^s} x^s$ where $c = \lim_{x \to 0} \frac{G(u)}{u^s}$ is positive and finite for some real s.

Four parametric families that satisfies the above-mentioned constraints are proposed in Naveau et al. [4] for $G(v)$, where $v \in [0, 1]$:

1. $G(v) = v^\nu$, $\nu > 0$
2. $G(v) = pv^{\nu_1} + (1-p)v^{\nu_2}$, $\nu_2 \geq \nu_1 > 0$ and $p \in [0, 1]$
3. $G(v) = 1 - Q_\delta\{(1-v)^\delta\}$, $\delta > 0$ where Q_δ is the CDF of a beta random variable with parameters $1/\delta$ and 2, that is: $Q_\delta(v) = \frac{1+\delta}{\delta} v^{1/\delta}\left(1 - \frac{v}{1+\delta}\right)$
4. $G(v) = [1 - Q_\delta\{(1-v)^\delta\}]^{\nu/2}$, $\nu, \delta > 0$

In model 1, μ controls the rate of upper tail decay, ν the shape of the lower tail and σ is a scale parameter. $\nu = 1$ allows us to recover the GPD model, while varying it gives flexibility in the description of the lower tail. Model 2, as a mixture of power laws, allows to increase the flexibility provided by model 1. Therein, ν_1 controls the lower tail behavior while ν_2 modifies the shape of the density in its central part. Model 3 is connected to the work of Falk et al. [3]. μ keeps controlling the upper extreme tail, δ the central part of the distribution in a threshold tuning way. However, this model imposes a behavior type X^2 for the lower tail (null density at 0) instead of allowing the data to constrain it. Hence model 4, which adds an extra parameter ν to circumvent this limitation. In this last model, ν, δ, μ control respectively the lower, moderate and upper parts of the distribution. By construction, the lower and upper tails are GPD of shape parameters ν and μ respectively.

We implemented these four families in R and used in particular the potential of symbolic derivation (by means of the package `Deriv`) for remaining flexible and allowing new definitions of parametric families in the future. The inverse function $G^{-1}(u)$ is computed numerically (as we want to remain general and open to new definitions for G). To simulate from Equation (1), the user randomly draws a uniform variable U in $[0, 1]$ and then applies the quantile function $X = F^{-1}(U)$, given for $p \in [0, 1]$ by:

$$x_p = F^{-1}(p) = \begin{cases} \frac{\sigma}{\mu}[\{1 - G^{-1}(p)\}^{-\mu} - 1] & \text{for } \mu \neq 0 \\ -\sigma \log\{1 - G^{-1}(p)\} & \text{for } \mu = 0 \end{cases} \tag{3}$$

2 Implementing the EGPD family with GAMLSS

We first use the following set of commands to load the `GAMLSS` package [5], the generic implementation of EGPD provided in `GenericEGPDc.R` and create the desired parametric model of EGPD by providing G as a function of z and at maximum two parameters ν and τ (so far, due to limitations of the GAMLSS implementation). In the following, we use as an illustration the Model 1 from Naveau et al. [4], $G(z) = z^\nu, \nu > 0$ (EGPD1).

```
library(gamlss)

source ("GenericEGPDc.R")

EGPD1Family <- MakeEGPD (function (z,nu) z^nu, Gname = "Model1")

EGPD1Model() # Instantiate the family object
```

Given parameters μ_0, σ_0 and ν_0, we can directly use the density function (PDF), CDF, quantile function (inverse CDF) and EGPD1-distributed random number generation function, respectively:

```
dEGPDModel1(x,mu=mu0,sigma=sigma0,nu=nu0)

pEGPDModel1(x,mu=mu0,sigma=sigma0,nu=nu0)

```

```
qEGPDModel1(u,mu=mu0,sigma=sigma0,nu=nu0)

rEGPDModel1(n,mu=mu0,sigma=sigma0,nu=nu0)
```

To change the link functions $\eta_i = g_i(\theta_i)$ by means of which each EGPD parameter θ_i (here μ, σ or ν) is indirectly optimised, or the default starting values for the parameters to fit, or to print these links, we use the commands:

```
EGPD1Family(mu.link="identity",mu.init=1, # log , inverse, own , ...
            sigma.link="log",sigma.init=3,
            nu.link="log",nu.init=0.5)

show.link(EGPD1Family) #not working at the moment
```

It is possible to create original link functions "own", by gathering the expressions of the link function η_i, its inverse, the derivative of the latter w.r.t. parameter θ_i and its area of validity. For instance to create a log-link function shifted to the right for μ [1]:

```
own.linkfun <- function (mu) {
    .shift = 0.0001
    log (mu - .shift) }
own.linkinv <- function (eta) {
    shift = 0.0001
    thresh <- - log (.Machine$double.eps)
    eta <- pmin (thresh, pmax(eta, -thresh))
    exp (eta) + shift
}
own.mu.eta <- function (eta) {
    shift = 0.0001
    pmax (exp (eta), .Machine$double.eps)
}
own.valideta <- function (eta) TRUE

EGPD1Family(mu.link="own")
```

3 Simulation and fitting of stationary EGPD marginal distributions

We now perform a synthetic experiment where we first simulate a field $X(\boldsymbol{s}, t)$ ("x") over $N_{stat} = 10$ stations identified by their 2-dimensional coordinates ("latitude","longitude") in a square of size 10×10 and over 20 years. The time index is considered cyclic, taking value from 1 to 365 (day of the year omitting lap years) and noted "cyc". In a first step, the field is marginally distributed according to the EGPD1 family, with constant parameters $\mu_0 = 0.1, \sigma_0 = 1, \nu_0 = 2$. We use the random generation function rEGPD1Family described above. Figure 1 shows the CDF and PDF empirically observed and their theoretical counterparts, obtained by using directly the CDF and PDF, pEGPD1Family and dEGPD1Family respectively.

```
library(pracma)

mu0 <- 0.1
sigma0 <- 1
nu0 <- 2
Ny <- 20
N <- Ny*365
Nstat <- 10
db <- data.frame(matrix(ncol = 4, nrow = N*Nstat))
colnames(db) <- c("longitude","latitude","x","cyc")
Lat=runif(10,min=0,max=10)
Long=runif(10,min=0,max=10)
```

[1] More on link function definition p. 179 of Stasinopoulos et al. [8])

```
time=1:365
for (j in 1:Nstat){
    p <- runif(N)
    pr.Sim <- qEGPDModel1(p,mu=mu0,sigma=sigma0,nu=nu0)
    db$latitude[((j-1)*N+1):(j*N)] <- repmat(Lat[j],N,1)
    db$longitude[((j-1)*N+1):(j*N)] <- repmat(Long[j],N,1)
    db$cyc[((j-1)*N+1):(j*N)] <- repmat(time,1,Ny)
    db$x[((j-1)*N+1):(j*N)] <-pr.Sim
}

Fn <- ecdf(db$x)
q <- 0.1*(1:500)
simCDF <- numeric(length(q))
simPDF <- numeric(length(q))
for (i in 1:length(q)) {
    simCDF[i] <- pEGPDModel1(q[i],mu=mu0,nu=nu0,sigma=sigma0)
    simPDF[i] <- dEGPDModel1(q[i],mu=mu0,nu=nu0,sigma=sigma0,log=FALSE)
}
```

Fitting the parameters in the constant hypothesis (which in terms of regression reads $x \sim 1$, stationarity in both space and time) is performed by using the `gamlss()` function. `predict()` and `fitted` allow to extract the fitted parameters. The former can also be used to predict the response variable distribution parameters for old or new data values of the covariates.

```
con <- gamlss.control (n.cyc = 100,
                       mu.step = 0.1, sigma.step = 0.1, nu.step = 0.1,
                       tau.step = 0.1, trace = FALSE, autostep=TRUE)

simfit.egpd.0 <- gamlss(x ~ 1, data = db, family = EGPD1Family(mu.link = "identity"),
                        control = con,
                        mu.start=1, sigma.start=1, nu.start=1,
                        method=CG())

muFit <- fitted(simfit.egpd.0,"mu")[1]
sigmaFit <- predict(simfit.egpd.0,what="sigma", type="response")[[1]]
nuFit <- predict(simfit.egpd.0,what="nu", type="response")[[1]]

print(muFit)
0.09980239
print(sigmaFit)
[1] 1.000595
print(nuFit)
[1] 2.000454
```

Through the control option "con" we provide starting values (all 1) for the parameters, we limit the number of cycles to 100, we put the default step-size to 0.1 for all parameters and we allow (autostep=TRUE, which is the default in `GAMLSS`) the steps to be halved automatically if the new global deviance is greater that the old one. Maximum (penalized) likelihood estimation is used to fit the models and a Newton-Raphson/Fisher scoring algorithm is used to maximize the (penalized) likelihood. Two different approaches are implemented in `GAMLSS`. The `RS()` algorithm is a generalization of the algorithm used by Rigby and Stasinopoulos [6, 7] for fitting mean and dispersion in additive models. and does not require the cross derivatives of the loglikelihood. The `CG()` algorithm is a generalization of the Cole and Green's algorithm [1], and requires the knowledge of the first and (expected or pproximated) second and cross derivatives of the log-likelihood function with respect to the distribution parameters. `mixed(a,b)` is a combination of a first iterations with `RS()` followed by $b - a$ iterations with `CG()`.

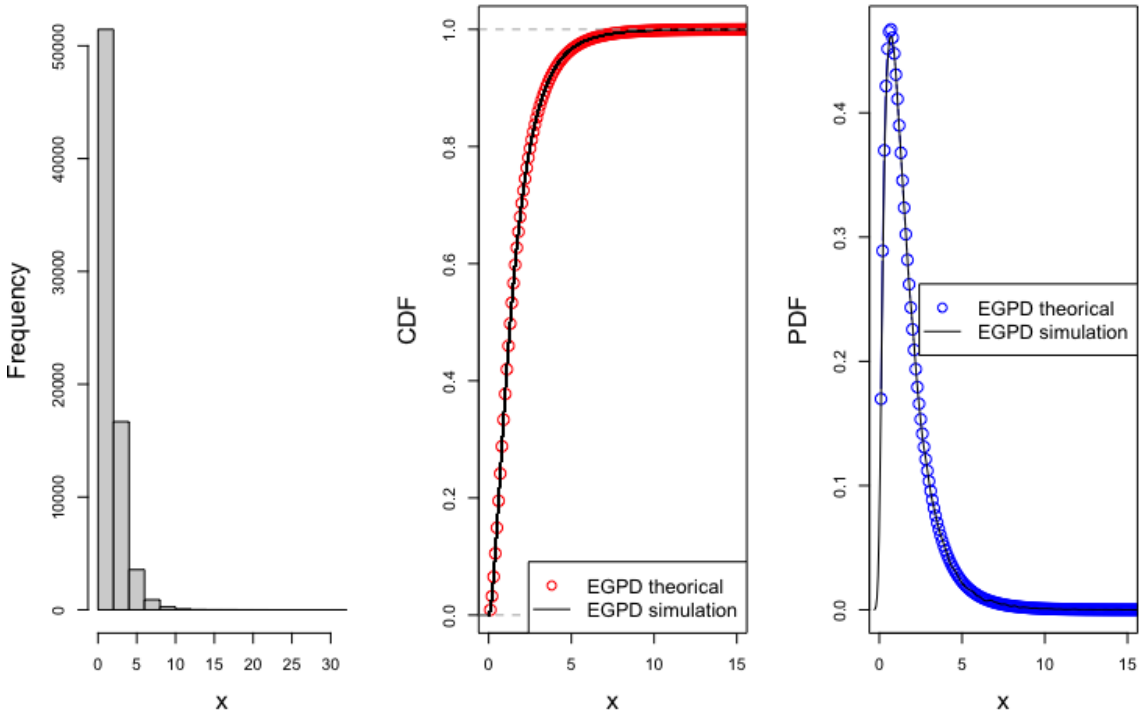

Figure 1: Empirical distributions of the simulations of EGPD1 and associated theoretical curves.

4 Simulation and fitting of non-stationary EGPD marginal distributions

In practice, the field $X(\boldsymbol{s}, t)$ often has margins whose parameters vary in space and time (e.g. due to seasonality or relief variations). Modelling such field consists in regressing the parameters of the marginal distributions as additive functions of possible explanatory variables, here space $\boldsymbol{s}$ and time or a time-index capturing seasonality. In `GAMLSS`, standing for Generalized Additive Models for Location, Scale and Shape, the linear predictor becomes a sum of smooth parametric functions of the covariates. Such a representation is particularly flexible and allows to capture complex dependence patterns between distribution parameters and explanatory variables, along with uncertainty estimation. Conversely, it is less powerful than a purely stochastic modeling of the parameters variations (e.g. considering each EGPD parameter as a random field) for extrapolation beyond the borders of the training space, due to the limitations of spline modelling *per se*.

Let $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)$ the vector of p parameters, $\boldsymbol{y_{k_j}}$ the covariates for parameter θ_k, η_{θ_k} monotonic link functions and h_j unknown functions, different for each parameter θ_k and parametric or not. The general model of `GAMLSS` then reads:

$$\eta_{\theta_k}(\boldsymbol{y}_k) = \beta_0 + h_1(y_{k_1}) + +h_2(y_{k_2}) + h_3(y_{k_3}, y_{k_4}) + ..., \text{ for } k = 1, ..., p \tag{4}$$

which in our case boils down to:

$$\eta_{\theta_k}(\boldsymbol{s}, t) = h_2(\boldsymbol{s}) + h_2(t) \tag{5}$$

where θ_k represents any parameter of the EGPD model. The unknown h_j are expressed as sums of known basis functions: $h_j(y_k) = \sum_{i=1}^{d_j} \beta_{ji} b_{ji}(y_k)$ where the β_{ji} are the coefficients to be estimated and d_j the dimension of the basis [10]. These smoothers h_j, or splines, can be of different types, thoroughly presented e.g. in Stasinopoulos et al. [8].

In practice, we first reproduce the previous experiment yet we add a linear trend to the parameter μ, setting *a priori* $\mu_0 = \mu(latitude = 0)$ and $\mu_{10} = \mu(latitude = 10)$. We then fit the EGPD-based model, this time assuming a linear dependence between μ and Lat.

```
mu0 <- 0.1
mu10 <- 0.5

db <- data.frame(matrix(ncol = 4, nrow = N*Nstat))
colnames(db) <- c("longitude","latitude","x","cyc")
Lat=runif(10,min=0,max=10)
Long=runif(10,min=0,max=10)
time=1:365
for (j in 1:Nstat){
    p <- runif(N)
    pr.Sim <- numeric(N)
    mu_lat <- ((mu10-mu0)/10)*Lat[j]+mu0
    for (i in 1:N) {
        pr.Sim[i] <- qEGPDModel1(p[i],mu=mu_lat,sigma=sigma0,nu=nu0)
    }
    db$latitude[((j-1)*N+1):(j*N)] <- repmat(Lat[j],N,1)
    db$longitude[((j-1)*N+1):(j*N)] <- repmat(Long[j],N,1)
    db$cyc[((j-1)*N+1):(j*N)] <- repmat(time,1,Ny)
    db$x[((j-1)*N+1):(j*N)] <-pr.Sim
}

# Fit parameters (here only for mu, others are considered constant)
mod.egpd.mu.lat <- gamlss(x~latitude,
                          data=db,
                          family = EGPD1Family,
                          start.from = mod.egpd.0,
                          method=CG())

```

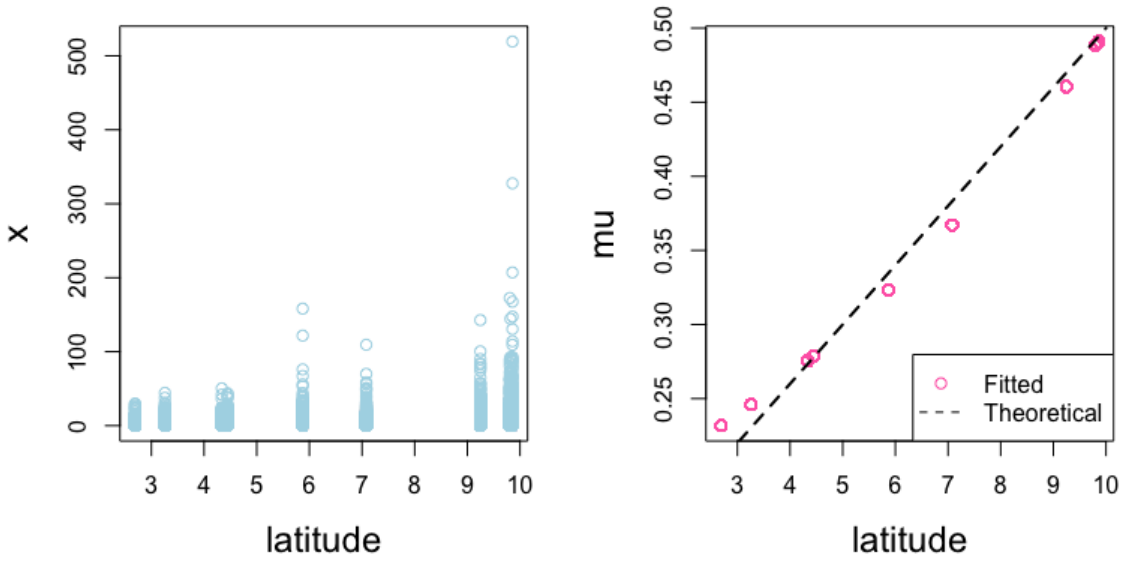

Figure 2: Simulated values of the field (left) and fitted and theoretical parameter μ as a function of the covariate "latitude".

```
muFit <- fitted(mod.egpd.mu.lat,"mu")
sigmaFit <- predict(mod.egpd.mu.lat,what="sigma", type="response")[[1]]
print(sigmaFit)
[1] 0.9932371
> nuFit <- predict(mod.egpd.mu.lat,what="nu", type="response")[[1]]
print(nuFit)
[1] 2.00237

# See fitted parameters as functions of covariates
fittedPlot(mod.egpd.mu.lat, x=db$latitude, line.type = TRUE,xlab="latitude")
```

Again, we compare the fitted theoretical PDF and CDF (using the fitted parameters) in a given station to their empirical (from simulation) counterparts. Results are shown in Figures 2 and 3 .

```
Lat_s=Lat[5]
Fn <- ecdf(db$x[db$latitude==Lat_s])
# Get the fitted mu for that station
muPred <- mean(muFit[db$latitude==Lat_s])
q <- 0.1*(1:500)
simCDF <- numeric(length(q))
simPDF <- numeric(length(q))
for (i in 1:length(q)) {
    simCDF[i] <- pEGPDModel1(q[i],mu=muPred,nu=nu0,sigma=sigma0)
    simPDF[i] <- dEGPDModel1(q[i],mu=muPred,nu=nu0,sigma=sigma0,log=FALSE)
}
```

We can compare the EGPD-based model for instance to a model with classical Gamma margins, by means of the function `GAIC()`, allowing to extract the Akaike information criterion (AIC) or the Bayesian Information Criterion (BIC):

```
mod.ga.mu.lat <- gamlss(x~latitude,
                        data=db, family=GA)
```

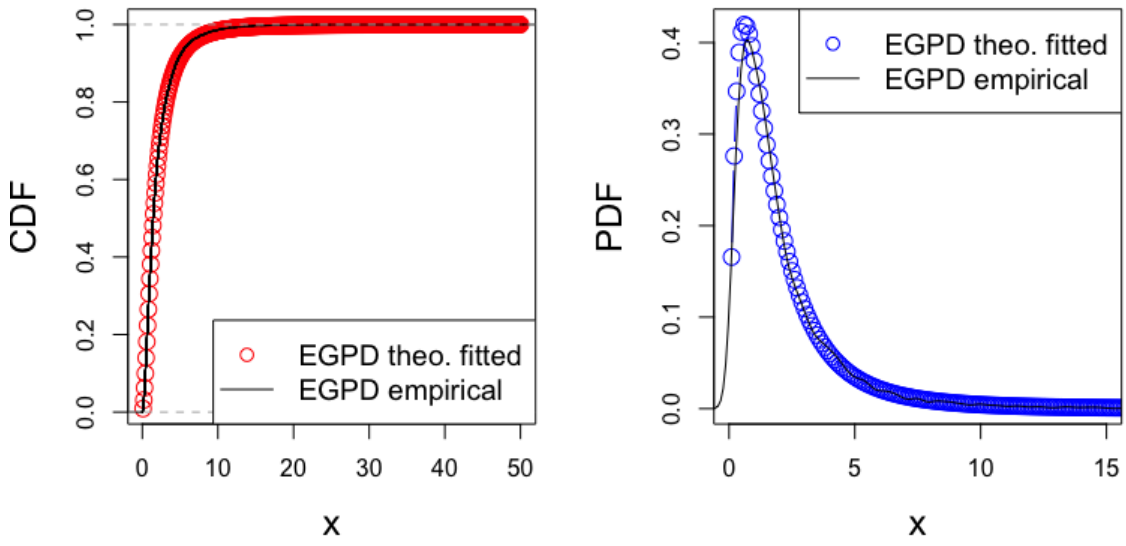

Figure 3: CDF and PDF at a given station, either empirically estimated from simulations or theoretically drawn with the fitted parameters.

```
# AIC
> GAIC(mod.ga.mu.lat,mod.egpd.mu.lat)
                  df       AIC
mod.egpd.mu.lat   4 269891.7
mod.ga.mu.lat     3 282664.0
> # BIC
> GAIC(mod.ga.mu.lat,mod.egpd.mu.lat,k=log(dim(db)[1]))
                  df       AIC
mod.egpd.mu.lat   4 269928.5
mod.ga.mu.lat     3 282691.6
```

Finally, we reproduce the last experiment but we add a cyclic trend in "cyc", the covariate representing the day of the year, which reads for a latitude l and a day of the year c:

$$\mu(l,c) = \big(l(\mu_{10}-\mu_0)/10) + \mu_0\big)\Big(1 + 0.4\sin\frac{2\pi c}{200}\Big) \tag{6}$$

```
mu0 <- 0.1
mu10 <- 0.5
sigma0 <- 1
nu0 <- 2

Ny <- 20
time=1:365
N <- Ny*length(time)
Nstat <- 10
db <- data.frame(matrix(ncol = 4, nrow = N*Nstat))
colnames(db) <- c("longitude","latitude","x","cyc")
Lat=runif(10,min=0,max=10)
Long=runif(10,min=0,max=10)
for (j in 1:Nstat){
```

```
    mu_lat <- ((mu10-mu0)/10)*Lat[j]+mu0
    mu_lat_year <- mu_lat * (1+ 0.4* sin(2*time*pi/200))
    mu_lat_allyears <- repmat(mu_lat_year,1,Ny)
    p <- runif(N)
    pr.Sim <- numeric(N)
    for (i in 1:N) {
        pr.Sim[i] <- qEGPDModel1(p[i],mu=mu_lat_allyears[i],sigma=sigma0,nu=nu0)
    }
    db$latitude[((j-1)*N+1):(j*N)] <- repmat(Lat[j],N,1)
    db$longitude[((j-1)*N+1):(j*N)] <- repmat(Long[j],N,1)
    db$cyc[((j-1)*N+1):(j*N)] <- repmat(time,1,Ny)
    db$x[((j-1)*N+1):(j*N)] <-pr.Sim
}
```

We can now fit a model with EGPD marginals to this dataset by means of `gamlss()`. This time, the relationship between μ and the covariates of X is not linear, so we resort to generalized additive models (GAM). The GAM form allows more flexible modelling of the relationship between the distribution parameters and continuous explanatory variables. Within `GAMLSS`, a range of nonparametric smoothing functions are implemented, among which the `pb()` (P-splines), `cs()` (cubic splines), `lo()` (locally weighted regression) and `nn()` (neural networks) functions. We report the reader to the Chapter 9 of Stasinopoulos et al. [8] for a full development. Here we use the cyclic version of the smoothing function `pb()`, namely `pbc()`, imposing that the initial and last values of the parameter over the covariate range are equal. Note that for pb-type smoothing functions, the smoothing parameter (and consequently the effective degrees of freedom) are estimated automatically using the default local maximum likelihood method. Figures 4, 5, 6 and 7 show how to extract and display the results of the fitting.

```
mod.egpd.mu.lat.cyc <- gamlss(x~latitude+pbc(cyc),
                          data=db,
                          family = EGPD1Family,
                          start.from = mod.egpd.mu.lat,
                          method=CG())

> edf(mod.egpd.mu.lat.cyc, "mu")
Effective df for mu model
$`pbc(cyc)`
[1] 12.93309

> drop1(mod.egpd.mu.lat.cyc)
Single term deletions for mu
Model:
x ~ latitude + pbc(cyc)
               Df    AIC    LRT   Pr(Chi)
<none>            265192
latitude  0.78938 268055 2864.9 < 2.2e-16 ***
pbc(cyc) 11.93309 266218 1049.8 < 2.2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1

> summary(mod.egpd.mu.lat.cyc)
```

Information about the model, significance of parameters, among others, are obtained by means of the function `summary()`. However when fitting a smooth nonparametric term in `GAMLSS`, the displayed coefficient of the smoothing term and its standard error (s.e.) only refer to the *linear component* of the term. Approximate signifcance of the smoothing terms is better obtained using the function `drop1()`, but this may be slow for large datasets and many smoothing terms. `term.plot()` allows to visualise the whole smoothing function and assess the effect of each covariate on a given parameter.

Finally, distributions up to four parameters (so far in this generic code due to limitations in `GAMLSS`) can be fitted in a GAM form. The call for regressing other parameters beyond μ reads:

```
mod.egpd.all.lat.cyc <- gamlss(x~latitude+pbc(cyc),
                                    sigma.fo=~latitude,
```

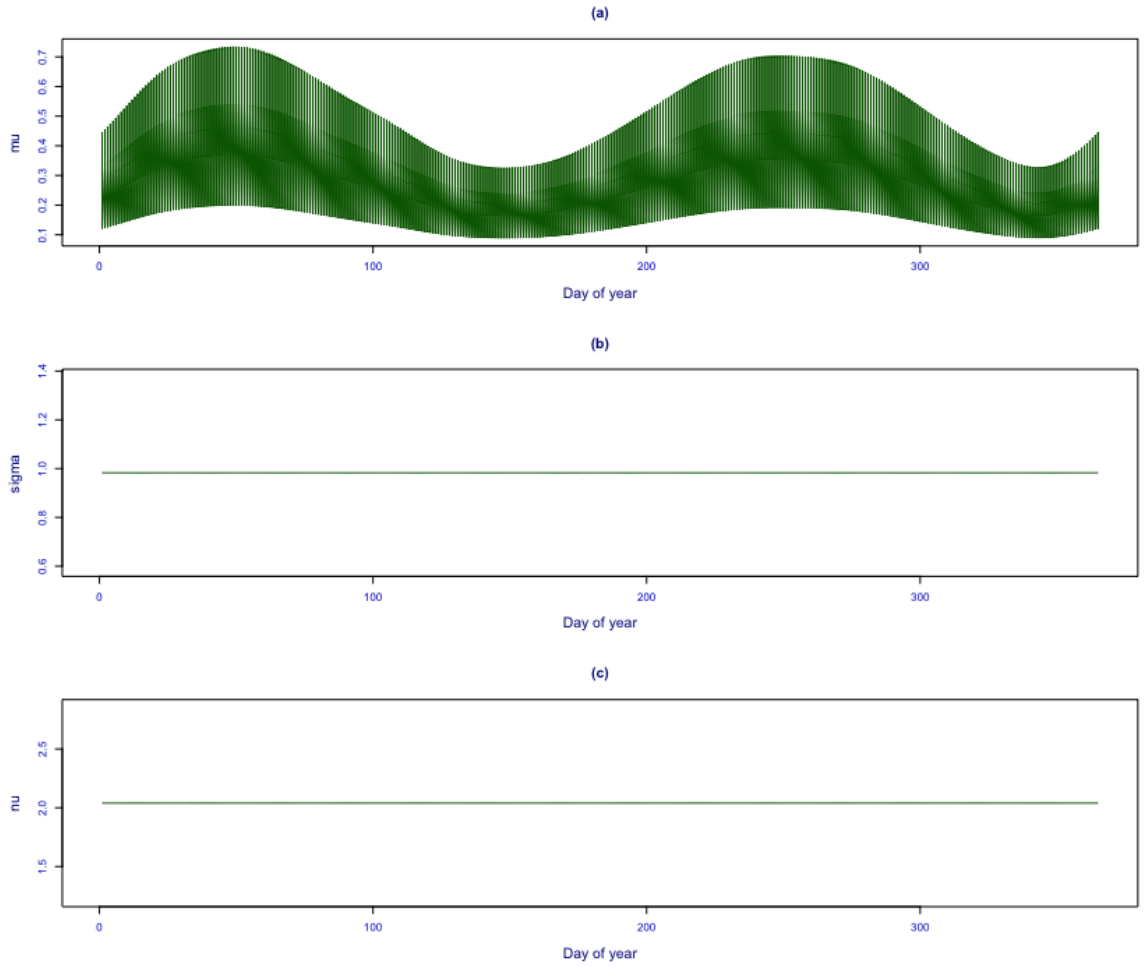

Figure 4: Fitted parameters μ, σ, ν versus the covariate "cyc".

```
                                    nu.fo=~cs(latitude),
                                data=db,
                                family = EGPD1Family,
                                start.from = mod.egpd.mu.lat,
                                method=CG())
```

To obtain the effective degrees of freedom for all the distribution parameters we use `edfAll()`. When it comes to diagnostic plots, useful commands include `plot()` and `wp()` that plot 1) the normalized quantile residuals against fitted values and against index, the density estimate vs quantile residuals and normalized QQ plots and 2) worm plots respectively.

References

[1] Timothy J Cole and Pamela J Green. Smoothing reference centile curves: the lms method and penalized likelihood. *Statistics in medicine*, 11(10):1305–1319, 1992.

[2] Guillaume Evin, Anne-Catherine Favre, and Benoit Hingray. Stochastic generation of multi-site daily precipitation focusing on extreme events. *Hydrology and Earth System Sciences*, 22(1):655–672, 2018.

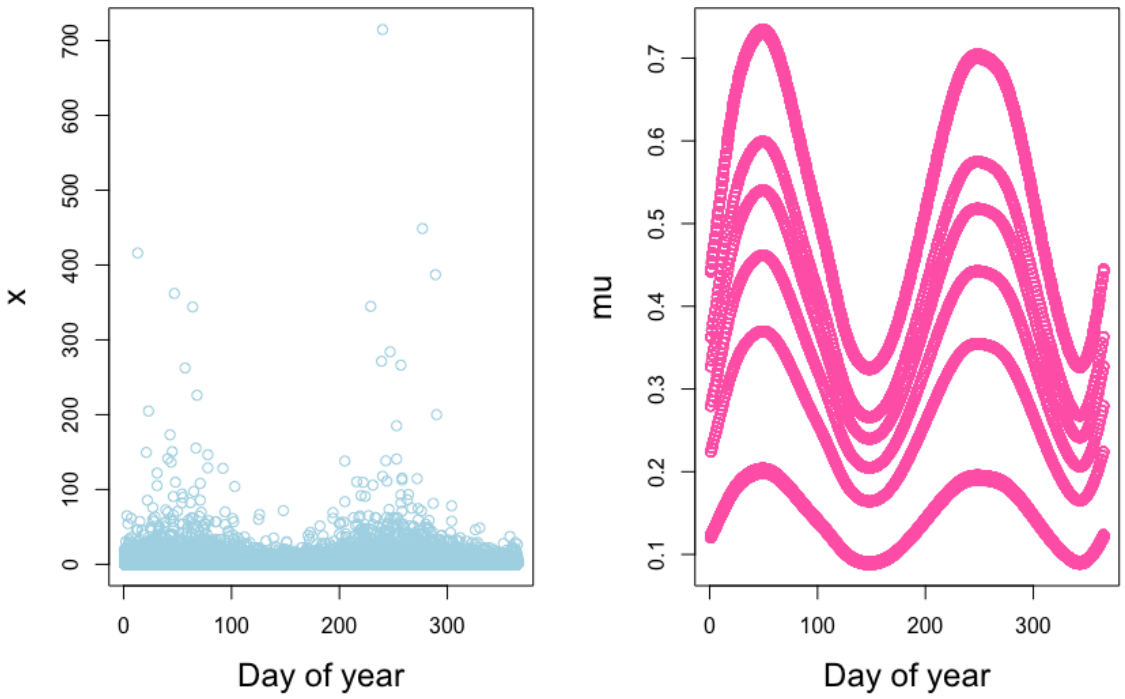

Figure 5: Field values (left) and fitted parameter μ (right) versus the covariate "cyc". There are 10 stations but two groups each of 3 stations are extremely close in latitude ($0.3, 0.4, 0.5$ and $9.70, 9.68, 9.73$), explaining the presence of only 6 curves, corresponding to the different levels of latitude -since μ is linearly increasing with the latitude.

[3] Michael Falk, Jürg Hüsler, and Rolf-Dieter Reiss. *Laws of Small Numbers: Extremes and Rare Events*. Springer, Basel, 2010.

[4] Philippe Naveau, Raphaël Huser, Pierre Ribereau, and Alexis Hannart. Modeling jointly low, moderate, and heavy rainfall intensities without a threshold selection. *Water Resources Research*, 52(4):2753–2769, 2016.

[5] R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape,(with discussion). *Applied Statistics*, 54:507–554, 2005.

[6] Robert A Rigby and DM Stasinopoulos. A semi-parametric additive model for variance heterogeneity. *Statistics and Computing*, 6(1):57–65, 1996.

[7] Robert A Rigby and Mikis D Stasinopoulos. Mean and dispersion additive models. In *Statistical theory and computational aspects of smoothing*, pages 215–230. Springer, 1996.

[8] Mikis D Stasinopoulos, Robert A Rigby, Gillian Z Heller, Vlasios Voudouris, and Fernanda De Bastiani. *Flexible regression and smoothing: using GAMLSS in R*. CRC Press, 2017.

[9] Patricia Tencaliec, A-C Favre, Philippe Naveau, Clémentine Prieur, and Gilles Nicolet. Flexible semiparametric generalized pareto modeling of the entire range of rainfall amount. *Environmetrics*, 31(2):e2582, 2020.

[10] Simon N Wood. *Generalized additive models: an introduction with R*. CRC press, 2017.

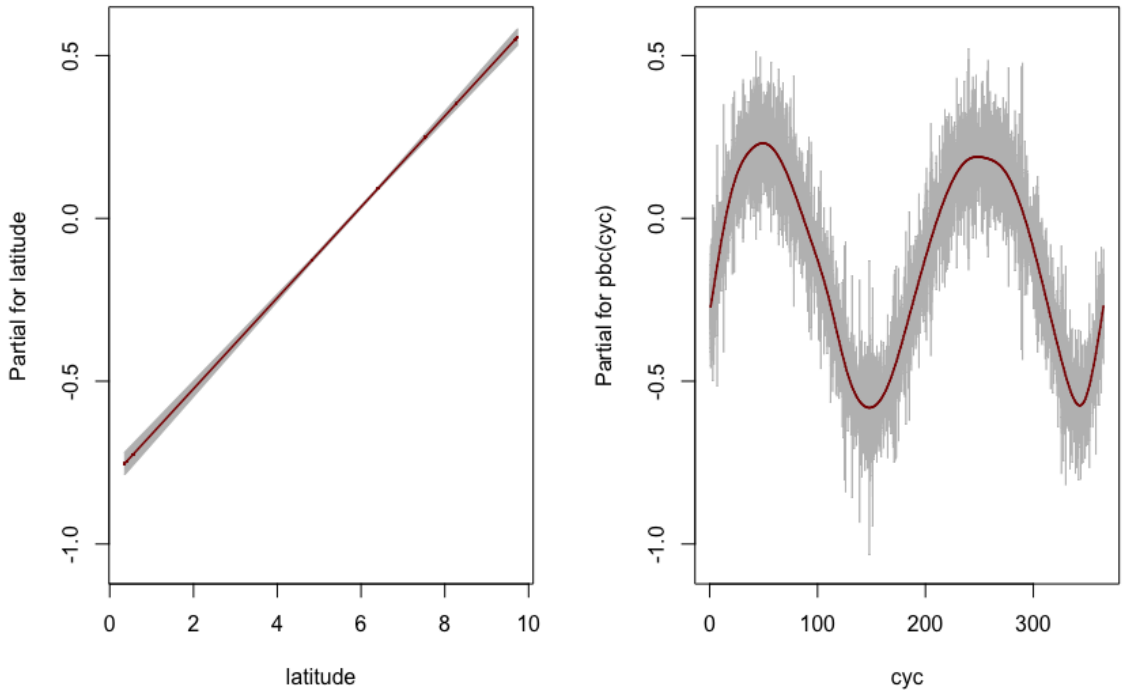

Figure 6: Smoothing functions versus the covariates, obtained by means of the function `term.plot()`.

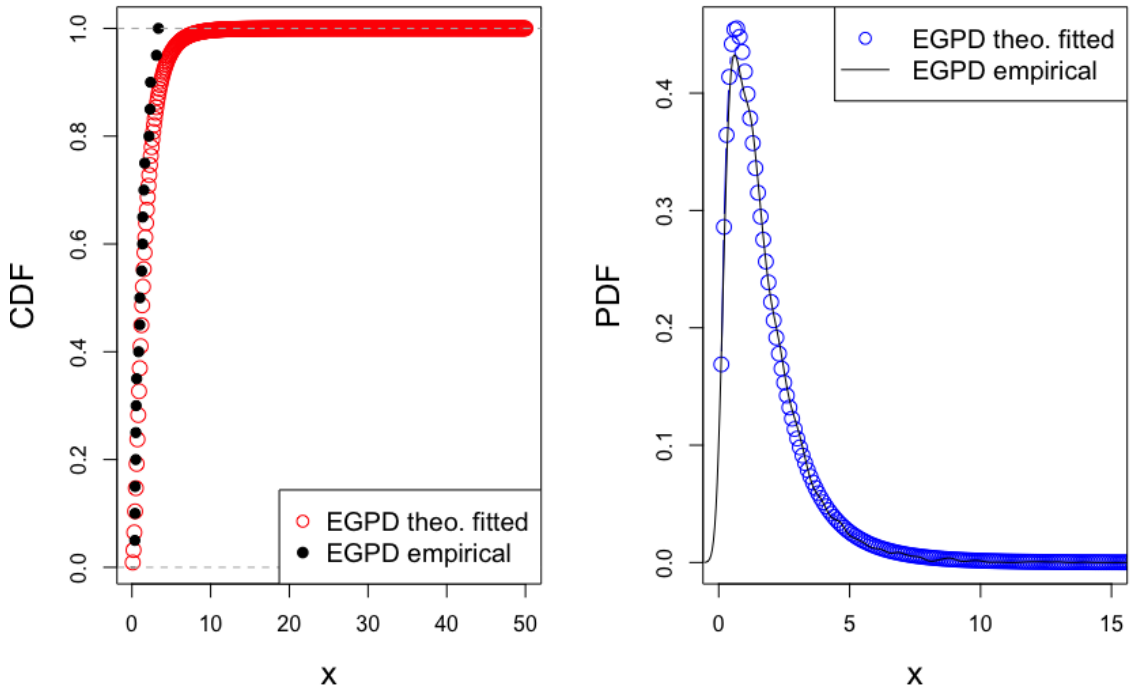

Figure 7: CDF and PDF at a given station and time of the year, either empirically estimated from simulations or theoretically drawn with the fitted parameters. The sample set size for the empirical distributions is N_y=20 (years) hence the steps in the plotted CDF.