Temporal ETAS automatic - Tutorial

Francesco Serafini

2022-09-29

Aim

This tutorial aims to illustrate how to efficiently use our method to implement a temporal ETAS model. Our goal is to give to researcher the possibility of using our temporal ETAS implementation with minimum effort and to be able to retrieve informations from the model in an easy and accessible way. We provided a number of functions to accomplish the basic tasks required for a Bayesian analysis of an ETAS model. Indeed we provide functions to fit the model, retrieve the posterior distribution of the parameters, sampling from the posterior distribution of the parameters, retrieve the posterior distribution of the number of events, produce catalogue-based forecast and extract informations from the forecasts such as posterior predictive intervals of the number of events for each period.

The input file

The present code works taking as input a `txt` file provided by the user. For the present example we have called our file `user_input.txt`.

The rows of the input file can be placed in any order but the names of the variable should always remain fixed. The input file is divided in six sections: Sequence parameters, Priors, Initial values, Inlabru options, Time binning parameters, Forecast, and Output name. Below we report the arguments needed in each section with a brief explanation of their meaning.

Sequence parameters

The sequence parameters needs the following arguments:

- **start.date** = starting date of the catalogue with format "yyyy-mm-dd hh:mm:ss". All events recorded before this date will be discarded.
- **end.date** = end date of the catalogue with format "yyyy-mm-dd hh:mm:ss". All events recorded after this date will be discarded.
- **magnitude.completeness** = magnitude of completeness for the catalogue. All events with magnitude greater than this value will be discarded.
- **min.longitude** = minimum longitude coordinate. All events with longitude smaller than this value will be discarded.
- **max.longitude** = maximum longitude coordinate. All events with longitude greater than this value will be discarded.
- **min.latitude** = minimum latitude coordinate. All events with latitude smaller than this value will be discarded.
- **max.latitude** = maximum latitude coordinate. All events with latitude greater than this value will be discarded.
- **catalog.path** = path of the catalogue file, it must be in `txt` format
- **catalog.header** = TRUE or FALSE indicating if the first row of the provided catalogue contains the name of the columns
- **catalog.sep** = string representing the columns separator

- **catalog.skip** = integer indicating the number of lines of the data file to skip before beginning to read data
- **catalog.colnames** = vector of column names to be provided in the format `c(col1.name, col2.name, ....)` if `catalog.header = TRUE` this will be ignored.

The provided catalogue needs to be in a specific format. First of all, it has to be a `txt` file. Second, the column of the observed times must be called `time_string` and the times must be formatted using "yyyy-mm-ddThh:mm:ss" or "yyyy-mm-dd hh:mm:ss"; the longitude column must be called `Lon`; the latitude column must be called `Lat`; the magnitude column must be called `magnitudes`. If the provided catalog do not follow the above rules the code will return an error.

Priors

The priors parameters define the prior distribution for each parameter of the temporal ETAS model, namely μ, K, α, c, p. For the time being, the distribution family for each parameter is fixed and the user can only control the parameters of such family. The parameters are assumed to have the following prior distributions

$$\begin{aligned}\mu &\sim \text{Gamma}(a_\mu, b_\mu)\\ K &\sim \text{LogGaus}(a_K, b_K)\\ \alpha &\sim \text{Unif}(a_\alpha, b_\alpha)\\ c &\sim \text{Unif}(a_c, b_c)\\ p &\sim \text{Unif}(a_p, b_p)\end{aligned}$$

The user has to provide the values of $a_\mu, b_\mu, a_K, b_K, a_\alpha, b_\alpha, a_c, b_c, a_p, b_p$.

Initial values

The user has to specify the initial values of the parameters in the INLA internal scale. They will be used as first step in the inlabru algorithm and it is safer if they are set to not represents particular cases. Example of cases to be avoided are values for which one of the parameters is zero or one of the parameters has a particularly high value. The initial values are single digit for each parameter. For example, the parameter `th.mu.init` is the initial value of parameter μ, `th.K.init` for parameter K, and so on.

Inlabru options

These options regulates the inlabru algorithm. We provide the user the possibility to speicify only two parameters:

- **max_iter** = maximum number of iterations for the inlabru algorithm. The number of iterations will be less than this number if the algorithm have converged
- **max_step** = this parameters refers to *how far* the parameter value can jump from one iteration to another. The greater the value the greater the potential jump. Setting a value different from `NULL` prevents the inlabru algorithm to check for convergence and the algorithm will run exactly the number of iterations specified in `max_iter`.

Time binning parameters

This parameters defines the strategy with which the time interval is divided in bins for each observed time. Specifically, the strategy is defined by three parameters $\delta, \Delta > 0$ and n_{max}. For an observed time t_i the bins are defined as follow:

$$t_i, t_i + \Delta, t_i + \Delta(1+\delta), t_i + \Delta(1+\delta)^2, ...$$

The parameter n_{max} is the maximum value assumed by the exponent and regulates the maximum number of bins. If $t_i + \Delta(1+\delta)^{n_{max}} < T_2$ then we have exactly $n_{max} + 2$ bins, which is the maximum number of

bins per observations. The parameter Δ regulates the length of the first bin (and subsequent ones). The parameter δ regulates the increase in length considering additional bins. They have to be provided as follow

- **coef.t** = value of the parameter δ
- **DELTA** = value of the parameter Δ
- **Nmax** = value of the parameter n_{max}

Forecast

The parameters defining the forecasting experiment are:

- **n.periods** = number of periods to be forecasted
- **period.length** = the length of each forecasting period in days
- **start.date.fore** = starting date of the forecasting experiment with format "yyyy-mm-dd hh:mm:ss"
- **magnitude.update** = magnitude at which we split a period. If we record an event at time t_k in a forecasting period with magnitude greater than the provided value, then, the end of the forecasting period is set to be t_k and a new forecasting period is started at from t_k.

Output name

A string containing the name of the output `pdf` document. It does not need to include the `.pdf` at the end, this will be added automatically with the date in which the document has been produced. Given an `output.name` the resulting `pdf` document will be called `output.name.currentdate.pdf`.

Reading the input

We provide a function to automatically read the `txt` input file provided by the user. The only input required by this function is the path of the input file.

```
list.input <- input.file.to.list('user_input.txt')
```

```
## Finish loading & preparing catalog
```

The function returns a list of 17 elements containing:

1. `catalog` = the input catalog as it is provided.
2. `catalog.bru` = the input catalog in the format needed for inlabru
3. `time.int` = the provided start and end date in string format
4. `T12` = the start and end date as number of days from the provided starting date
5. `lat.int` = the provided minimum and maximum latitudes
6. `lon.int` = the provided minimum and maximum longitudes
7. `M0` = the provided magnitude of completeness
8. `link.functions` = a list of functions use to trasform the parameters from the internal scale to the ETAS scale
9. `bru.opt.list` = a list of options for Inlabru, contains also the starting value of the parameters in the internal scale
10. `coef.t` = δ parameter of the binning strategy
11. `delta.t` = Δ parameter of the binning strategy
12. `Nmax` = n_{max} parameter of the binning strategy
13. `n.periods` = the provided number of forecasting periods
14. `period.length` = the provided length of a forecasting period in days
15. `start.date.fore` = the provided starting date of the forecasting experiment
16. `magnitude.update` = the provided magnitude above which a new forecasting period is started.
17. `output.name` = the provided string used as name for the output `pdf` document

It is possible to check the arguments as follows:

```
str(list.input)
```

```
## List of 17
##  $ catalog         :'data.frame':    1137 obs. of  9 variables:
##   ..$ Lon        : num [1:1137] 13.2 13.3 13.2 13.2 13.2 ...
##   ..$ Lat        : num [1:1137] 42.7 42.7 42.8 42.8 42.8 ...
##   ..$ magnitudes : num [1:1137] 5.7 4.5 3.2 3.9 3.6 3.4 3.2 3.7 3.3 3.1 ...
##   ..$ time_string: chr [1:1137] "2016-08-24T01:36:32.00" "2016-08-24T01:37:26.58" "2016-08-24T01:40:4
##   ..$ depth      : num [1:1137] 8.1 9 9.7 9.7 9.7 10.7 10.6 6.6 6.5 10.9 ...
##   ..$ event.id   : int [1:1137] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##   ..$ cat.id     : int [1:1137] 7073641 7073711 7073771 7073781 7073811 7073841 7073921 7073901 7073
##   ..$ time_date  : POSIXct[1:1137], format:  ...
##   ..$ time_diff  : num [1:1137] 0 0.000632 0.002947 0.003552 0.003879 ...
##  $ catalog.bru     :'data.frame':    1137 obs. of  3 variables:
##   ..$ ts         : num [1:1137] 0 0.000632 0.002947 0.003552 0.003879 ...
##   ..$ magnitudes: num [1:1137] 5.7 4.5 3.2 3.9 3.6 3.4 3.2 3.7 3.3 3.1 ...
##   ..$ idx.p      : int [1:1137] 1 2 3 4 5 6 7 8 9 10 ...
##  $ time.int        : POSIXct[1:2], format:  ...
##  $ T12             : num [1:2] 0 357
##  $ lat.int         : num [1:2] 42.4 43.1
##  $ lon.int         : num [1:2] 12.9 13.5
##  $ M0              : num 2.99
##  $ link.functions  :List of 5
##   ..$ mu   :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 783 23 783 49 23 49 783 783
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x56049fd75a68>
##   ..$ K    :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 784 22 784 48 22 48 784 784
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x56049fd75a68>
##   ..$ alpha:function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 785 26 785 57 26 57 785 785
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x56049fd75a68>
##   ..$ c_   :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 786 23 786 46 23 46 786 786
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x56049fd75a68>
##   ..$ p    :function (x)
##   .. ..- attr(*, "srcref")= 'srcref' int [1:8] 787 22 787 45 22 45 787 787
##   .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x56049fd75a68>
##  $ bru.opt.list    :List of 4
##   ..$ bru_verbose : num 3
##   ..$ bru_max_iter: num 100
##   ..$ inla.mode   : chr "experimental"
##   ..$ bru_initial :List of 5
##   .. ..$ th.mu   : num 0.5
##   .. ..$ th.K    : num 0.5
##   .. ..$ th.alpha: num -2
##   .. ..$ th.c    : num -2
##   .. ..$ th.p    : num -2
##  $ coef.t          : num 1
##  $ delta.t         : num 0.1
##  $ Nmax            : num 3
##  $ n.periods       : num 120
##  $ period.length   : num 1
##  $ start.date.fore : chr "2016-08-24 02:00:00 BST"
```

```
##  $ magnitude.update: num 5.5
##  $ output.name     : chr "report_ETAS"
```

Fitting a model

To fit a model we provide a function that reads the input list created in the previous section (output of `input.file.to.list`) and fit the model. The functions returns the fitted model.

```
ETAS.model.fit <- Temporal.ETAS.fit(list.input)
```

```
## Start model fitting
## Start creating grid...
## Finished creating grid, time  2.633106
## Finish model fitting
```

It is convenient for the next passages to create a new list adding the fitted model to the input list. This list will be used as input of all the functions in the next sections.

```
list.output <- append(list.input, list(model.fit = ETAS.model.fit))
```

Parameters' posterior distribution

We provide a function that extract the posterior distribution of the parameters in the ETAS scale taking as input the output of the `input.file.to.list` function. The function can be used as follow:

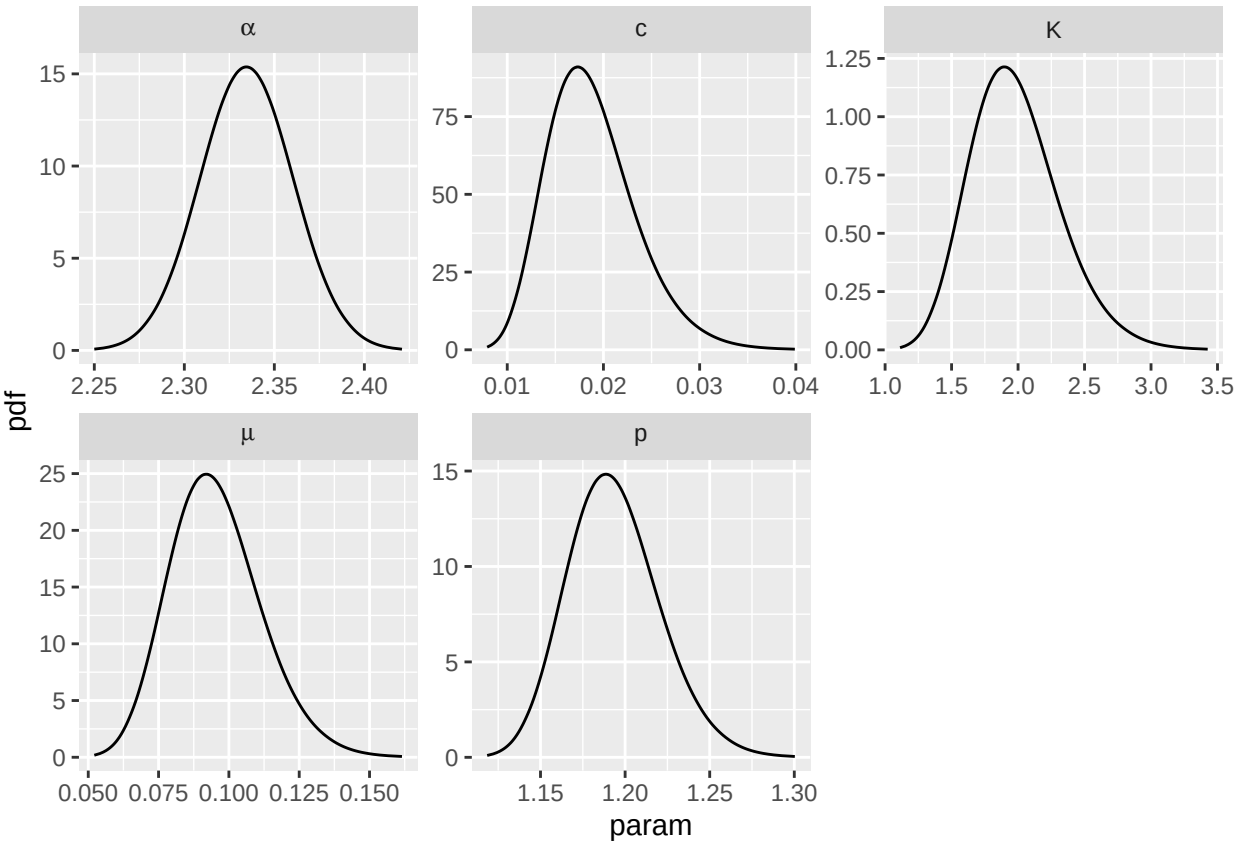
```
post.list <- get_posterior_param(input.list = list.output)
```

The function returns a list of two elements:

- `post.df` = a `data.frame` with three columns: `x` the value of the parameter, `y` the value of the posterior density at `x`, and `param` a string with the parameter name.
- `post.plot` = a `ggplot` object containing a plot of the posteriors

To plot the posterior we only need to run

```
post.list$post.plot
```



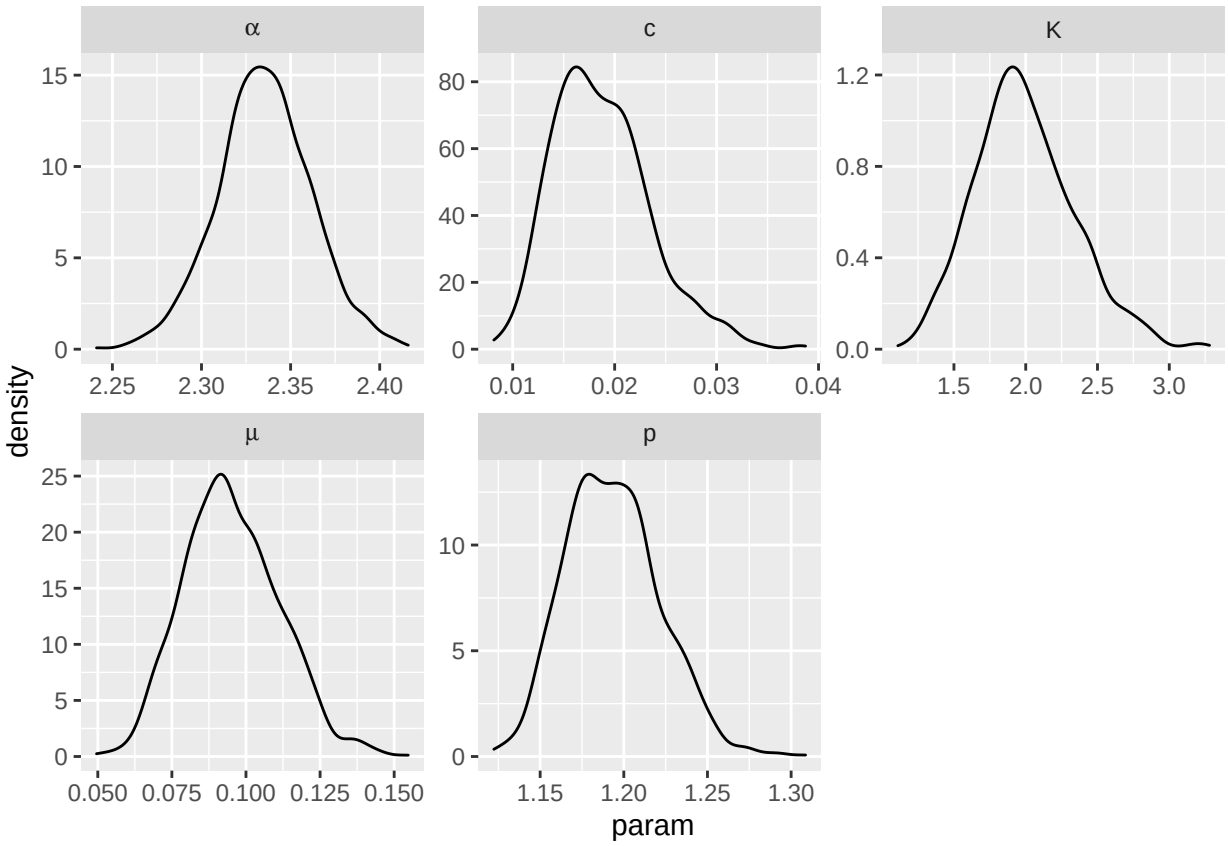
Sampling from the parameters' posterior distribution

We provide a function to sample from the posterior distribution distribution of the ETAS parameters. The function takes in input the output of the function `input.file.to.list` and the number of desired samples. It can be used as follows:

```
p.samp <- post_sampling(input.list = list.output,
                        n.samp = 1000)
```

The function returns a `data.frame` with as many rows as samples from the posterior and columns representing the ETAS parameters. We can plot the empirical distribution obtained from a sample of 1000 values from the parameters posterior to check that they match the true posterior distributions.

```
ggplot(rbind(data.frame(x = p.samp$mu, param = 'mu'),
             data.frame(x = p.samp$K, param = 'K'),
             data.frame(x = p.samp$alpha, param = 'alpha'),
             data.frame(x = p.samp$c, param = 'c'),
             data.frame(x = p.samp$p, param = 'p')), aes(x = x)) +
  geom_density() +
  facet_wrap(facets = vars(param), scales = 'free', labeller = label_parsed)+
  xlab('param')
```



Number of events posterior distribution

We provide a function to calculate the posterior summary statistics of the distribution of the distribution number of events. The number of events regards the time period used to fit the model, therefore, it can be seen as a retrospective test to check if the number of events expected by the model is coherent with the observed one. More specifically, the number of events in a time period (T_1, T_2) has a Poisson distribution

$$N \sim \text{Pois}(\lambda_N)$$

where

$$\lambda_N = \int_{T_1}^{T_2} \lambda(t|\mathcal{H}_t)dt$$

where $\lambda(t|\mathcal{H}_t)$ is the conditional intensity given the history of the process up to time t ($\mathcal{H}_t$). The parameter λ_N has its own posterior distribution because it is a function of the ETAS parameters. This induces a distribution on the space of possible Poisson distributions for N. The functions returns posterior summary statistics of the distribution of the distribution of the number of events.

The function takes in input the output of the function `input.file.to.list`.

```
post.N <- get_posterior_N(list.output)
```

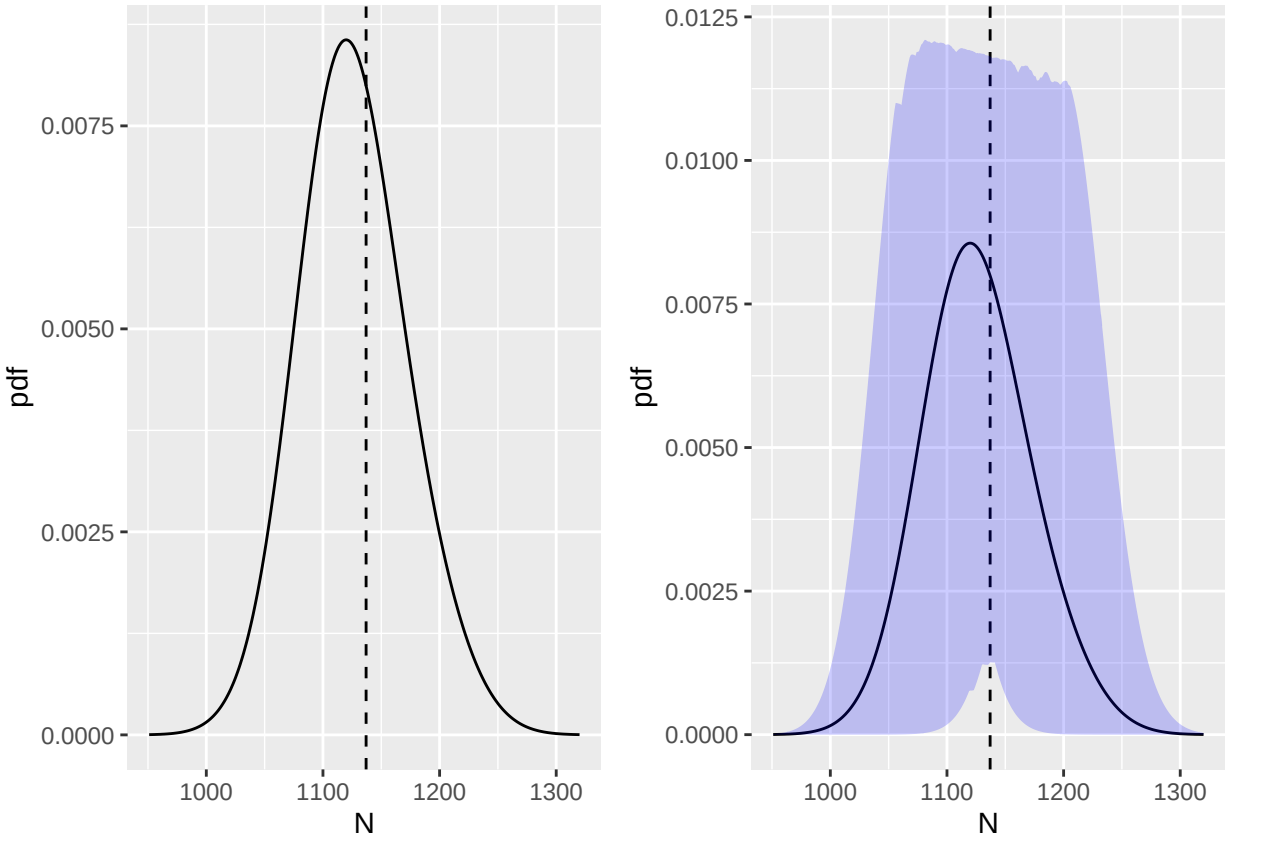
The function returns a list of three objects:

- `post.df` = `data.frame` containing summary posterior informations on the posterior distribution of the distribution of the number of events

- `post.plot` = `ggplot` object containing the mean distribution of the number of events and a vertical dashed line representing the observed number of events
- `post.plot.shaded` = `ggplot` as above but with the 95% credibility interval for the distribution of the number of events

They can be accessed

```
multiplot(post.N$post.plot, post.N$post.plot.shaded, cols = 2)
```



Forecasting experiment

We provide a function to run a forecasting experiment producing catalogue-based forecast of seismicity of a number of periods provided by the user in the input file. Each period has the same length which is also provided by the user and must be in days. Also, we require a magnitude value for which a new forecasting interval is started when an event with magnitude above the provided value is recorded. For each forecasting period, the simulate a number of simulated catalogue equal to the number of parameters' posterior samples provided. Each catalogue is generate using one sample from the posterior of the ETAS parameters. The catalogue-based forecast for each period is stored in a `txt` file with name and path provided by the user.

The function takes in input

- **input.list** = the output of the function `input.file.to.list`.
- **par.sample** = the output of the function `post_sampling`.
- **beta.par** = the parameter of the GR law used for sampling the magnitude ($b = \beta/log(10)$)
- **folder.path** = path of the folder in which the forecasts will be stored
- **fore.name** = string representing the name of each forecast at which will be pasted the period number at which it is referring to.

```
#' Leave the code as it is only if forecasts have already been produced
#' Otherwise move eval = FALSE to the next code chunk if we want to produce forecasts
beta.p <- (1/mean(list.output$catalog.bru$magnitudes - list.output$M0))

prod.fore <- produce.forecast(input.list = list.output,
                              par.sample = p.samp,
                              beta.par = beta.p,
                              folder.path = 'fore_cat/',
                              fore.name = 'fore.day.')
```

The function returns a list with just one element `t.lims` which is a `data.frame` with the extremes of each forecasting period.

If we already have a forecast, then we just need the `t.lims data.frame` to run the remaining code. This can be obtained with the function `find.fore.tlims` which only needs the output of `input.file.to.list` function as input.

```
prod.fore <- find.fore.tlim(list.output)
```

Forecasted number of events

We provide a function to extract information on the number of events distribution in each forecasting period. The function takes in input:

- **input.list** = the output of the function `input.file.to.list`.
- **n.rep** = number of simulated catalogues for each forecasting period. It is equal to the number of rows of the posterior samples dataframe provided above.
- **t.lims** = the output of the function `produce.forecast`.
- **folder.path** = path of the folder in which the forecasts will be stored
- **fore.name** = string representing the name of each forecast at which will be pasted the period number at which it is referring to.

```
fore.N <- summary.forecast.N(input.list = list.output,
                             n.rep = nrow(p.samp),
                             t.lims = prod.fore$t.lims,
                             folder.path = 'fore_cat/',
                             fore.name = 'fore.day.')
```

The function returns a `data.frame` with as many rows as the number of forecasted periods and 4 columns:

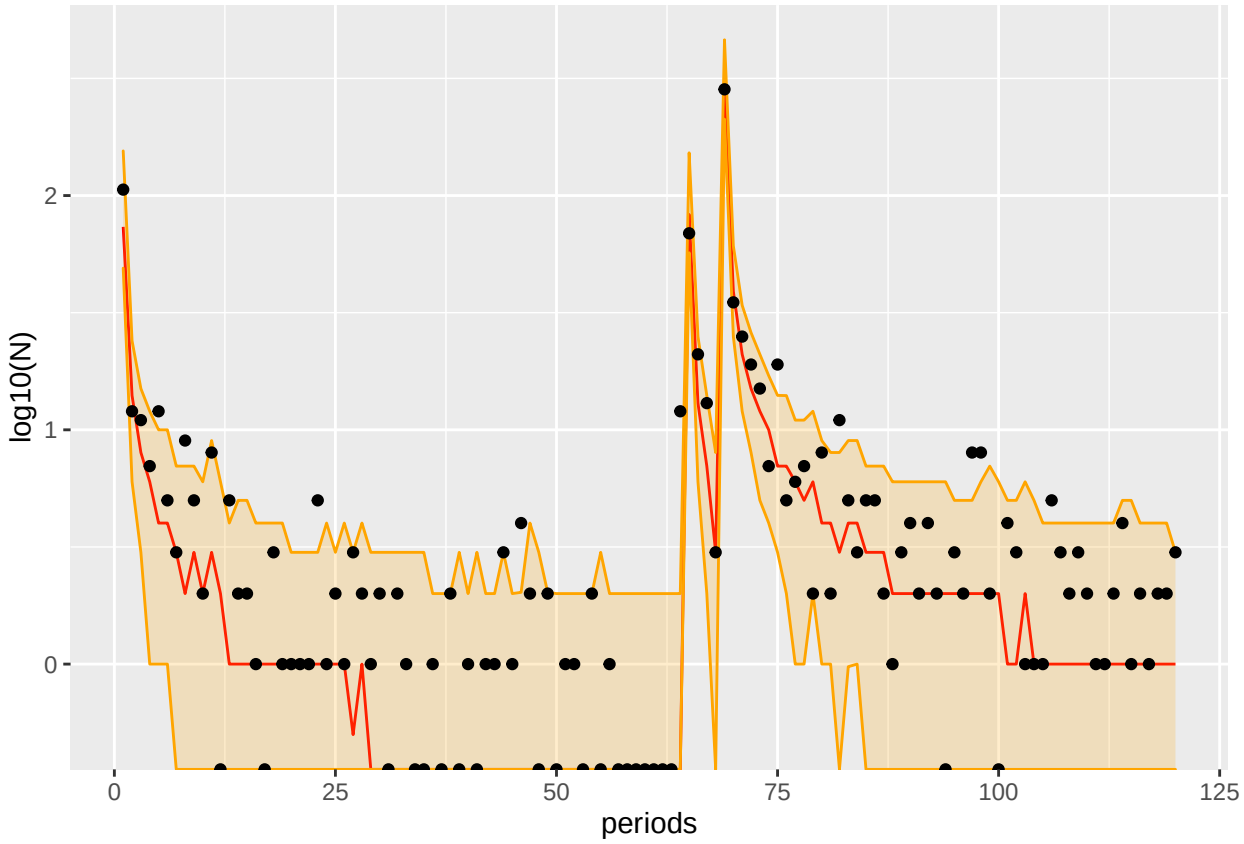
- **q0.025** = 0.025 quantile of the empirical distribution of the number of events
- **median** = median of the empirical distribution of the number of events
- **q0.975** = 0.975 quantile of the empirical distribution of the number of events
- **true** = observed number of events

We can plot this values as follows:

```
#' this plot is in log10 scale, if we want it in the natural scale just remove all the
#' log10 commands from below. Also the scale_y_log10() function does not work very
#' well in this example, I recommend to be avoided.

ggplot(fore.N, aes(x = 1:list.output$n.periods,
                   y = log10(median))) +
  geom_line(color = 'red') +
  geom_ribbon(aes(xmin = 1:list.output$n.periods,
                  xmax = 1:list.output$n.periods,
                  ymin = log10(q0.025),
```

```
                ymax = log10(q0.975)),
            alpha = 0.2, color = 'orange', fill = 'orange') +
  geom_point(aes(x = 1:list.output$n.periods,
                 y = log10(true))) + #, size = 0.5)) +
  #scale_y_log10() +
  xlab('periods') +
  ylab('log10(N)')
```



Produce document

To produce a `pdf` document in output. The function takes as input the name of the current `Rmarkdown` document to render and the `output.name` contained in the input document provided by the user. The option `eval = FALSE` it is crucial in this, if we remove it then we would not be able to render the document. Indeed, the command below needs to be copied and ran from the console. Running it from the code chunk itself will cause an error. The command can also be ran without opening the `Rmarkdown` file, however, in that case \texttt{list.output$output.name} needs to be replaced by a string with the desired name for the document (e.g `'report_ETAS'`).

```
rmarkdown::render('temporal_ETAS_tutorial.Rmd',
                  output_file = paste(list.output$output.name, '.', Sys.Date(),
                                      '.pdf', sep=''))
```