Earthquake Forecasting

Dissertation Project 2

Robin Lin s2435943

July 2023

```r
require(ETAS.inlabru)
require(ggplot2)
require(dplyr)
require(magrittr)
require(tidyquant)
require(rnaturalearth)
require(terra)
require(sf)
require(ggspatial)
require(rnaturalearthdata)
require(lubridate)

# Increase/decrease num.cores if you have more/fewer cores on your computer.
# future::multisession works on both Windows, MacOS, and Linux
num.cores <- 8
future::plan(future::multisession, workers = num.cores)
INLA::inla.setOption(num.threads = num.cores)
# To deactivate parallelism, run
#   future::plan(future::sequential)
#   INLA::inla.setOption(num.threads = 1)
```

Copula transformation of the priors

```r
# set copula transformations list
link.f <- list(
  mu = \(x) gamma_t(x, 0.3, 0.6),
  K = \(x) unif_t(x, 0, 10),
  alpha = \(x) unif_t(x, 0, 10),
  c_ = \(x) unif_t(x, 0, 10),
```

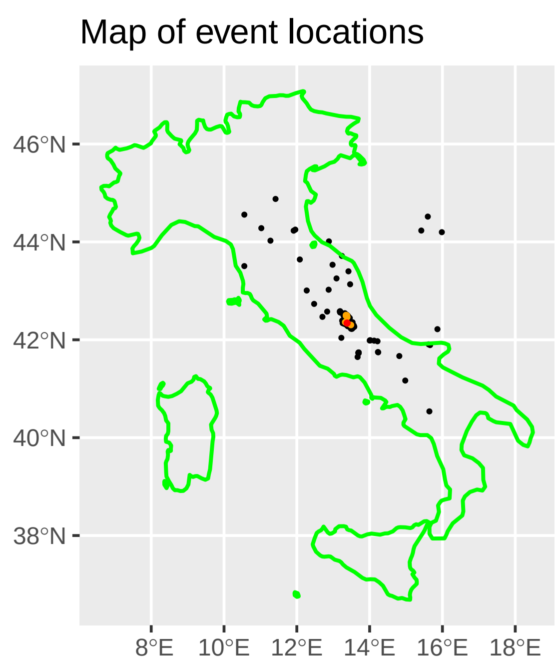
```
  p = \(x) unif_t(x, 1, 10)
)

# set inverse copula transformations list
inv.link.f <- list(
  mu = \(x) inv_gamma_t(x, 0.3, 0.6),
  K = \(x) inv_unif_t(x, 0, 10),
  alpha = \(x) inv_unif_t(x, 0, 10),
  c_ = \(x) inv_unif_t(x, 0, 10),
  p = \(x) inv_unif_t(x, 1, 10)
)
```

Italy

```
# transform time string in Date object
horus$time_date <- as.POSIXct(
  horus$time_string,
  format = "%Y-%m-%dT%H:%M:%OS",
  tz = "UTC"
)
# There may be some incorrectly registered data-times in the original data set,
# that as.POSIXct() can't convert, depending on the system.
# These should ideally be corrected, but for now, we just remove the rows that
# couldn't be converted.
# horus <- na.omit(horus)

# set up parameters for selection
start.date <- as.POSIXct("2009-01-01T00:00:00",
                         format = "%Y-%m-%dT%H:%M:%OS")
end.date <- as.POSIXct("2010-01-01T00:00:00", format = "%Y-%m-%dT%H:%M:%OS")
min.longitude <- 10.5
max.longitude <- 16
min.latitude <- 40.5
max.latitude <- 45
M0 <- 2.5

# set up conditions for selection
aquila.sel <- (horus$time_date >= start.date) &
  (horus$time_date < end.date) &
  (horus$lon >= min.longitude) &
  (horus$lon <= max.longitude) &
  (horus$lat >= min.latitude) &
```

```r
  (horus$lat <= max.latitude) &
  (horus$M >= M0)

# select
aquila <- horus[aquila.sel, ]
```

```r
italy.map <- ne_countries(country = 'Italy', returnclass = "sf",
                          scale = 'medium')

aquila.sf <- st_as_sf(aquila,
                      coords = c("lon", "lat"),
                      crs = st_crs('EPSG:4326'))
ggplot() +
  geom_sf(data = aquila.sf[aquila$M > 3,], size = 0.4) +
  geom_sf(data = italy.map, fill = alpha("lightgrey", 0), color = 'green',
          linewidth = 0.7) +
  geom_sf(data = aquila.sf[aquila$M > 5,], size = 0.5, color = 'orange') +
  geom_sf(data = aquila.sf[aquila$M > 6,], size = 0.6, color = 'red') +
  ggtitle("Map of event locations")
```



```r
ggplot(aquila, aes(time_date, M)) +
  geom_point() +
  theme_bw()
```

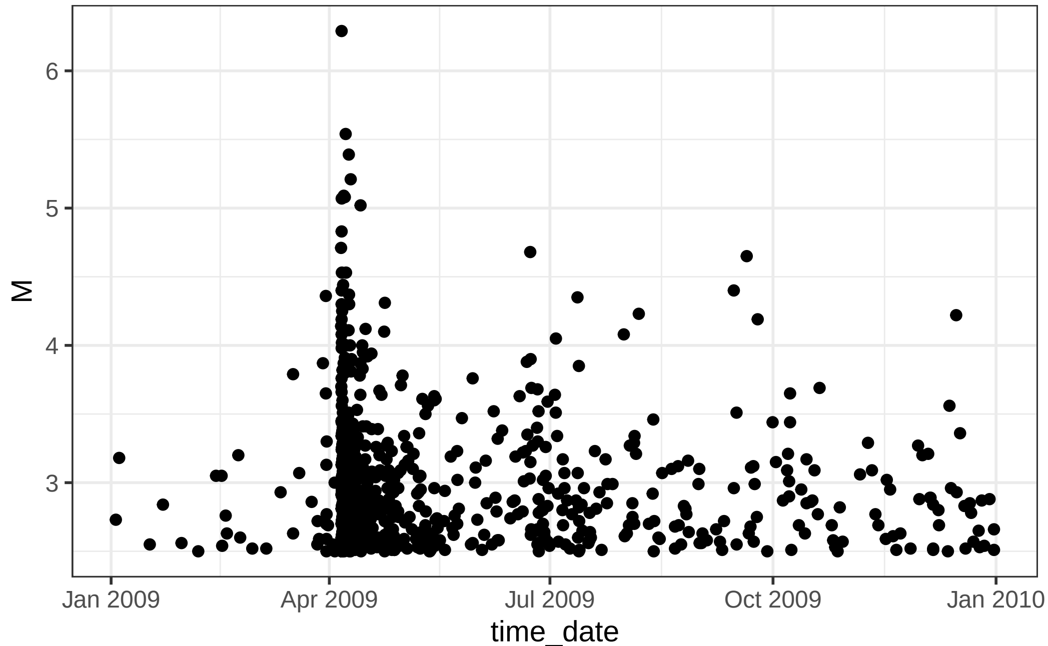


Figure 1: L'Aquila seismic sequence, times versus magnitudes

```r
# set up data.frame for model fitting
aquila.bru <- data.frame(
  ts = as.numeric(
    difftime(aquila$time_date, start.date, units = "days")
  ),
  magnitudes = aquila$M,
  idx.p = 1 : nrow(aquila)
)
```

```r
# set up list of initial values
th.init <- list(
  th.mu = inv.link.f$mu(0.5),
  th.K = inv.link.f$K(0.1),
  th.alpha = inv.link.f$alpha(1),
  th.c = inv.link.f$c_(0.1),
```

```r
  th.p = inv.link.f$p(1.1)
)
```

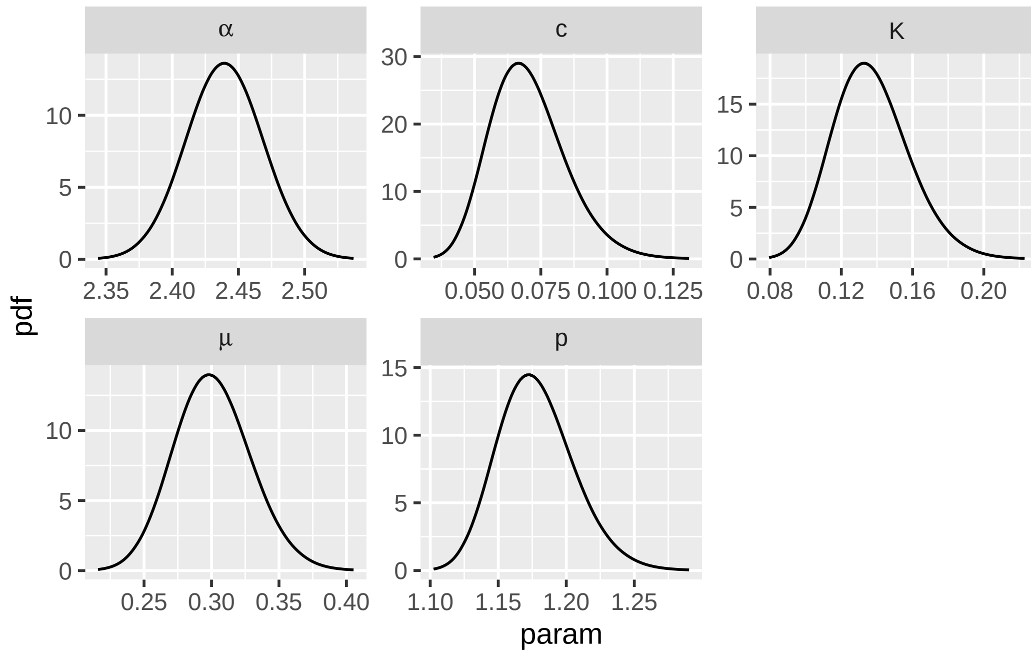
```r
# set up list of bru options
bru.opt.list <- list(
  bru_verbose = 3, # type of visual output
  bru_max_iter = 70, # maximum number of iterations
  # bru_method = list(max_step = 0.5),
  bru_initial = th.init # parameters' initial values
)
```

```r
# set starting and time of the time interval used for model fitting. In this case,
# we use the interval covered by the data.
T1 <- 0
T2 <- max(aquila.bru$ts) + 0.2 # Use max(..., na.rm = TRUE) if there may still be
# NAs here
# fit the model
aquila.fit <- Temporal.ETAS(
  total.data = aquila.bru,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  link.functions = link.f,
  coef.t. = 1,
  delta.t. = 0.1,
  N.max. = 5,
  bru.opt = bru.opt.list
)
```

```
Start creating grid...
Finished creating grid, time  3.370399
```

```r
# create input list to explore model output
input_list <- list(
  model.fit = aquila.fit,
  link.functions = link.f
)
```

```r
# get marginal posterior information
post.list <- get_posterior_param(input.list = input_list)

# plot marginal posteriors
post.list$post.plot
```




```r
post.samp <- post_sampling(
  input.list = input_list,
  n.samp = 1000,
  max.batch = 1000,
  ncore = num.cores
)
```

```
Warning: The `ncore` argument of `post_sampling()` is deprecated as of ETAS.inlabru
1.1.1.9001.
i Please use future::plan(future::multisession, workers = ncore) in your code
  instead.
```

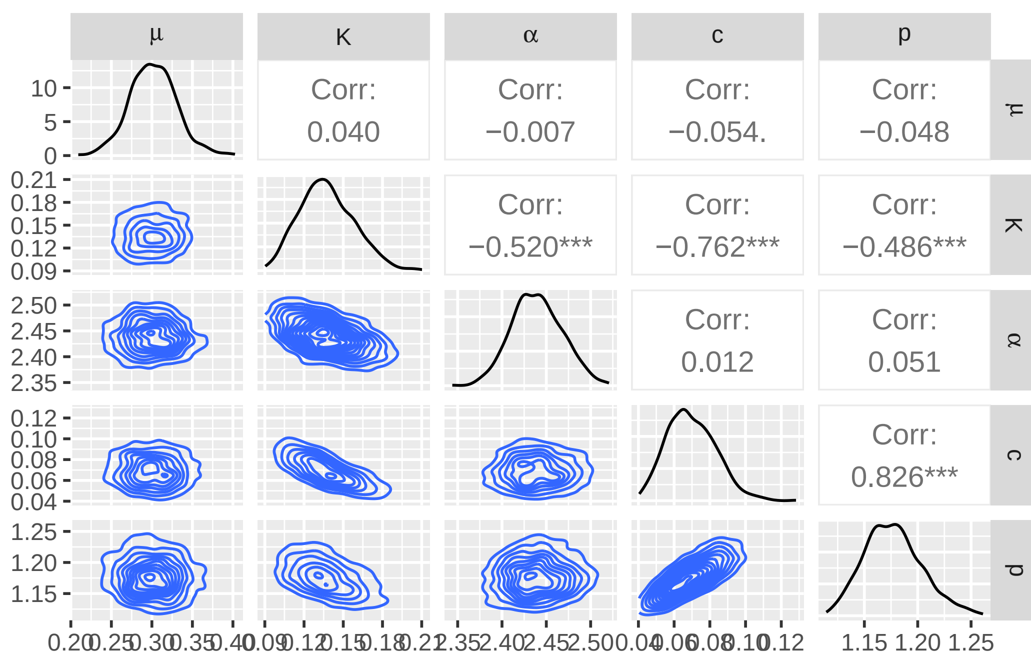
```r
head(post.samp)
```

```
         mu         K    alpha          c        p
1 0.3056772 0.1391004 2.464493 0.06523322 1.196130
2 0.3257255 0.1312240 2.484702 0.06451594 1.188393
3 0.3045494 0.1322441 2.439568 0.07248708 1.183810
4 0.3179632 0.1397335 2.465885 0.05662846 1.167155
5 0.3632828 0.1448948 2.425989 0.05823957 1.147717
6 0.3214774 0.1310820 2.443953 0.07707418 1.197168
```

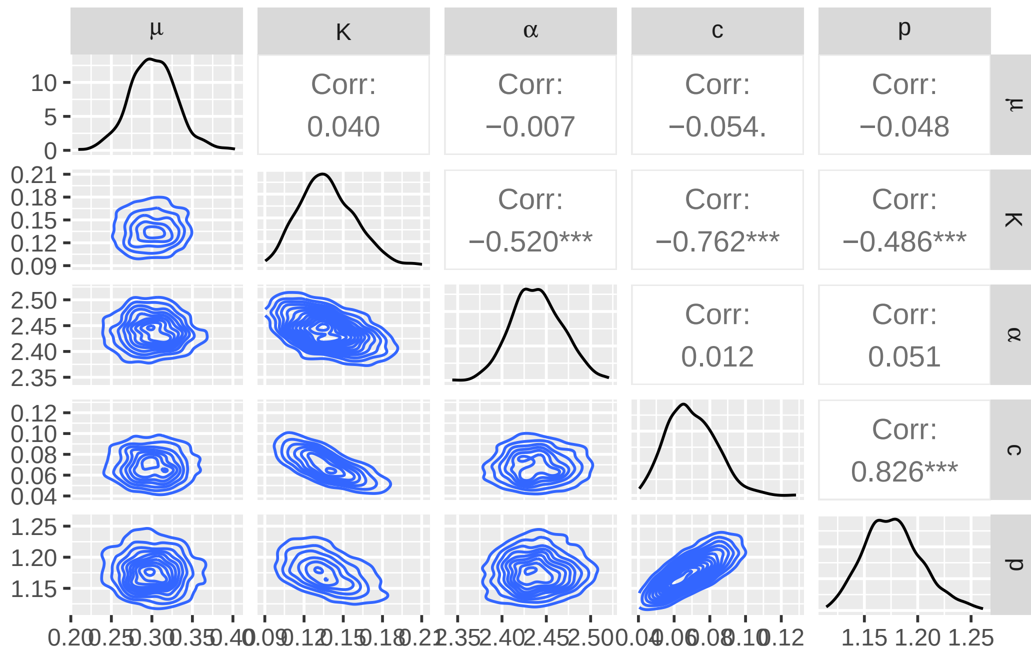
```r
pair.plot <- post_pairs_plot(
  post.samp = post.samp,
  input.list = NULL,
  n.samp = NULL,
  max.batch = 1000
)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

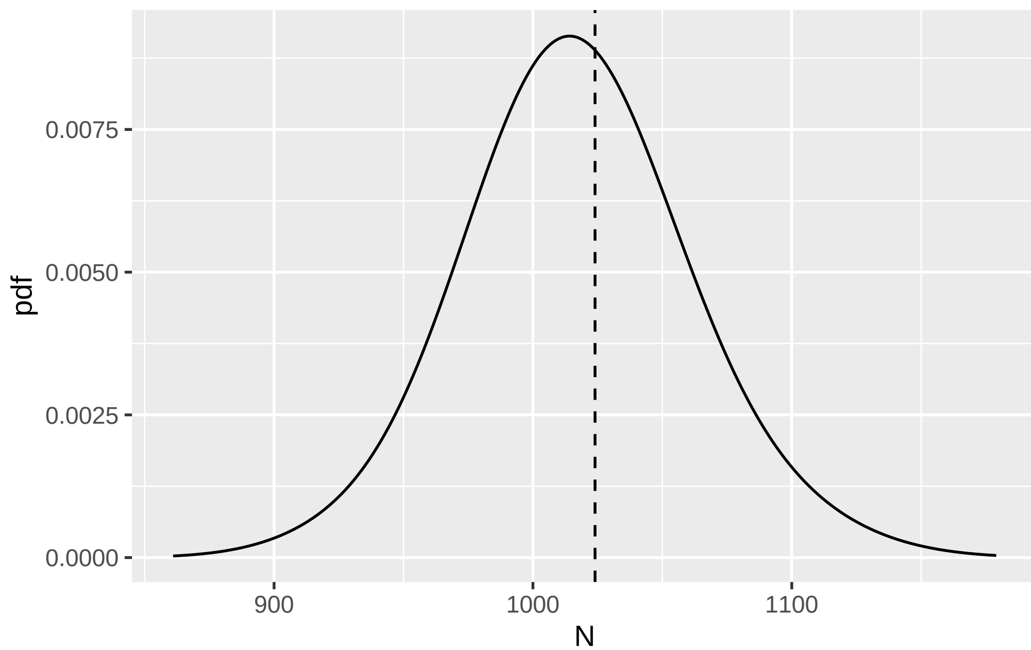
```r
pair.plot$pair.plot
```



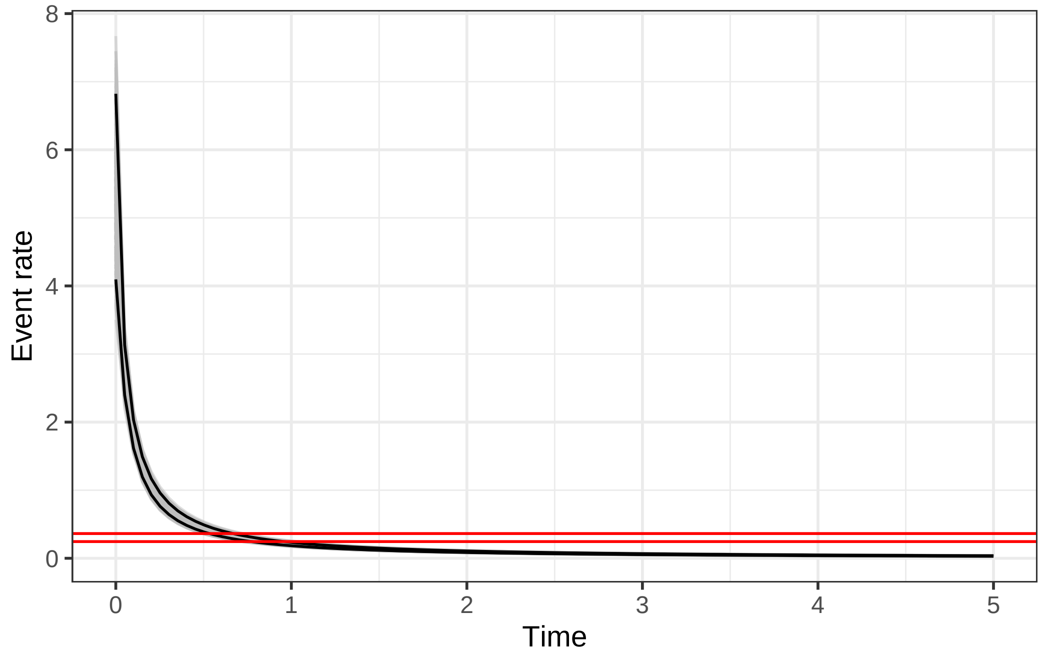
```
pair.plot <- post_pairs_plot(
  post.samp = post.samp,
  input.list = NULL,
  n.samp = NULL,
  max.batch = 1000
)
pair.plot$pair.plot
```



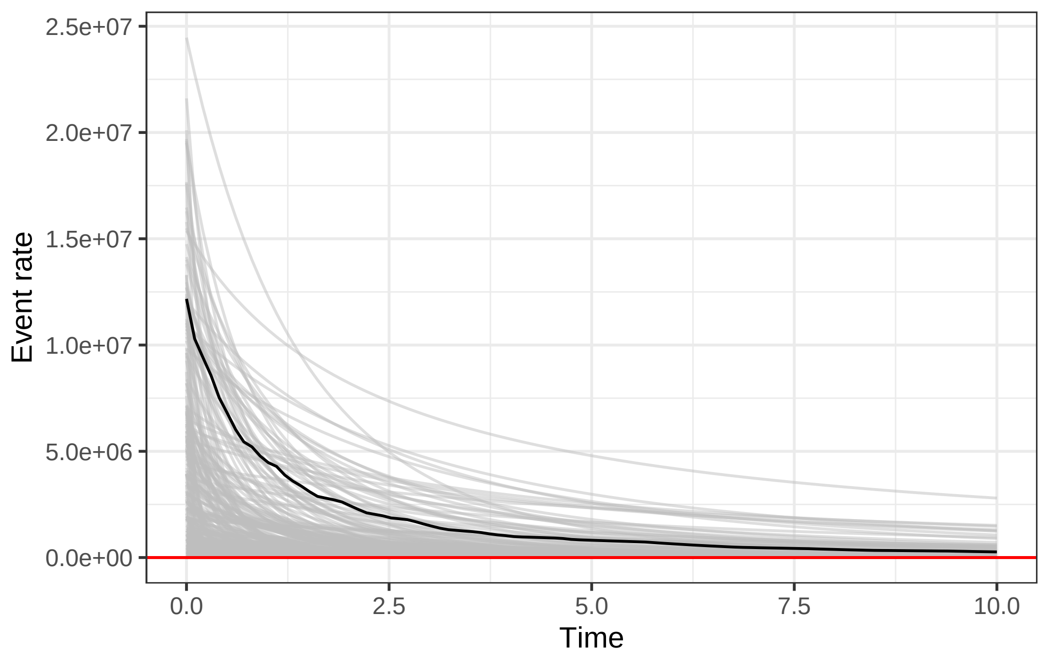

```
# set additional elements of the list
input_list$T12 <- c(T1, T2)
input_list$M0 <- M0
input_list$catalog.bru <- aquila.bru
N.post <- get_posterior_N(input.list = input_list)
N.post$post.plot
```



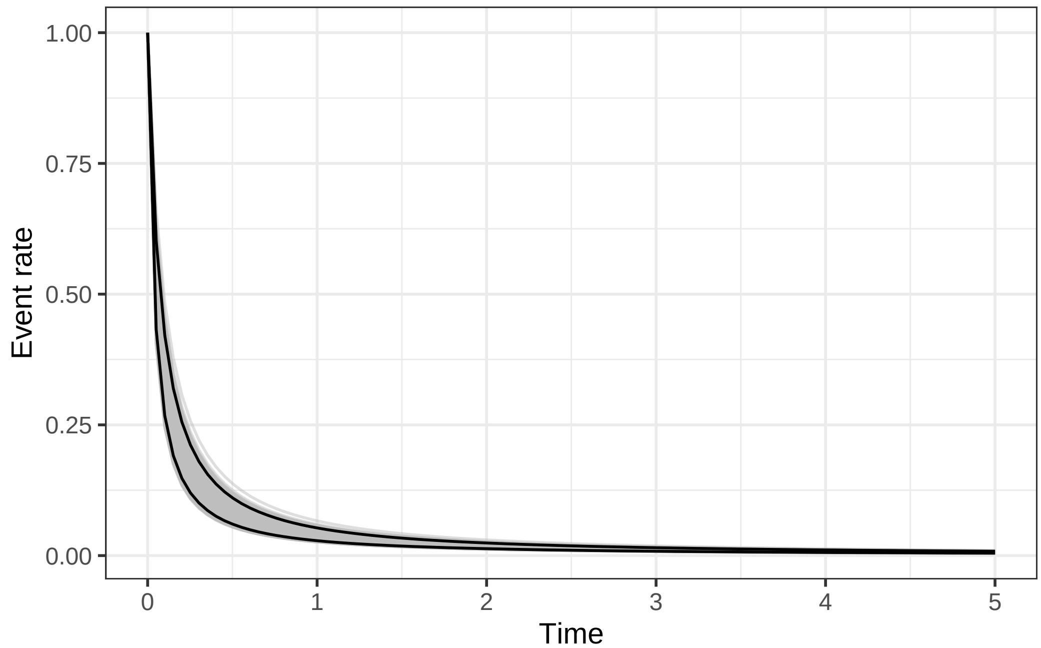

```
triggering_fun_plot(
  input.list = input_list,
  post.samp = post.samp,
  n.samp = NULL, magnitude = 4,
  t.end = 5, n.breaks = 100
)
```



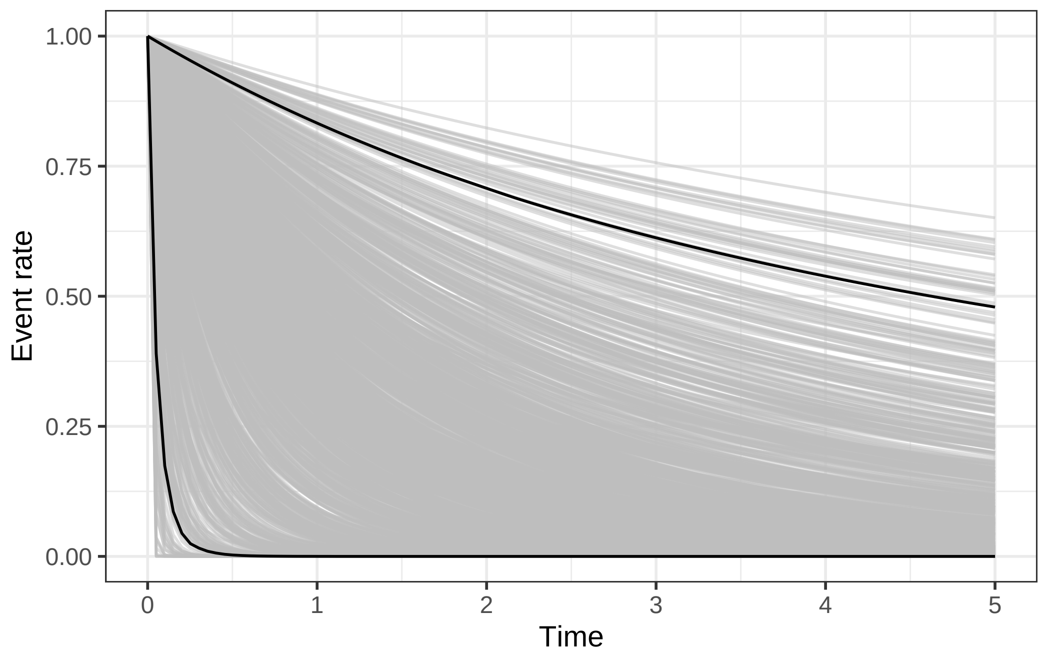

```
triggering_fun_plot_prior(input.list = input_list, magnitude = 4, n.samp = 1000, t.end = 1
```



```
omori_plot_posterior(input.list = input_list, post.samp = post.samp, n.samp = NULL, t.end
```




```
omori_plot_prior(input.list = input_list, n.samp = 1000, t.end = 5)
```

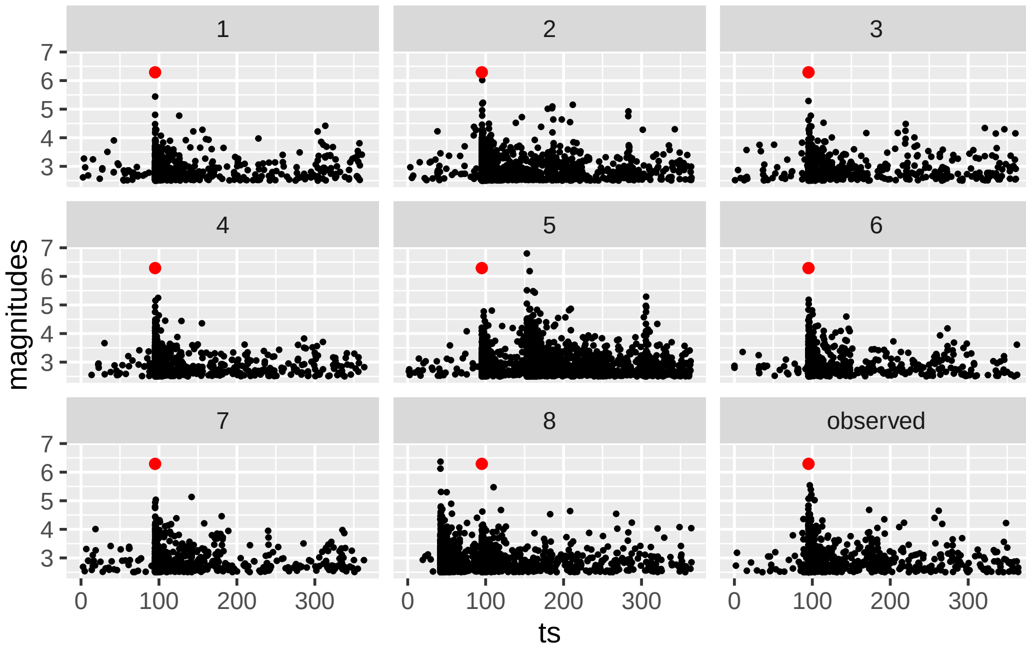


Synthetic catalogues generation

```r
# maximum likelihood estimator for beta
beta.p <- 1 / (mean(aquila.bru$magnitudes) - M0)
```

```r
set.seed(2)
n.cat <- 8
# generate catalogues as list of lists
multi.synth.cat.list <- lapply(seq_len(n.cat), \(x)
generate_temporal_ETAS_synthetic(
  theta = post.samp[x, ],
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  Ht = aquila.bru[which.max(aquila.bru$magnitudes), ]
))

# store catalogues as list of data.frames
multi.synth.cat.list.df <- lapply(multi.synth.cat.list, \(x) do.call(rbind, x))
# set catalogue identifier
```

```r
multi.synth.cat.list.df <- lapply(seq_len(n.cat), \(x) cbind(multi.synth.cat.list.df[[x]],
  cat.idx = x
))
# merge catalogues in unique data.frame
multi.synth.cat.df <- do.call(rbind, multi.synth.cat.list.df)

# we need to bing the synthetics with the observed catalogue for plotting
cat.df.for.plotting <- rbind(
  multi.synth.cat.df,
  cbind(aquila.bru[, c("ts", "magnitudes")],
    gen = NA,
    cat.idx = "observed"
  )
)

# plot them
ggplot(cat.df.for.plotting, aes(ts, magnitudes)) +
  geom_point(size = 0.5) +
  geom_point(
    data = aquila.bru[which.max(aquila.bru$magnitudes), ],
    mapping = aes(ts, magnitudes),
    color = "red"
  ) +
  facet_wrap(facets = ~cat.idx)
```



Forecasting

```
# express 1 minute in days
min.in.days <- 1 / (24 * 60)
# find time of the event with the greatest magnitude
t.max.mag <- aquila.bru$ts[which.max(aquila.bru$magnitudes)]
# set starting time of the forecasting period
T1.fore <- t.max.mag + min.in.days
# set forecast length
fore.length <- 1
# set end time of the forecasting period
T2.fore <- T1.fore + fore.length
# set known data
Ht.fore <- aquila.bru[aquila.bru$ts < T1.fore, ]

# produce forecast
daily.fore <- Temporal.ETAS.forecast(
  post.samp = post.samp, # ETAS parameters posterior samples
  n.cat = nrow(post.samp), # number of synthetic catalogues
  beta.p = beta.p, # magnitude distribution parameter
  M0 = M0, # cutoff magnitude
  T1 = T1.fore, # forecast starting time
```

```r
  T2 = T2.fore, # forecast end time
  Ht = Ht.fore, # known events
  ncore = num.cores
) # number of cores
```

```r
# find number of events per catalogue
N.fore <- vapply(
  seq_len(daily.fore$n.cat),
  \(x) sum(daily.fore$fore.df$cat.idx == x), 0
)
# find number of observed events in the forecasting period
N.obs <- sum(aquila.bru$ts >= T1.fore & aquila.bru$ts <= T2.fore)
# plot the distribution
ggplot() +
  geom_histogram(aes(x = N.fore, y = after_stat(density)), binwidth = 1) +
  geom_vline(xintercept = N.obs) +
  xlim(100, 500)
```