Impact of Varied High Magnitude Events on alpha Estimation

Author: Zhile Xu

UUN: s2500393

This document explores the impact of varied high-magnitude events on the estimation of the alpha parameter in the ETAS model using synthetic earthquake catalogues. The analysis includes setting up the environment, generating synthetic data, and evaluating the ETAS model's performance.

Set Up

```r
# Remove all objects from the current R environment to start fresh
rm(list = ls())
# Set seed for reproducibility of random number generation
set.seed(123)
```

```r
# Load necessary libraries
library(ETAS.inlabru)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
##
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(ggridges)
library(ggdist)
```

```
##
## Attaching package: 'ggdist'
##
## The following objects are masked from 'package:ggridges':
##
##     scale_point_color_continuous, scale_point_color_discrete,
##     scale_point_colour_continuous, scale_point_colour_discrete,
##     scale_point_fill_continuous, scale_point_fill_discrete,
##     scale_point_size_continuous
```

```
# Set up parallel processing
num.cores <- 12
future::plan(future::multisession, workers = num.cores)
INLA::inla.setOption(num.threads = num.cores)
```

Set True ETAS Parameters

```
# set true parameters
true.param <- list(
  mu = 0.30106014, K = 0.13611399, alpha = 2.43945301,
  c = 0.07098607, p = 1.17838741
)
# create a dataframe of true parameters
df.true.param <- data.frame(
  x = unlist(true.param),
  param = names(true.param)
)
# set magnitude distribution parameter
beta.p <- 2.353157
# set cutoff magnitude
M0 <- 2.5
# set starting time of the synthetic catalogue
T1 <- 0
# set end time of the synthetic catalogue
T2 <- 365
```

```
# a function to generate synthetic catalogues
generate_synthetic_catalogue <- function(
    theta,
    beta.p,
    M0,
    T1,
    T2,
```

```r
    idx = NULL,
    Ht = NULL) {
  # generate the catalogue-1 - it returns a list of data.frames
  synth.cat.list <- generate_temporal_ETAS_synthetic(
    theta = theta,
    beta.p = beta.p,
    M0 = M0,
    T1 = T1,
    T2 = T2,
    Ht = Ht
  )
  synth.cat.df <- do.call(rbind, synth.cat.list)
  synth.cat.df$idx <- as.integer(idx)
  return(synth.cat.df)
}
```

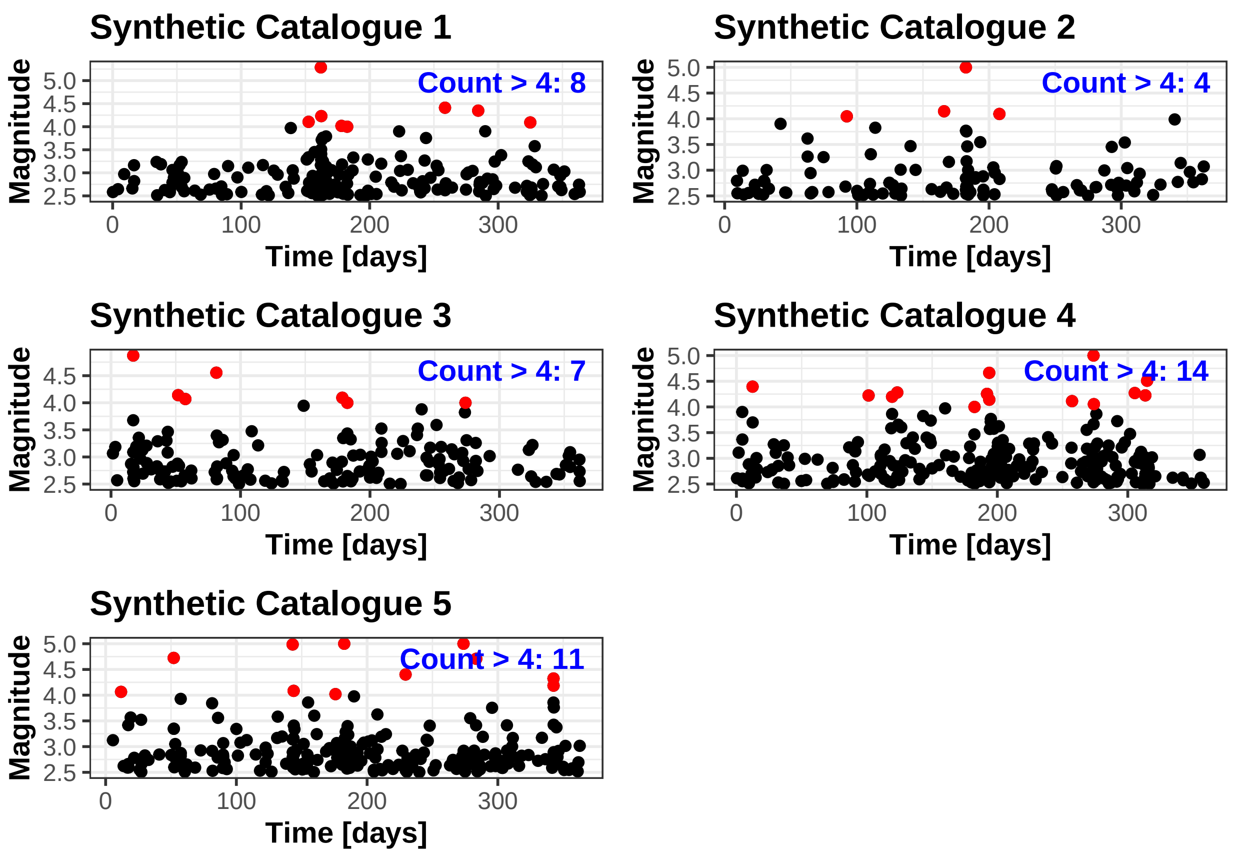
Generate Synthetic Catalogues

```r
# create a dataframe containing the known events
known.events.df.1 <- data.frame(
  ts = c(T2 * 0.5),
  magnitudes = c(4)
)
# gerate the synthetic catalogue-1
synth.cat.df.1 <- generate_synthetic_catalogue(
  theta = true.param,
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  idx = 1,
  Ht = known.events.df.1
)
# create a dataframe containing the known events
known.events.df.2 <- data.frame(
  ts = c(T2 * 0.5),
  magnitudes = c(5)
)
# generate the catalogue-2
synth.cat.df.2 <- generate_synthetic_catalogue(
  theta = true.param,
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  idx = 2,
  Ht = known.events.df.2
)
# create a dataframe containing the known events
known.events.df.3 <- data.frame(
  ts = c(T2 * 0.5, T2 * 0.75),
  magnitudes = c(4, 4)
)
# generate the catalogue-3
synth.cat.df.3 <- generate_synthetic_catalogue(
  theta = true.param,
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  idx = 3,
  Ht = known.events.df.3
)
# create a dataframe containing the known events
known.events.df.4 <- data.frame(
  ts = c(T2 * 0.5, T2 * 0.75),
  magnitudes = c(4, 5)
)
# generate the catalogue-4
synth.cat.df.4 <- generate_synthetic_catalogue(
  theta = true.param,
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  idx = 4,
  Ht = known.events.df.4
)
# create a dataframe containing the known events
known.events.df.5 <- data.frame(
  ts = c(T2 * 0.5, T2 * 0.75),
  magnitudes = c(5, 5)
)
# generate the catalogue-5
synth.cat.df.5 <- generate_synthetic_catalogue(
  theta = true.param,
  beta.p = beta.p,
  M0 = M0,
  T1 = T1,
  T2 = T2,
  idx = 5,
  Ht = known.events.df.5
)
synth.cat.list <- list(
  synth.cat.df.1,
  synth.cat.df.2,
  synth.cat.df.3,
  synth.cat.df.4,
  synth.cat.df.5
)
```

Plot Synthetic Catalogues to Visualize the Distributions of Magnitudes

```r
# create a function to plot the synthetic catalogues
plot_synthetic_catalogue <- function(synth.cat.df, title) {
  count_greater_than_4 <- sum(synth.cat.df$magnitudes >= 4)
  plt <- ggplot(synth.cat.df, aes(x = ts, y = magnitudes)) +
    geom_point() +
    # mark the events with magnitude greater than 4
    geom_point(
      data = synth.cat.df[synth.cat.df$magnitudes >= 4, ],
      aes(x = ts, y = magnitudes), color = "red"
    ) +
    theme_bw() +
    theme(
      # text = element_text(size = 14),
      plot.title = element_text(face = "bold"),
      legend.position = "bottom",
      strip.background = element_rect(fill = "white", color = "black"),
      axis.title = element_text(face = "bold")
    ) +
    labs(x = "Time [days]", y = "Magnitude") +
    labs(title = title) +
    # Add the count of points with magnitudes greater than 4 to the title
    annotate("text",
      x = Inf, y = Inf,
      label = paste("Count > 4:", count_greater_than_4),
      hjust = 1.1, vjust = 1.5,
      size = 4, color = "blue", fontface = "bold"
    )
  return(plt)
}
```

```r
# Generate plots
plots <- lapply(1:length(synth.cat.list), function(i) {
  title <- paste("Synthetic Catalogue", i)
  plot_synthetic_catalogue(synth.cat.list[[i]], title)
})
# Arrange the plots in a column
plots <- grid.arrange(grobs = plots, ncol = 2)
```



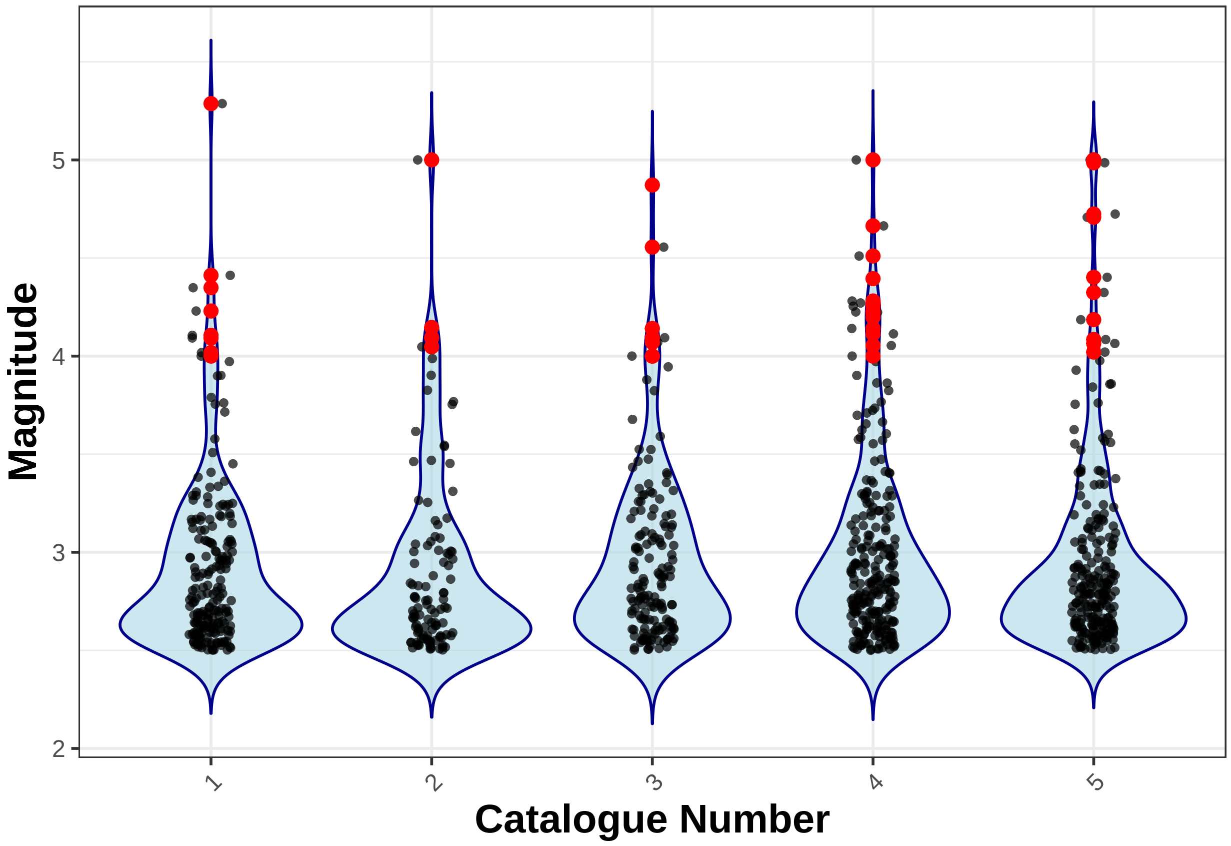
```
ggsave("q1-alpha-catalogue.png",
  plot = plots, width = 11.69,
  height = 8.27, units = "in", dpi = 300
)
```

Prepare the Data for Model Fitting

```
# Sort each catalogue by occurrence time and add event identifier
synth.cat.list <- lapply(synth.cat.list, function(df) {
  df <- df[order(df$ts), ]
  df$idx.p <- seq_len(nrow(df))
  return(df)
})
```

```
# create a dataframe containing all catalogues
synth.cat.df <- do.call(rbind, synth.cat.list)
```

```
# draw a viloin plot of the magnitudes
# x is the category identifier
plot.violin <- ggplot(synth.cat.df, aes(x = as.factor(idx), y = magnitudes)) +
  geom_violin(trim = FALSE, fill = "lightblue", color = "darkblue", alpha = 0.6) +
  geom_jitter(width = 0.1, alpha = 0.7, color = "black", size = 1) +
  geom_point(
    data = synth.cat.df[synth.cat.df$magnitudes >= 4, ],
    aes(x = as.factor(idx), y = magnitudes), color = "red", size = 2
  ) +
  theme_bw() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold"),
    axis.title = element_text(face = "bold", size = 15),
    axis.text.x = element_text(angle = 45, hjust = 1)
  ) +
  labs(x = "Catalogue Number", y = "Magnitude") # +
# ggtitle("Violin Plot of Magnitudes")

plot.violin
```



```
ggsave("q1-alpha-violin.png",
  plot = plot.violin, width = 11.69,
  height = 8.27, units = "in", dpi = 300
)
```

Model Fitting

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

```r
  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)
)

# 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),
  th.p = inv.link.f$p(1.1)
)

# set up list of bru options
bru.opt.list <- list(
  bru_verbose = 0, # 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
# create a list of model fitting results
fit.list <- lapply(synth.cat.list, function(synth.cat.df) {
  # fit the model
  fit <- Temporal.ETAS(
    total.data = synth.cat.df,
    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
  )
  print("finish...")
  return(fit)
})
```

```
## Start creating grid...
## Finished creating grid, time  0.206296
## [1] "finish..."
## Start creating grid...
## Finished creating grid, time  0.1168022
## [1] "finish..."
## Start creating grid...
## Finished creating grid, time  0.15504
## [1] "finish..."
## Start creating grid...
## Finished creating grid, time  0.2496831
## [1] "finish..."
## Start creating grid...
## Finished creating grid, time  0.2522659
## [1] "finish..."
```

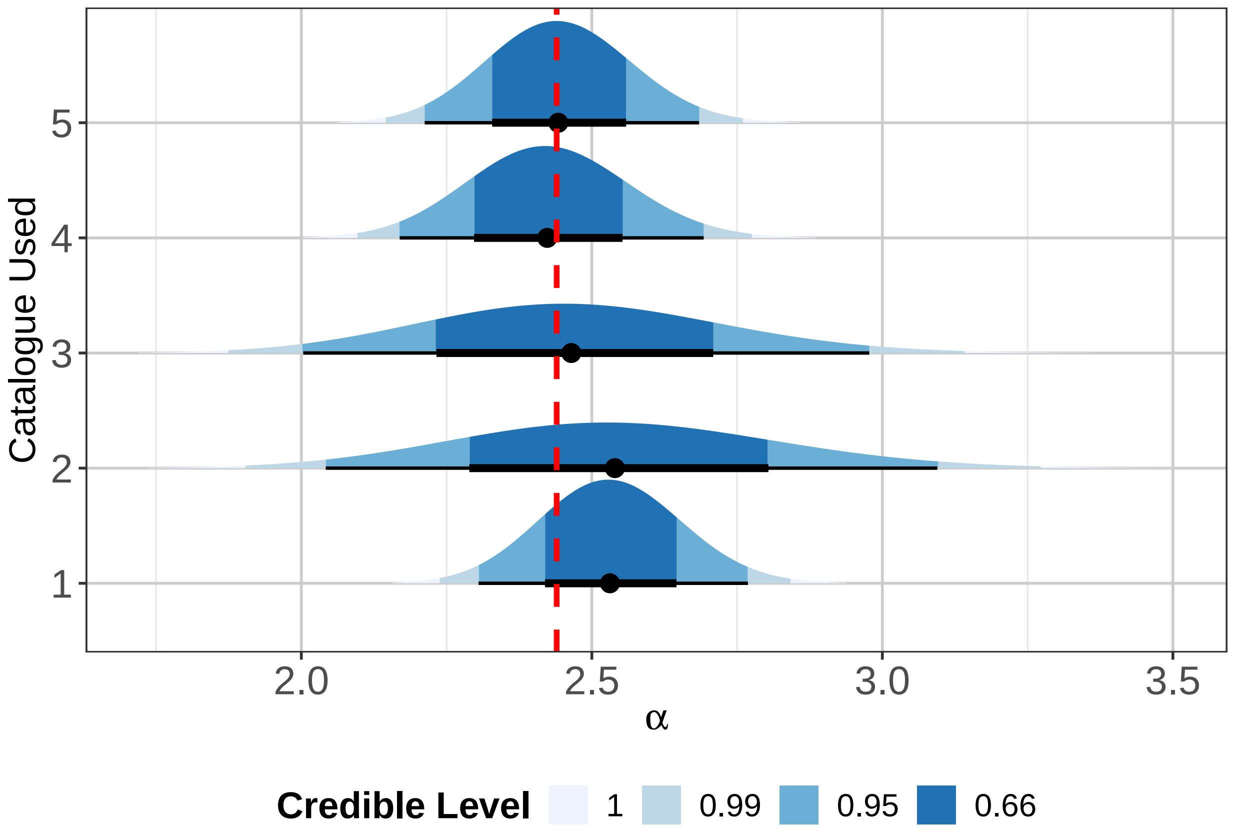
```r
# create a list of model fitting results
input_lists <- lapply(1:length(synth.cat.list), function(i) {
  list(
    model.fit = fit.list[[i]],
    link.functions = link.f
  )
})

# retrieve marginal posterior distributions and set model identifier
post_lists <- lapply(1:length(synth.cat.list), function(i) {
  post.list <- get_posterior_param(input.list = input_lists[[i]])
  post.list$post.df$cat.used <- as.character(i)
  return(post.list)
})

# bind marginal posterior data.frames
bind.post.df <- do.call(rbind, lapply(post_lists, function(x) x$post.df))
```

Plot the Marginal Posteriors (alpha)

```r
# Plot the marginal posteriors of alpha with additional visual enhancements
plot.posters <- ggplot(
  bind.post.df[bind.post.df$param == "alpha", ],
  aes(x = x, y = cat.used)
) +
  stat_slab(aes(fill = after_stat(level)), .width = c(.66, .95, .99, 1)) +
  stat_pointinterval() +
  scale_fill_brewer(
    palette = "Blues", na.translate = FALSE,
    name = "Credible Level"
  ) +
  geom_vline(
    data = df.true.param[df.true.param$param == "alpha", ],
    aes(xintercept = x), linetype = 2, color = "red", size = 1
  ) +
  theme_bw() +
  theme(
    text = element_text(size = 14),
    legend.title = element_text(size = 14, face = "bold"),
    legend.text = element_text(size = 12),
    legend.position = "bottom",
    plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 15),
    panel.grid.major = element_line(color = "grey80"),
    panel.grid.minor = element_line(color = "grey90")
  ) +
  labs(
    x = expression(alpha),
    y = "Catalogue Used",
    # title = "Marginal Posterior Distributions of Alpha",
    fill = "Credible Interval"
  )
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
plot.posters
```



```
ggsave("q1-alpha-posteriors.png",
  plot = plot.posters, width = 11.69,
  height = 8.27, units = "in", dpi = 300
)
```