Journal of Statistical Software

March 2015, Volume 64, Issue 6. *http://www.jstatsoft.org/*

NHPoisson: An R Package for Fitting and Validating Nonhomogeneous Poisson Processes

Ana C. Cebrián
University of Zaragoza

Jesús Abaurrea
University of Zaragoza

Jesús Asín
University of Zaragoza

Abstract

NHPoisson is an R package for the modeling of nonhomogeneous Poisson processes in one dimension. It includes functions for data preparation, maximum likelihood estimation, covariate selection and inference based on asymptotic distributions and simulation methods. It also provides specific methods for the estimation of Poisson processes resulting from a peak over threshold approach. In addition, the package supports a wide range of model validation tools and functions for generating nonhomogenous Poisson process trajectories. This paper is a description of the package and aims to help those interested in modeling data using nonhomogeneous Poisson processes.

Keywords: nonhomogeneous Poisson process, peak over threshold, validation analysis, covariate selection.

1. Introduction

Despite the large number of potential applications of Poisson processes, the scientific community in many fields has not yet taken full advantage of Poisson process modeling, particularly in a nonstationary framework. Most of the published applications are restricted to the environmental field, see the review of extreme event studies on hydrology by Katz, Parlange, and Naveau (2002). Ogata (1988) used Poisson processes to model earthquakes, and Madsen, Rasmussen, and Rosbjerg (1997) and Abaurrea and Cebrián (2002) modeled hydrological and meteorological drought occurrences, respectively. The statistical techniques for modeling and fitting this type of processes are well-known, see Embrechts, Klüppelberg, and Mikosch (1997), Coles (2001), or Kutoyants (1998) who discuss the estimation theory for nonhomogeneous Poisson processes. However, the implementation of these techniques is a time-consuming and non trivial task and the tools for validating nonhomogeneous Poisson processes are not as well developed as for other statistical models. Accordingly, an easy-

to-use software, which provides simple and fast algorithms that facilitate the modeling and validation of nonhomogeneous Poisson processes, can be very useful.

NHPoisson (Cebrian 2015) is an R (R Core Team 2015) package which provides an assembly of tools for all the steps involved in data modeling using this type of processes. In particular, it supplies functions to model the Poisson intensity as a function of covariates, including the estimation of Poisson processes resulting from a peak over threshold approach. It also contains a wide toolkit for model selection and validation analysis, including several types of residuals and diagnostic techniques adapted from other statistical models. **NHPoisson** is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=NHPoisson`.

The outline of the paper is as follows. Section 2 introduces Poisson processes and the peak over threshold approach. Some R packages related to this topic are reviewed in Section 3, and Section 4 describes the main functions and capabilities of the package. Section 5 shows an example of data analysis using **NHPoisson**: the modeling of the occurrence of extreme heat events in a daily temperature series. Finally, some conclusions and future work are discussed in Section 6.

2. Poisson processes and peak over threshold models

A Poisson process (PP in short) is a point process, i.e., a random collection of points in a space where each point represents the occurrence of an event. Time processes are the most common, but PPs can also model events in space or in space-time. The PP assumes that points occur randomly at a given intensity λ (> 0), which characterizes the frequency these events are expected to occur with. The model allows us to calculate values such as the probability of observing a certain number of events in a time period, or the expected time between two events. Examples of events that can be modeled as a PP include the occurrence of extreme environmental events such as fires, earthquakes or heat waves, the times of customer arrivals at queues, the occurrence times of claims in a given period, etc. Herein, we will restrict our attention to the modeling of one-dimensional PPs, in particular, to time PPs.

Statistical modeling based on a PP requires the estimation of the intensity of the process. A realization of a time PP is a sequence of points $(t_i)_{i=1}^n$ observed in a region of R^+. The information provided is that events occur at those points and nowhere else. In most of the real problems, the intensity is time-varying and a nonhomogeneous Poisson process (NHPP) must be considered. The behavior of a time NHPP is generally modeled by representing the intensity $\lambda(t)$ as a function of covariates, that can include time trends, seasonal terms and/or external factors. In order to guarantee that λ is positive, a logarithm link is usually used,

$$\log(\lambda(t;\beta)) = \mathbf{X}^\top(t)\beta$$

where $\mathbf{X}^\top(t)$ is the row vector of covariates at time t and β the vector of parameters.

PPs often appear in the framework of peak over threshold (POT) models. POT models consider the occurrence times of peaks in a sequence of random variables, where a peak is the maximum value in a run of consecutive excesses (observations over a threshold). A well-known result of extreme value theory states that, in an i.i.d. series, the occurrences of excesses over a sufficiently extreme threshold u follow a PP, and the excesses have a generalized Pareto

distribution. More precisely, given a series of i.i.d. random variables $Y_1, Y_2, \ldots$, the process

$$N_n = \left\{ \left(\frac{i}{n+1}, Y_i \right), i = 1, \ldots, n \right\},$$

on regions of the form $(0, 1] \times [u, \infty)$ is approximately a PP for sufficiently high u.

Although these results require i.i.d. series (Y_i), most of them are also applicable, under quite mild conditions, to processes with short-term dependence (Coles 2001).

3. Point processes and POT in other R packages

There are several R packages dealing with point processes. Package **ppstat** (Hansen 2013) focuses on the analysis of marked point processes with discrete marks. It fits generalized linear point process models, including the Hawkes and the Gibbs families, where the intensity is formulated in terms of a linear combination of linear filters. No validation tools are implemented.

The extensive package **spatstat** (Baddeley and Turner 2005; Baddeley, Turner, Mateu, and Bevan 2013) is a package for analyzing spatial point patterns. It supports the model fitting of spatial point processes, in particular Poisson processes, and related inference and diagnostic tools. The function `ppm` fits a spatial point process to an observed point pattern and allows the inclusion of covariates. Two estimation methods, pseudo-maximum likelihood and Huang-Ogata approximate maximum likelihood, are implemented. The available functions and data types (point patterns, spatial windows, pixel images, etc.) are intended for spatial point processes. Some of its functions could be applied to one-dimensional data, but the analysis of time point processes has their own peculiarities, and the spatial framework, in particular the specification of the points of the process, is unnecessarily complicated for the one-dimensional case. The package **yuima** (Brouste *et al.* 2014) develops tools for inference and simulation of stochastic differential equations to model random evolution along continuous, or practically continuous, time. A wide variety of time processes, including compound Poisson processes, can be fitted using a quasi-likelihood approach. Different tools for model selection are available in the package, but the inclusion of covariates is not implemented.

There are also some R packages devoted to POT and extreme analysis, such as **evir** (Pfaff, McNeil, and Stephenson 2012), **POT** (Ribatet 2012), and **ismev** (Heffernan and Stephenson 2014) and its graphical user interface **extRemes** (Gilleland and Katz 2011). The functions `fitpp` and `pot`, from **POT** and **evir** respectively, fit the exceedances above a threshold, in the homogeneous case, using the point process characterization as described in Coles (2001). This approach requires both the occurrence times and the exceedance values. The function `pp.fit` from **ismev** fits the point process model with a generalized linear modeling of each parameter, by maximum likelihood. There are a wide set of tools in these packages for selecting an adequate threshold to apply the POT model, such as the functions `meplot` (in **evir**), `mrl.plot` (in **ismev**) or `tcplot` and `mrlplot` (in **POT**). Since these functions cover the analysis of threshold selection, no function of this type is included in **NHPoisson**.

As far as we know, no commercial statistical software, such as SPSS (IBM Corporation 2013) or Minitab (Minitab Inc. 2010), supports the fit of NHPPs.

4. Features and capabilities

NHPoisson aims to provide a global framework for the modeling of data using NHPPs in one dimension. One of the new functionalities is the maximum likelihood estimation of PP models with the intensity modeled as a function of time-dependent covariates. In particular, it includes a specific likelihood function to estimate the PP resulting from the POT approach, with the peculiarity that it uses information only on the point occurrences, and not on the exceedance values. This likelihood function can also handle missing values. Another capability is the estimation of models with fixed β parameters. The package provides automatic tools for model selection by AIC and likelihood ratio tests, and a comprehensive toolkit for residual analysis and diagnostics to check the fitted model. Random generation of NHPPs with a given time-varying intensity and inference methods based on asymptotic distributions and simulation are also implemented.

4.1. Data and exploratory analysis

The usual input to fit a time PP is the vector of points containing the occurrence times of the events $(t_i)_{i=1}^n$. Since the PP is often used in the framework of POT models, the function `fitPP.fun` also provides the possibility of using as input the original series $(y_i)_{i=1}^T$ and the threshold which defines the extreme events. The function `POTevents.fun` calculates some characteristics of these extreme events: the first and the maximum excess positions, the mean excess, the maximum excess and the length of the events. The maximum excess position defines the occurrence point in the point process. Missing values in the original series are not allowed in this function, since it is difficult to adopt a general rule about how to handle them, specially in the case of missing information between two exceedances.

The functions `emplambda.fun` and `emplambdaD.fun` calculate the empirical occurrence rates on overlapping and disjoint intervals, respectively. The occurrence rate is calculated as the number of points in the considered interval divided by its length, and each rate is assigned to the mean point of the interval. Overlapping intervals are defined by a constant length (argument `lint`), while disjoint intervals can be specified by the number of intervals (argument `nint`) or by a constant length `lint`. All the intervals have the same length except the last one, which will be shorter if the length of the series is not a multiple of `lint` or `nint`. In both cases, a plot of the empirical rate over time can be optionally performed.

4.2. Fitting the model

The function `fitPP.fun` fits a NHPP by conditional maximum likelihood, given the covariates. It is worth to recall that efficient estimation based only on the conditional model, ignoring the information of the marginal process, can only be carried out if weak exogeneity holds (Ericsson and Irons 1994). The conditional likelihood function is,

$$L\left(\beta;(t_i)_{i=1}^n\right) = \exp\left[-\int_A \lambda(t;\beta)dt\right]\prod_{i=1}^n \lambda(t_i;\beta),$$

where $\lambda(t;\beta) = \exp(\mathbf{X}^\top(t)\beta)$ and A is the space where the point process is defined. Assuming that $\lambda(t;\beta)$ is constant in each time unit, the expression of the loglikelihood is

$$LL\left(\beta;(t_i)_{i=1}^n\right) = -\sum_{t=1}^T \lambda(t;\beta) + \sum_{i=1}^n \log\lambda(t_i;\beta),$$

where T is the length of the observation period.

Some adjustments are necessary to define the likelihood function of a NHPP based on the POT approach. Only one occurrence point can be assigned to each event, and that is the point of maximum intensity. The other observations in the extreme event cannot be considered as non occurrence points, since their covariates would show their extreme character. To avoid this problem, these observations are eliminated from the estimation sample. The result is the likelihood we would have obtained, if only the point of maximum intensity in each event had been observed. This is done using an index variable `inddat`, equal to one in the observations to be used and zero otherwise.

This index also allows us to handle missing values by assigning them a value zero. Consequently, all the functions with the argument `inddat`, such as `fitPP.fun`, `emplambda.fun` or `emplambdaD.fun`, can deal with missing values. The function `fitPP.fun` can also fit homogeneous Poisson processes (HPP) by defining the intensity with an intercept and no covariate.

The maximum likelihood (ML) estimation can be done using two different optimization routines, `optim` or `nlminb`, selected in the argument `minfun`. This is useful since, in some cases, one routine can succeed to converge when the other fails. `fitPP.fun` also allows us to fit a model keeping fixed some β parameters (offset terms) using the `fixed` argument. This can be used to specify an a priori known component to be included in the linear predictor.

The output of `fitPP.fun` is an object of the S4 class '`mlePP`', which encapsulates the most relevant information about the maximum likelihood estimation of a NHPP. Some of the slots of this class are: the maximum value of the loglikelihood function, the ML estimators of the β parameters, their covariance matrix, the fitted values $\hat{\lambda}(t)$, and the lower and upper limits of the confidence intervals for each $\lambda(t)$, whose calculation is described below. The estimation of the $\hat{\beta}$ covariance matrix is based on the asymptotic distribution of ML estimators and calculated as the inverse of the negative of the Hessian,

$$\hat{V}(\hat{\beta}) = \left[-\frac{\partial^2 LL(\beta)}{\partial\beta_l \partial\beta_k}\bigg|_{\beta=\hat{\beta}} \right]^{-1},$$

where the second derivative of the considered loglikelihood is

$$\frac{\partial^2 LL(\beta)}{\partial\beta_l \partial\beta_k} = -\sum_{t=1}^{T} \lambda(t;\beta) X_l(t) X_k(t).$$

The class '`mlePP`' is a subclass of the S4 class '`mle`' defined in **stats4**, and most of the generic functions with '`mle`' methods, such as `logLik`, `extractAIC` or `summary`, can be applied to the output of `fitPP.fun`. Some other generic functions related to fitted models, such as `AIC` or `BIC`, can also be applied to '`mlePP`' objects. There is an '`mle`' method for `vcov`, which calculates the approximate variance-covariance matrix of $\hat{\beta}$. However, the `solve` function used to obtain the inverse of the negative of the Hessian matrix may not work, specially when its dimension is high. For this reason, we have implemented the function `VARbeta.fun`, which works in a wider range of situations, since when `solve` gives an error, the calculation of the inverse via its Cholesky decomposition is tried.

The package also provides a method for '`mlePP`' objects for the generic function `profile`, very similar to the '`mle`' method for this function in **stats4**. As in the '`mle`' method, the output is a '`profile.mle`' object, and the '`mle`' method for `confint` can be used to obtain confidence

intervals based on profile likelihoods of the β parameters. Additionally, confidence intervals based on the asymptotic normal approximation of the ML estimators of the β parameters are calculated by the function `confintAsin.fun`.

Confidence intervals for the intensity

Confidence intervals for $\lambda(t) = \exp(\nu(t))$ can be obtained by transforming the confidence intervals for $\nu(t) = \mathbf{X}^\top(t)\beta$. Another option is based on the delta method, but this approach uses an approximation and does not always guarantee the confidence level of the interval. Both alternatives can be calculated by `fitPP.fun`, which calls either the function `CItran.fun` or `CIdelta.fun` depending on the argument `CIty = "Transf"` or `CIty = "Delta"`. In general, the transformation approach (the default) should be preferred.

The transformation approach applies an exponential transformation to the confidence interval of the linear predictor, $\hat{\nu}(t) \pm z_{1-\alpha/2} s.e.(\hat{\nu}(t))$, where $s.e.(\hat{\nu}(t))$ is obtained from,

$$\hat{V}(\hat{\nu}(t)) = \mathbf{X}^\top(t)\hat{V}(\hat{\beta})\mathbf{X}(t).$$

Given that $\hat{\lambda}(t)$ is the ML estimator of $\lambda(t)$ (due to the functional invariance of ML estimators), another confidence interval of $\lambda(t)$ is $\hat{\lambda}(t) \pm z_{1-\alpha/2} s.e.(\hat{\lambda}(t))$. $s.e.(\hat{\lambda}(t))$ is obtained by the delta method (Casella and Berger 2002), which applied to $\exp(\mathbf{X}^\top(t)\hat{\beta})$ gives

$$\hat{V}[\hat{\lambda}(t)] = \sum_{l=1}^{C}\sum_{k=1}^{C} \hat{\lambda}(t)X_l(t)\hat{\lambda}(t)X_k(t)\hat{V}[\hat{\beta}_l, \hat{\beta}_k]. \tag{1}$$

That interval might include values outside the range $[0, \infty)$, so the lower limit is defined as the maximum between 0 and $\hat{\lambda}(t) - z_{1-\alpha/2} s.e.(\hat{\lambda})$.

The function `fitPP.fun` optionally performs, with the argument `dplot = TRUE`, a plot of the fitted intensities together with the confidence intervals for each $\lambda(t)$.

Model selection

There are different tools for aiding in model selection. When the number of covariates is high, automatic procedures such as a stepwise selection can be useful. The function `stepAICmle.fun` performs a stepwise model selection by AIC for '`mlePP`' objects. Three directions, `"forward"`, `"backward"` and `"both"`, are implemented. The initial model is specified by the argument `ImlePP`, and the algorithm stops when, according to the AIC, none of the covariates eliminated from the model or added from the potential covariates set (argument `covariatesAdd`) improves the model fitted in the previous step. The definition of AIC uses constant $k = 2$, but a different value k can be passed as an additional argument.

Authors such as Cook and Weisberg (1999) warn that automatic model selection methods must be used with caution since they can reject reasonable submodels from further consideration. For a more thorough selection, it can be useful to perform the likelihood ratio test by the function `testlik.fun`, which compares a general model `ModG`, with a reduced model `ModR` (a particular case of the first).

To make the selection procedure easier, the function `LRTpv.fun` calculates, for each covariate in a model, the p value of a likelihood ratio test comparing the fitted NHPP with the model excluding that covariate from the linear predictor. The functions `dropAIC.fun` and `addAIC.fun`

(both used in `stepAICmle.fun`) perform a similar task but using the AIC value. The first calculates the AIC for all one-covariate deletions from the current model, while the second calculates it for all one-additions from the covariates supplied in `covariatesAdd`. In both cases, the best covariate to be dropped from (added to) the model is returned together with some additional information. All these automatic selection functions are not implemented for models with fixed parameters.

4.3. Generation of NHPP trajectories and simulation based inference

The generation of trajectories from a NHPP is a useful tool, particularly for simulation based inference. There exist quite a few methods to generate the points from a NHPP. One of the most popular is the thinning approach by Lewis and Shedler (1979), an exact method which does not require the discretization of the intensity function. However, an inversion approach (Çinlar 1975) has been preferred here, since it is faster when the inversion can be easily computed, as in this case. The inversion approach is based on the fact that a NHPP can be transformed into a unit-rate HPP, using the following time scale transformation (Daley and Vere-Jones 2003),

$$t_i^H = \int_0^{t_i^{NH}} \lambda(t)dt. \tag{2}$$

The first step is to generate the points t_i^H in a unit rate HPP via the generation of independent exponential distances. Then, the points t_i^{NH} in a NHPP with intensity $\lambda(t)$ are the result from transforming the points t_i^H with the inverse of the function in Equation 2. Since it is assumed that $\lambda(t)$ is constant in $[t, t+1)$, the points t_i^{NH} are calculated as the lowest integer such that $\sum_{r=0}^{t_i^{NH}} \lambda(r) \geq t_i^H$, where r runs in the non-negative integer numbers.

This method is supported by the function `simNHP.fun`, which generates the points in a period $(0, T)$. The intensity in each time of this period must be provided by the argument `lambda`, and the value T is determined by the length of this intensity vector.

In order to allow the exact reproduction of the results, a seed can be set in the generation process with the argument `fixed.seed`. If the argument is an integer, it gives the seed and if it is `NULL` (the default option), a random seed will be used.

Simulation based inference

Inference on the β parameters and on $\lambda(t)$ intensities can be carried out using likelihood ratio tests, as indicated in Section 4.2. However, there are other values whose inference is difficult to be performed using exact or approximate distributions. In those cases, simulation based inference is a useful too to calculate, for example, envelopes for the number of events to occur in a given time interval, or for the time of the k-th occurrence. The upper and lower limits of these envelopes are the m lowest and m highest values of a simulated sample. These envelopes are not real confidence intervals for the value of the function, but significance bands that specify the critical points for a Monte Carlo test with a significance level $\alpha = 2m/(1 + niter)$, where *niter* is the number of simulated trajectories (Ripley 1981).

The function `GenEnv.fun` calculates simulated envelopes for the value of the function specified in the argument `fun.name`, which must be a function of the NHPP points and any additional argument. It calls the function `funSim.fun` (not intended for the users), which generates a

simulated sample and evaluates the function in that sample. As in `simNHP.fun`, the argument `fixed.seed` allows us to set the seed in the simulation process.

4.4. Model validation

NHPoisson includes the function `globalval.fun` which carries out a complete standard analysis to check the fitted NHPP. However, users can customize their own validation analysis using the available individual functions described in this section.

Two types of residuals can be calculated: uniform (or exponential) and raw residuals. Both of them are useful and provide complementary information.

Exponential or uniform residuals

A usual approach to validate a NHPP with intensity $\lambda(t)$ is to transform it into a unit-rate HPP, using the time transformation in Equation 2. Then, the validation analysis consists in analyzing the inter-event distances $d_i^* = t_i^* - t_{i-1}^*$ that, under the null hypothesis, must be an i.i.d. exponential sample. Equivalently, the validation can be based on the transformed distances $\exp(-d_i^*)$ that must be an i.i.d. uniform sample. These residuals may be plotted against covariates, or potential covariates, as an informal assessment of the need of including them. For more details on this topic, see the diagnostic analysis for Cox-Snell type residuals in Collett (1994).

Hence, the validation analysis starts by applying the function `transfH.fun` to transform the NHPP into a HPP. Then, the exponential and uniform residuals of the homogeneous process are calculated using `unifres.fun`. The function `graphresU.fun` checks the uniform behavior and the serial correlation in the uniform residuals. To analyze the correlation, the function calculates the Pearson correlation coefficient, the Ljung-Box p values and a lagged serial correlation plot. To analyze the uniform behavior, a Kolmogorov-Smirnov test and a uniform qqplot with a 95% confidence envelope are performed. In addition, an index plot of the residuals and residual plots against the variables specified in the argument `Xvariables` are drawn.

Raw residuals

A complementary approach to validate a NHPP is the analysis of the raw residuals, see Lewis (1972) and Baddeley, Turner, Møller, and Hazelton (2005) for this type of residuals in time and space-time processes. The error (or innovation) process is $\varepsilon(t) = N(t) - \int_0^t \lambda(u)du$ which, when the model is true, must be a zero-mean martingale. The corresponding raw residual process is

$$R(l) = \sum_{t_i \in (0,l)} I_{t_i} - \int_0^l \hat{\lambda}(u)du,$$

with I_{t_i} an indicator variable, equal to 1 at time t_i. The residuals are defined by the increments of the raw process in intervals (l_1, l_2), which can be disjoint or overlapping, divided by its length,

$$r(l_1, l_2) = \frac{1}{l_2 - l_1}(R(l_2) - R(l_1)) = \frac{1}{l_2 - l_1}\left(\sum_{t_i \in (l_1, l_2)} I_{t_i} - \int_{l_1}^{l_2} \hat{\lambda}(u)du\right). \quad (3)$$

These residuals can be interpreted as observed minus fitted value residuals and they are useful to check the fitted PP intensity since, under the correct model, $r(l_1, l_2) \approx 0$.

The functions `CalcRes.fun` and `CalcResD.fun` calculate the raw residuals using overlapping and disjoint intervals (l_1, l_2) respectively. The residuals $ro(l_1, l_2)$ are based on overlapping intervals of a given length, centered on each time t. It is noteworthy that, due to their overlapping structure, they will not be independent even under the model. The residuals $rd(l_1, l_2)$ are calculated on disjoint intervals of equal length, and assigned to the mean point of the interval. In both functions, the empirical and the mean fitted intensities used to calculate the raw residuals, $\sum_{t_i \in (l_1, l_2)} I_{t_i}/(l_2 - l_1)$ and $\int_{l_1}^{l_2} \hat{\lambda}(u)du/(l_2 - l_1)$, are elements of the output list. Scaled versions of the residuals,

$$r_{sca}(l_1, l_2) = \frac{1}{l_2 - l_1} \left(\sum_{t_i \in (l_1, l_2)} h(t_i) I_{t_i} - \int_{l_1}^{l_2} h(u) \hat{\lambda}(u) du \right), \tag{4}$$

can be calculated using a non negative weight function $h(u)$. By default, the Pearson residuals are calculated, $h(u) = 1/\sqrt{\hat{\lambda}(u)}$, but any other weight function can be provided through the argument `h`. The function `graphrate.fun` plots, for comparative purposes, the empirical and the mean fitted intensities against time in the same graph.

A usual diagnostic tool for analyzing the covariate effects is a residual plot. In the case of raw residuals, which are not linked to a time instant but to an interval, two different plots can be used depending on the characteristics of the covariate: the time residual plot or the lurking variable plot. A qqplot to check the residual distribution is also implemented.

Time residual plots The function `graphres.fun` plots any type of residuals (raw or scaled, overlapping or disjoint) against time or a specific type of variables. In these plots, the residuals are plotted against the value of the variable in the mean point of the interval and, consequently, they are adequate only for variables which are lineal or, at least, monotonous functions of time. A lowess smoother of the residuals can be optionally plotted to make the interpretation easier. In the case of overlapping intervals, the residuals of the occurrence points are marked with different symbols.

Approximate confidence envelopes can be optionally plotted in these graphs. These envelopes are based on the approach suggested by Baddeley *et al.* (2005) for spatial inhomogeneous Poisson processes. Their interpretation, including the pointwise significance, is similar to the limits ± 2 used for the standardized residuals in linear regression analysis. The envelopes are based on the variance of the error process $\varepsilon(l_1, l_2) = \varepsilon(l_2) - \varepsilon(l_1)$. The variance of the raw error processes for NHPPs is

$$V[\varepsilon_r(l_1, l_2)] = \int_{l_1}^{l_2} \lambda(u) du,$$

and for the scaled Pearson error process

$$V[\varepsilon_P(l_1, l_2)] = |l_2 - l_1|.$$

These variances can be estimated by plugging into the expressions the fitted intensity $\hat{\lambda}(u)$. It is noteworthy that the variance of the error $\hat{V}[\varepsilon(l_1, l_2)]$ will overestimate the variance of the

residual process and, consequently, $\pm 2\sqrt{\hat{V}[\varepsilon(l_1, l_2)]}$ will be wider than a confidence interval based on the residual process variance. For the raw residuals defined in (3), the envelopes are,

$$\pm \frac{2}{l_2 - l_1} \sqrt{\sum_{i \in (l_1, l_2)} \hat{\lambda}(i)},$$

where i runs over the integers in (l_1, l_2). Analogously, for the Pearson residuals in Equation 4,

$$\pm \frac{2}{\sqrt{l_2 - l_1}}.$$

Lurking variable plots A possible way to analyze the relationship between the residuals linked to intervals and a covariate which is not a monotonous function of time is to plot them against the mean value of the variable in that interval. A more complicated but better approach is to calculate the residuals on intervals defined by the values of the covariate instead of by time. The following approach is a modification of the lurking variable plot by Baddeley *et al.* (2005). The intervals considered to calculate the residuals for a covariate $X(t)$ are

$$W(P_{X,i}, P_{X,i+1}) = \{t : P_{X,i} \leq X(t) < P_{X,i+1}\},$$

where $P_{X,i}$ is the sample i-percentile of X. The function `graphResX.fun` plots the residuals against the mean value of the covariate in that interval. In the case of raw or Pearson residuals, it also calculates approximate confidence envelopes as the ones described for time residual plots. The function `graphResCov.fun` performs the lurking variable plot for each variable in the argument `Xvar`.

Residual qqplot A common plot to check the distributional assumptions in a model is the qqplot. The theoretical distribution of the NHPP raw residuals is unknown but a simulation based qqplot can be calculated. The function `resQQplot.fun` compares the empirical quantiles of the residuals with the expected quantiles under the fitted model, obtained from the following simulation process.

First, the model is fitted to the data set and the residuals r_i are calculated. The order statistics $r_{(1)} \leq r_{(2)} \leq \ldots \leq r_{(n)}$ are obtained from the residual sample. Then, *niter* independent trajectories of a NHPP($\hat{\lambda}(t)$) are generated and a NHPP is fitted to each one. The corresponding fitted intensities and the ordered residuals from the i-th trajectory are denoted $\hat{\lambda}^{(i)}(t)$ and $r^{(i)}_{(1)} \leq r^{(i)}_{(2)} \leq \ldots \leq r^{(i)}_{(n)}$, respectively. The expected j-th quantile of the residuals under the fitted model is calculated as the sample mean of the j-th order statistic

$$e_j = \frac{1}{niter} \sum_{i=1}^{niter} r^{(i)}_{(j)}.$$

Individual $(1 - \alpha)\%$ envelopes for each residual are defined by the corresponding $\alpha/2$ and $1 - \alpha/2$ sample quantiles.

The function `resQQplot.fun` calculates for each residual the simulated expected quantile and plots them together with the $1 - \alpha$ individual envelopes. Disjoint or overlapping and Pearson or raw residuals are supported by this function, but the overlapping option can require a high time computing cost. The output list of the function includes the expected quantiles and the

upper and lower limits of the envelopes of the residuals. The argument `fixed.seed` allows us to set the seed to obtain reproducible results.

A global validation function

A standard validation analysis to check a NHPP model can be performed in one step using the function `globalval.fun`. First, the uniform residuals are calculated and analyzed. The residuals are plotted against the covariates in `Xvar` and, if this argument is `NULL`, all the covariates in the fitted model are considered. The empirical and the cumulative mean fitted intensities (using optionally overlapping or disjoint intervals) are plotted together. Using these intensities, the raw or scaled residuals are calculated, and residual plots against time and the variables in `Xvart` are carried out. The lurking variable plots for the variables in `Xvar` are also performed; as before, if this argument is `NULL`, all the covariates in the fitted model are considered. Finally, a residual qqplot to check the distributional assumptions can be optionally calculated. By default, the qqplot is carried out for the Pearson residuals in disjoint intervals.

The argument `histWgraph = TRUE` (the default option) opens a new graphical device (using `dev.new(record = TRUE)`, which may not work on all systems), so that the history of all the plots carried out by the function is recorded in the same device. This means that the resulting plots can be accessed by scrolling up and down. The argument `plotDisp` provides the layout to display the residual plots, 2×2 by default. These both arguments are available in all the functions which carry out several residual plots: `graphres.fun`, `graphresU.fun`, `graphresCov.fun`, etc.

5. An example

In this section, we illustrate how the package **NHPoisson** can be used to model a data set, from the data preparation for a POT approach to the validation analysis. Data are daily maximum temperature series, and our aim is to model the occurrence of its extreme heat events using a NHPP.

5.1. Data

The data set `BarTxTn` is available in the package **NHPoisson**. The variables Tx and Tn are the daily maximum and minimum temperature series, in tenths of °C, corresponding to the summer months May, June, July, August and September (MJJAS) from 1951 to 2004 in Barcelona (Spain). The date (day, month and year) of the observations, and some variables representing the short and long-term temperature evolution are also available. The daily temperature series were provided by the Spanish Meteorological Office (AEMET).

The variables are recorded at a discrete time scale, as in most real data applications. However, given that the time unit is short compared with the length of the recorded period and the occurrence rate very low, the use of a continuous time model, such as the PP, is justified.

5.2. Data preparation

To define an extreme heat event (EHE) we use the POT approach. The extreme threshold is 31.8 °C, which is the 95th percentile of Tx during a reference period (months JJA in

[1971, 2000]). The selection of this threshold (not shown here) is based on climate basis and on statistical criteria about the Poisson behavior of the resulting peak series (using for example the functions `diplot`, `mrlplot` and `tcplot` of the **POT** package). The EHEs and their characteristics are obtained by,

```
R> data("BarTxTn", package = "NHPoisson")
R> dateB <- cbind(BarTxTn$ano, BarTxTn$mes, BarTxTn$diames)
R> BarEv <- POTevents.fun(T = BarTxTn$Tx, thres = 318, date = dateB)

Number of events:  137
Number of excesses over threshold 318 : 253
```

where the matrix `dateB` contains the date of each observation, and the object `BarEv` several characteristics of the 137 EHEs, such as `Px`, the points of maximum intensity in the extreme events, which are the occurrence points of the process. Another element of the object is the index `inddat`, which marks the observations to be used in the estimation process,

```
R> Px <- BarEv$Px
R> inddat <- BarEv$inddat
```

To describe the time evolution of the EHE occurrence, a plot of its empirical rate calculated in 153-day long overlapping periods is performed using the following commands (153 is the length of the period MJJAS),

```
R> tB <- BarTxTn$ano + rep(c(0:152) / 153,55)
R> emplambdaB <- emplambda.fun(posE = Px, inddat = inddat, t = tB,
+    lint = 153, tit = "Barcelona")
```

where `tB` is an index time of the summer months linked as consecutive time periods. The argument `inddat` must be specified only when there are missing values, or when the point process is defined using the POT approach and the length of the resulting events can be greater than one. By default, all the `inddat` values are equal to 1 so that all the available observations are included in the estimation process.

5.3. Fitting the model

Due to the characteristics of the EHE occurrence (seasonal behavior and a nonmonotonic trend), it must be modeled by a PP with a nonhomogeneous intensity (Abaurrea, Asín, Cebrián, and Centelles 2007). This intensity is modeled as a deterministic function of covariates such as long and short-term temperature, and seasonal terms (the part corresponding to the summer months of the harmonic functions describing the annual cycle). Temperature-harmonic interaction terms are also considered as possible covariates.

As an exploratory step to check which covariates are more influential, we carry out an automatic stepwise selection by AIC in both directions. First, the initial model which only includes the intercept is fitted, and the first order harmonic terms and $Txm31$, $Tnm31$, TTx and TTn temperature covariates (all available from $BarTxTn$) are considered as potential covariates. In order to make the interpretation of the output easier, names are assigned to

each covariate, using `dimnames`. It is recalled that with the arguments `modSim = TRUE`, `dplot = FALSE` and `modCI = FALSE`, the information on the screen is eliminated, the fitted intensity is not plotted and the confidence intervals are not calculated.

```
R> mod1Bind <- fitPP.fun(covariates = NULL, posE = BarEv$Px,
+    inddat = BarEv$inddat, tit = "BAR  Intercept", start = list(b0 = 1))
R> covB <- cbind(cos(2 * pi * BarTxTn$dia / 365),
+    sin(2 * pi * BarTxTn$dia / 365),  BarTxTn$Txm31, BarTxTn$Tnm31,
+    BarTxTn$TTx, BarTxTn$TTn)
R> dimnames(covB) <- list(NULL, c("Cos", "Sin", "Txm31",  "Tnm31",
+    "TTx", "TTn"))
R> aux <- stepAICmle.fun(ImlePP = mod1Bind, covariatesAdd = covB,
+    startAdd = c(1, -1, 0, 0, 0, 0), direction = "both")

Step Forward 1
 Initial model. AIC:  1400.471
 Initial model adding covariates
          AIC
Cos   1353.504
Sin   1388.329
Txm31 1072.380
Tnm31 1125.856
TTx   1367.911
TTn   1382.534
The best covariate to add is  Txm31
This covariate improves the model

Step Backward 1
 Initial model  1072.38
 Initial model deleting covariate
          AIC
Txm31 1400.471
The best covariate to drop is  Txm31
This drop  does not improve the model

Step Forward 2
 Initial model. AIC:  1072.38
 Initial model adding covariates
          AIC
Cos   1073.688
Sin   1073.257
Tnm31 1074.372
TTx   1074.379
TTn   1073.412
The best covariate to add is  Sin
This covariate does not improve the model
```

```
Step Backward 2
 Initial model  1072.38
 Initial model deleting covariate
           AIC
Txm31 1400.471
The best covariate to drop is  Txm31
This drop  does not improve the model

Final model
The covariates added and dropped to the initial model are:
  added
"Txm31"

Summary of the model:
Maximum likelihood estimation
Call:
fitPP.fun(covariates = ..1, start = ..2, posE = BarEv$Px,
 inddat = BarEv$inddat,    modCI = FALSE, tit = "BAR  Intercept",
  modSim = TRUE, dplot = FALSE)

Coefficients:
      Estimate
b0 -21.7177999
b1   0.0653722

-2 log L: 1068.38
```

According to the AIC, the best model should only include the short-term temperature covariate *Txm31*, but a more thorough covariate selection is advisable. We suggest a selection based on the likelihood ratio test, using a forward stepwise approach controlled by the user. We start by fitting a HPP,

```
R> modB.1 <- fitPP.fun(covariates = NULL, posE  = Px, inddat = inddat,
+    tit = "BARCELONA Tx; Intercept", start = list(b0 = 1), dplot = FALSE,
+    modCI = FALSE)

Number of observations  not used in the estimation process:  116
Total number of time observations:  8415
Number of events:  137
Convergence code:  0
Convergence attained
Loglikelihood:  -699.236
Estimated coefficients:
    b0
-4.104
Full coefficients:
    b0
```

```
-4.104
attr(,"TypeCoeff")
[1] "Fixed: No  fixed parameters"
```

The following commands analyze if the first order harmonic terms must be included,

```
R> covB <- cbind(cos(2 * pi * BarTxTn$dia / 365),
+    sin(2 * pi * BarTxTn$dia / 365))
R> modB.2 <- fitPP.fun(covariates = covB, posE = Px, inddat = inddat,
+    tit = "BARCELONA Tx; Cos, Sin", start = list(b0 = -100, b1 = 1,
+    b2 = 1), modSim = TRUE,  dplot = FALSE, modCI = FALSE)
R> aux <- testlik.fun(ModG = modB.2, ModR = modB.1)

General Model (hypothesis H1):  BARCELONA Tx; Cos, Sin
Reduced Model (hypothesis H0):  BARCELONA Tx; Intercept
ML ratio test statistic:  154.19
P-value: 0
```

Since the p value is 0, the first order harmonic is included and the inclusion of the second order harmonic is checked.

```
R> covB <- cbind(cos(2 * pi * BarTxTn$dia / 365),
+    sin(2 * pi * BarTxTn$dia / 365),
+    cos(4 * pi * BarTxTn$dia / 365),  sin(4 * pi * BarTxTn$dia / 365))
R> modB.3 <- fitPP.fun(covariates = covB, posE = Px, inddat = inddat,
+    tit = "BARCELONA Tx; Cos, Sin, Cos2, Sin2", start = list(b0 = -100,
+    b1 = 1, b2 = 1, b3 = 1, b4 = 1),  modSim = TRUE, dplot = FALSE,
+    modCI = FALSE)
R> aux <- testlik.fun(ModG = modB.3, ModR = modB.2)

General Model (hypothesis H1):  BARCELONA Tx; Cos, Sin, Cos2, Sin2
Reduced Model (hypothesis H0):  BARCELONA Tx; Cos, Sin
ML ratio test statistic:  0.96
P-value: 0.619
```

The p value 0.619 rejects the inclusion of the second order harmonic. Analogously, long and short term temperature signals *TTx*, *TTn*, *Txm31* and *Tnm31* and interaction terms between the first order harmonic and the significant temperature covariates are analyzed. The final model resulting from this forward covariate selection process includes the first order harmonic term and *Txm31*. To fit this model, the default values of `modSim`, `modCI`, `CItype` and `dplot` are used and, consequently, model information is shown, transformed confidence intervals are calculated and the fitted intensity is plotted, see Figure 1.

```
R> covB.final <- cbind(cos(2 * pi * BarTxTn$dia / 365),
+    sin(2 * pi * BarTxTn$dia / 365), BarTxTn$Txm31)
R> dimnames(covB.final) <- list(NULL, c("Cos", "Sin", "Txm31"))
R> modB.final <- fitPP.fun(covariates = covB.final,  posE = Px,
+    inddat = inddat,  tim = tB, tit = "BARCELONA Tx; Cos, Sin, Txm31",
+    start = list(b0 = -100, b1 = 1, b2 = 1, b3 = 0))
```

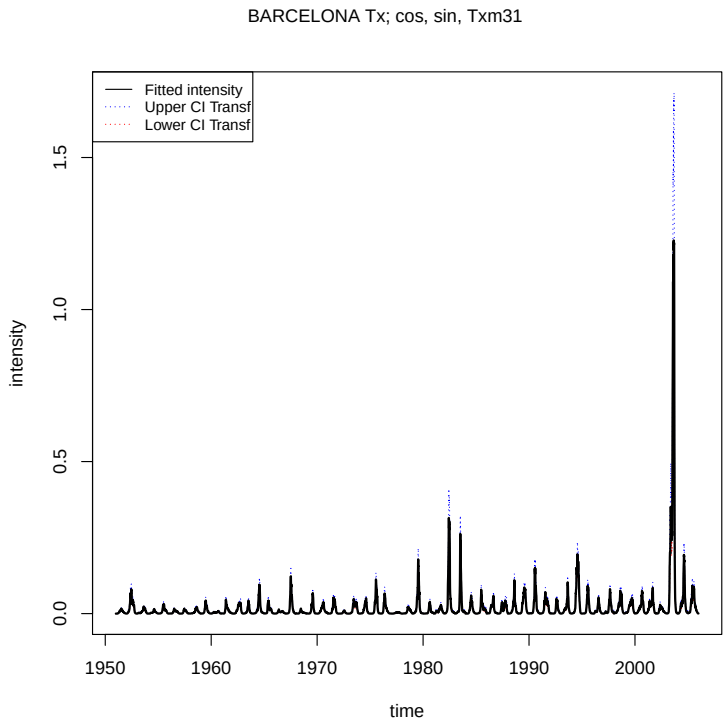

Figure 1: Fitted intensity and transformed confidence intervals (Section 4.2) of the selected NHPP model.

```
Number of observations  not used in the estimation process:  116
Total number of time observations:  8415
Number of events:  137

Convergence code:  0
Convergence attained
Loglikelihood:  -533.625

Estimated coefficients:
     b0      b1      b2      b3
-21.681   0.080   0.320   0.066
Full coefficients:
     b0      b1      b2      b3
-21.681   0.080   0.320   0.066
attr(,"TypeCoeff")
[1] "Fixed: No  fixed parameters"
```

`LRTpv.fun` shows that the first order harmonic terms are not individually significant when *Txm31* is included in the model. However, they are kept in order to make easier the comparison of the temperature effect with models for other locations.

```
R> aux <- LRTpv.fun(modB.final)
```

```
  The p-values of the LRT comparing the initial model and the model
  without the covariate

      p-values
Cos      0.935
Sin      0.508
Txm31    0.000
```

The standard error of the estimated coefficients and a summary of the final model is obtained using,

```
R> summary(modB.final)

Maximum likelihood estimation

Call:
fitPP.fun(covariates = covB.final, start = list(b0 = -100, b1 = 1,
    b2 = 1, b3 = 0), posE = Px, inddat = inddat, tim = tB,
     tit = "BARCELONA Tx; Cos, Sin, Txm31")

Coefficients:
       Estimate  Std. Error
b0 -21.68078342 1.027972827
b1   0.07971801 0.908504548
b2   0.32020449 0.447603298
b3   0.06591333 0.003559439

-2 log L: 1067.25
```

The 95% confidence intervals of the β parameters, based on the profile likelihood and on the properties of the ML estimators, can be obtained with the following commands,

```
R> confint(modB.final)

Profiling...
          2.5 %       97.5 %
b0 -24.03266857 -19.46601384
b1  -1.92800787   1.90282196
b2  -0.66536902   1.20239256
b3   0.05685904   0.07475327

R>  confintAsin.fun(modB.final)

          2.5 %      97.5 %
b0 -23.69557314 -19.6659937
b1  -1.70091819   1.8603542
b2  -0.55708186   1.1974908
b3   0.05893695   0.0728897
```

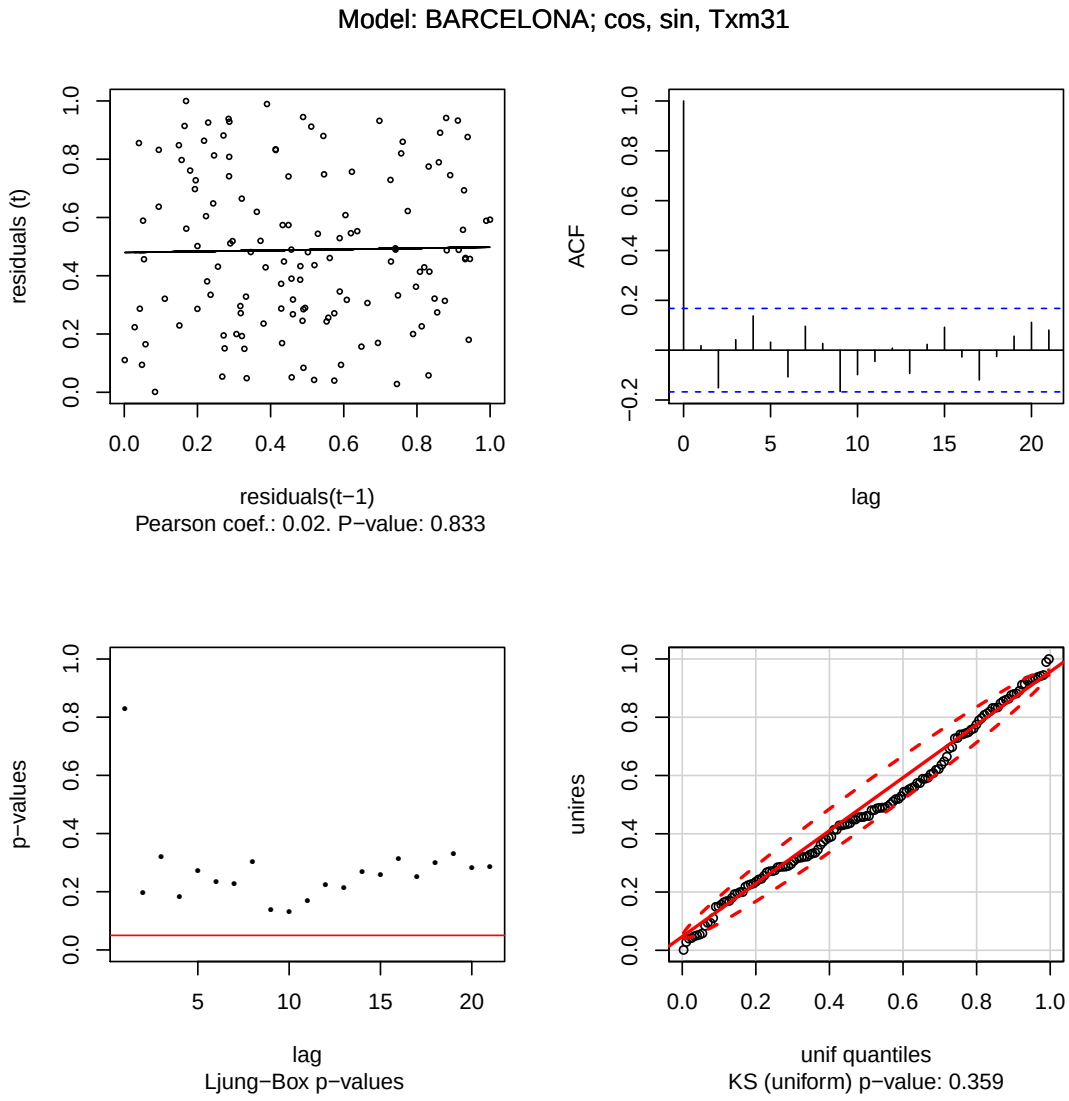

Figure 2: Validation plots of the uniform residuals of the final NHPP model.

5.4. Model validation

The validation starts by the analysis of the uniform residuals using the commands,

```
R> posEHB <- transfH.fun(modB.final)$posEH
R> resB <- unifres.fun(posEHB)
R> graphresU.fun(unires = resB$unires, posE = modB.final@posE,
+    Xvariables = cbind(covB.final, BarTxTn$dia),
+    namXv = c("cos", "sin", "Txm31", "summer day index"),
+    tit = "BARCELONA; cos, sin, Txm31", addlow = FALSE)
```

The results of the autocorrelation analysis and the uniform behaviour are satisfactory, see Figure 2. The plots of the uniform residuals against the covariates included in the model and the index time corresponding to the summer cycle, not shown here, are also requested for

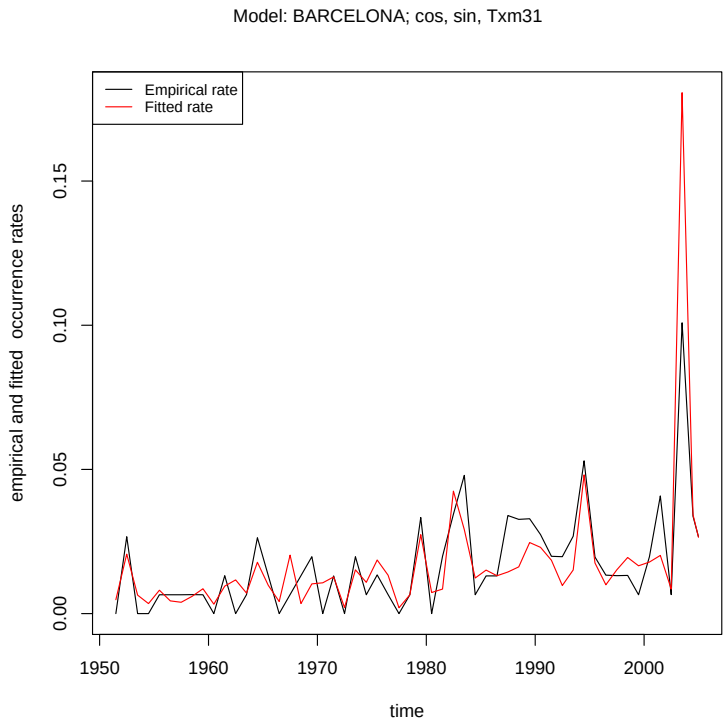

Figure 3: Empirical and fitted intensities calculated on disjoint summer intervals.

analysing the linear predictor.

In order to analyse the raw residual, the scaled Pearson residuals on 153-day long disjoint intervals are calculated as follows,

```
R> ResDB <- CalcResD.fun(mlePP = modB.final, lint = 153)

Number of intervals to calculate the disjoint residuals: 55 of length: 153
```

The output of the function `CalcResD.fun` is a list and one of its elements are the residuals, which can be used in other commands, for example to analyze their normal behavior

```
R> qqnorm(ResDB$RawRes)
```

Other elements in the output list are the empirical and the fitted intensities used to calculate the raw residuals. They can be graphically compared, see Figure 3, using the following command,

```
R> graphrate.fun(ResDB, tit = "BARCELONA; cos, sin, Txm31")
```

The lurking variable plots of nonscaled residuals against *Txm31* (a covariate included in the model) and *TTx* (not included), and their confidence envelopes, are obtained from the following code snippet,

```
R> covBtemp <- cbind(BarTxTn$Txm31, BarTxTn$TTx)
R> aux <- graphResCov.fun(mlePP = modB.final, nint = 50,
```

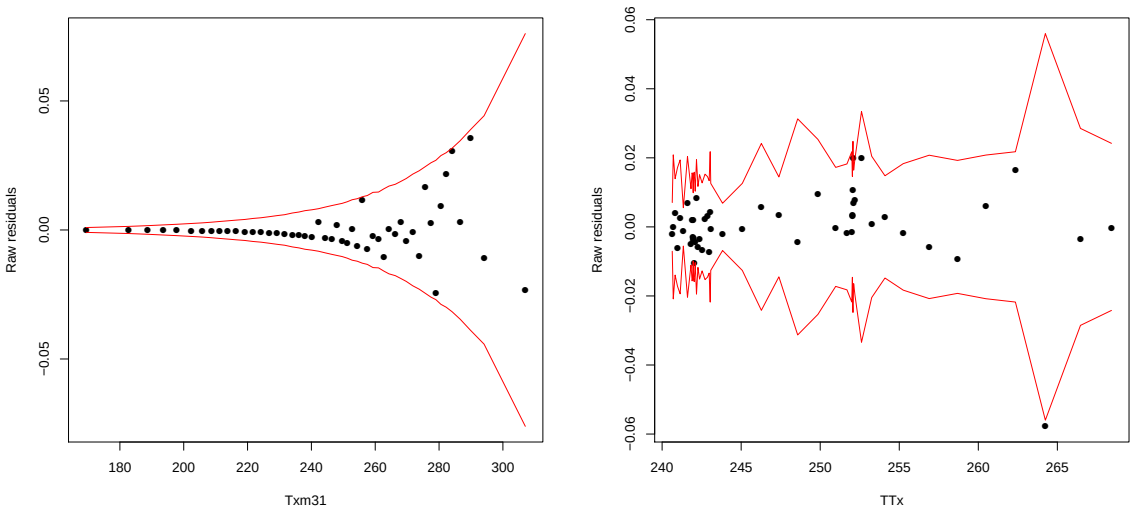

Figure 4: Lurking plots based on raw residuals for two temperature variables.

```
+     X = covBtemp, namX = c("Txm31", "TTx"),
+     tit = "BARCELONA; cos, sin, Txm31", typeRes = "Raw")
```

The lurking plots in Figure 4 show that the influence of *Txm31* is properly captured by the model and that the inclusion of *TTx* is not necessary. If we want to obtain plots displayed individually (but recorded on the same graphical device), the arguments `plotDisp=c(1,1)` and the default option `histWgraph=TRUE` should be used.

The time residual plot for the `ResDB` residuals and the residual qqplot are carried out using the following commands (in `resQQplot.fun` the seed is set to obtain reproducible results.),

```
R> graphres.fun(objres = ResDB, typeRes = "Pearson", addlow = TRUE,
+     plotDisp = c(1, 1), tit = "BARCELONA; cos, sin, Txm31")
R> aux <- resQQplot.fun(nsim = 100, objres = ResDB,  covariates = covB.final,
+     tit = "BARCELONA; cos, sin, Txm31", fixed.seed = 123)
```

The plots do not show any evidence against the fitted model, see Figure 5. It is noteworthy to recall that depending on the number of points in the process, the number of simulations and the computer specifications, the computation time of the function `resQQplot.fun` can be quite high. In any case, at least 100 simulations are recommended.

All the previous analysis can be obtained in one step using the `globalval.fun`,

```
R> aux <- globalval.fun(mlePP = modB.final, lint = 153,
+      typeI = "Disjoint", typeResLV = "Raw", nintLP = 50,
+      tit = "BARCELONA; cos, sin, Txm31",  Xvar = covBtemp,
+      namXvar = c("Txm31", "TTx"), resqqplot = TRUE, fixed.seed = 123)

Number of intervals to calculate the disjoint residuals: 55 of length: 153
```

5.5. Simulation based inference

Once the model is satisfactorily checked, inference can be performed. The following command

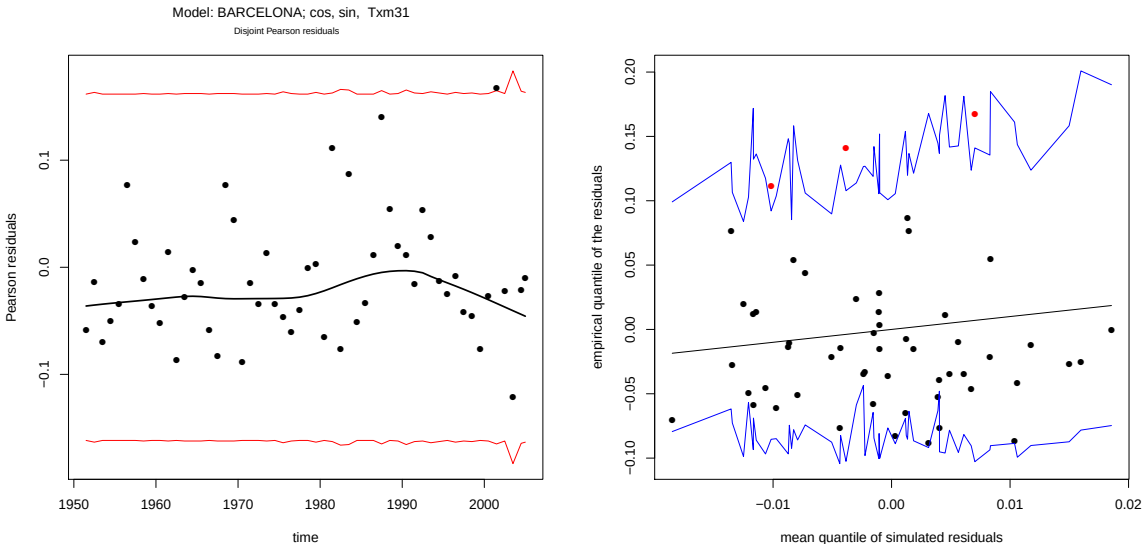

Figure 5: Time residual plot and residual qqplot (with seed 123 and calculated using one core) for the Pearson disjoint residuals.

calculates a point estimator (the mean) and a prediction envelope for the number of occurrences in a summer with a given intensity function; the results are obtained with R 3.1.3 for Windows (64 bit) and a seed equal to 123,

```
R> aux <- GenEnv.fun(lambda = modB.final@lambdafit[8263:8415],
+    fun.name = "length", nsim = 1000, fixed.seed = 123)

Lower interval:  1
Mean value:  3.975
Upper interval:  8
Number of valid simulations:  1000
```

Using the same approach, we can estimate the occurrence time of the first EHE,

```
R> aux <- GenEnv.fun(lambda = modB.final@lambdafit[8263:8415],
+    fun.name = "posk.fun", fun.args = 1, nsim = 1000, fixed.seed = 123)

Lower interval:  21
Mean value:  54.6383
Upper interval:  85.7
Number of valid simulations:  987
```

where `posk.fun(x, k)` returns the element in the k-th row of the vector x. The estimated occurrence time of the first EHE in a summer with the considered intensity is the 24th of June and the prediction envelope moves from the 21st of May to the 25th of July. It is noteworthy that this function is estimating the occurrence time of the first EHE given that, at least, one EHE occurs in that period. This is due to the fact that simulated trajectories

where no EHE occurs are not included in the calculations. Denoting the number of points in the specified time interval by $N(.,.)$, the probability that this occurs is $P(N(8263, 8415) = 0) = \int_{8263}^{8415} \lambda(t)dt = 0.023$, practically negligible. This probability could also be estimated by simulation using the function `simNHP.fun`. The simplest, but not fastest, way is

```
R> L <- NULL
R> for (i in c(1:1000))
+    L[i] <- length(simNHP.fun(lambda =
+      modB.final@lambdafit[8263:8415], fixed.seed = (123 + i))$posNH)
R> P0 <- sum(L == 0) / 1000
R> P0

[1] 0.014
```

6. Conclusions and future work

NHPoisson is an R package designed to facilitate the modeling of data using one-dimensional NHPPs whose intensity is described as a function of covariates. This is an important feature since the use of nonhomogeneous processes allows us to include information on the state of the system, the geographical position of an event, or different environmental characteristics. The package provides an assembly of all the steps involved in the modeling process: data preparation, maximum likelihood estimation (including fixed parameters), covariate selection, inference, simulation, and a wide toolkit for validation analysis, including residuals and diagnostic techniques adapted from other statistical models. Since the output of the main function `fitPP.fun` is an S4 class object which extends the '`mle`' class, many available methods for this type of objects and some generic functions related to fitted models can also be used.

This package does not include techniques on threshold selection for POT models since they are already available in other packages. However, an extension of the package could include the implementation of new techniques on this topic, such as the ones developed by Wadsworth and Tawn (2012).

Some future additions for the package are functions for the statistical modeling based on multivariate nonhomogeneous Poisson processes. An example of this type of model is the common Poisson shock process, a multivariate process with dependent components.

Acknowledgments

First, we thank all the people supporting the R project which provides a nice, powerful and open-source environment for developing and spreading new statistical techniques, specially the CRAN team for their efficient work. We also thank the anonymous reviewers for their helpful comments to improve the package and the manuscript. Forthcoming suggestions from all users are welcomed. This work was partially supported by Ministerio de Educación y Ciencia (Spanish Department of Science) through the project CGL2009-09646.

References

Abaurrea J, Asín J, Cebrián AC, Centelles A (2007). "Modeling and Forecasting Extreme Heat Events in the Central Ebro Valley, a Continental-Mediterranean Area." *Global and Planetary Change*, **57**(1–2), 43–58.

Abaurrea J, Cebrián AC (2002). "Drought Analysis Based on a Cluster Poisson Model: Distribution of the Most Severe Drought." *Climate Research*, **22**(4), 227–235.

Baddeley A, Turner R, Mateu J, Bevan A (2013). "Hybrids of Gibbs Point Process Models and Their Implementation." *Journal of Statistical Software*, **55**(11), 1–43. URL http://www.jstatsoft.org/v55/i11/.

Baddeley AJ, Turner R (2005). "**spatstat**: An R Package for Analyzing Spatial Point Patterns." *Journal of Statistical Software*, **12**(6), 1–42. URL http://www.jstatsoft.org/v12/i06/.

Baddeley AJ, Turner R, Møller J, Hazelton J (2005). "Residual Analysis for Spatial Point Processes." *Journal of the Royal Statistical Society B*, **67**(5), 617–666.

Brouste A, Fukasawa M, Hino H, Iacus S, Kamatani K, Koike Y, Masuda H, Nomura R, Ogihara T, Shimuzu Y, Uchida M, Yoshida N (2014). "The YUIMA Project: A Computational Framework for Simulation and Inference of Stochastic Differential Equations." *Journal of Statistical Software*, **57**(4), 1–51. URL http://www.jstatsoft.org/v57/i04/.

Casella G, Berger RL (2002). *Statistical Inference*. Brooks/Cole.

Çinlar E (1975). *Introduction to Stochastic Processes*. Prentice-Hall.

Cebrian AC (2015). ***NHPoisson**: Modelling and Validation of Non Homogeneous Poisson Processes*. R package version 3.1, URL http://CRAN.R-project.org/package=NHPoisson.

Coles S (2001). *An Introduction to Statistical Modeling of Extreme Values*. Springer-Verlag.

Collett D (1994). *Modelling Survival Data in Medical Research*. Chapman and Hall, London.

Cook RD, Weisberg S (1999). *Applied Regression Including Computing and Graphics*. John Wiley & Sons.

Daley D, Vere-Jones D (2003). *An Introduction to the Theory of Point Processes. Vol I: Elementary Theory and Methods*. Springer-Verlag.

Embrechts P, Klüppelberg C, Mikosch T (1997). *Modelling Extremal Events*. Springer-Verlag.

Ericsson NR, Irons JS (1994). *Testing Exogeneity*. Oxford University Press.

Gilleland E, Katz RW (2011). "New Software to Analyze How Extremes Change over Time." *Eos, Transactions American Geophysical Union*, **92**(2), 13–14.

Hansen NR (2013). ***ppstat**: Point Process Statistics*. R package version 0.9, URL http://CRAN.R-project.org/package=ppstat.

Heffernan JE, Stephenson AG (2014). ***ismev**: An Introduction to Statistical Modeling of Extreme Values*. R package version 1.40, URL http://CRAN.R-project.org/package=ismev.

IBM Corporation (2013). *IBM SPSS Statistics 22*. IBM Corporation, Armonk. URL http://www.ibm.com/software/analytics/spss/.

Katz RW, Parlange MB, Naveau P (2002). "Statistics of Extremes in Hydrology." *Advances in Water Resources*, **25**(8–12), 1287–1304.

Kutoyants YA (1998). *Statistical Inference for Spatial Poison Processes*. Springer-Verlag.

Lewis PAW (1972). "Recent Results in the Statistical Analysis of Univariate Point Processes." In PAW Lewis (ed.), *Stochastic Point Processes*, pp. 1–54. John Wiley & Sons.

Lewis PAW, Shedler GS (1979). "Simulation of Nonhomogenous Poisson Processes by Thinning." *Naval Research Logistics Quarterly*, **26**(3), 403–413.

Madsen H, Rasmussen PF, Rosbjerg D (1997). "Comparison of Annual Maximum Series and Partial Duration Series Methods for Modelling Extreme Hydrologic Events. 1. At Site Modelling." *Water Resources Research*, **33**(4), 747–757.

Minitab Inc (2010). *Minitab 17: Statistical Software for Process Control, Improvement and Education*. Minitab Inc., State College. URL http://www.minitab.com/.

Ogata Y (1988). "Statistical Models for Earthquake Occurrences and Residual Analysis for Point Processes." *Journal of the American Statistical Association*, **83**(401), 9–27.

Pfaff B, McNeil A, Stephenson AG (2012). ***evir**: Extreme Values in R*. R package version 1.7-3, URL http://CRAN.R-project.org/package=evir.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Ribatet M (2012). ***POT**: Generalized Pareto Distribution and Peaks over Threshold*. R package version 1.1-3, URL http://CRAN.R-project.org/package=POT.

Ripley B (1981). *Spatial Statistics*. John Wiley & Sons.

Wadsworth JL, Tawn JA (2012). "Likelihood-Based Procedures for Threshold Diagnostics and Uncertainty in Extreme Value Modelling." *Journal of the Royal Statistical Society B*, **74**(3), 543–567.

Affiliation:

Ana C. Cebrián
Departamento de Métodos Estadísticos

University of Zaragoza
Spain
Telephone: +34/976762885
E-mail: acebrian@unizar.es

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 64, Issue 6 *Submitted:* 2013-05-07
March 2015 *Accepted:* 2014-07-15