Earthquake Forecasting

Dissertation Project 2

Robin Lin s2435943

August 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),
```

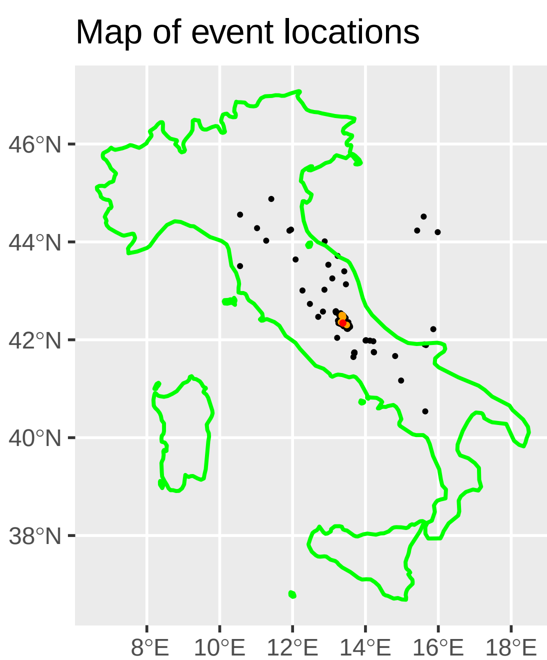
```
  p = \(x) unif_t(x, 1, 10)
)

# 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()
```

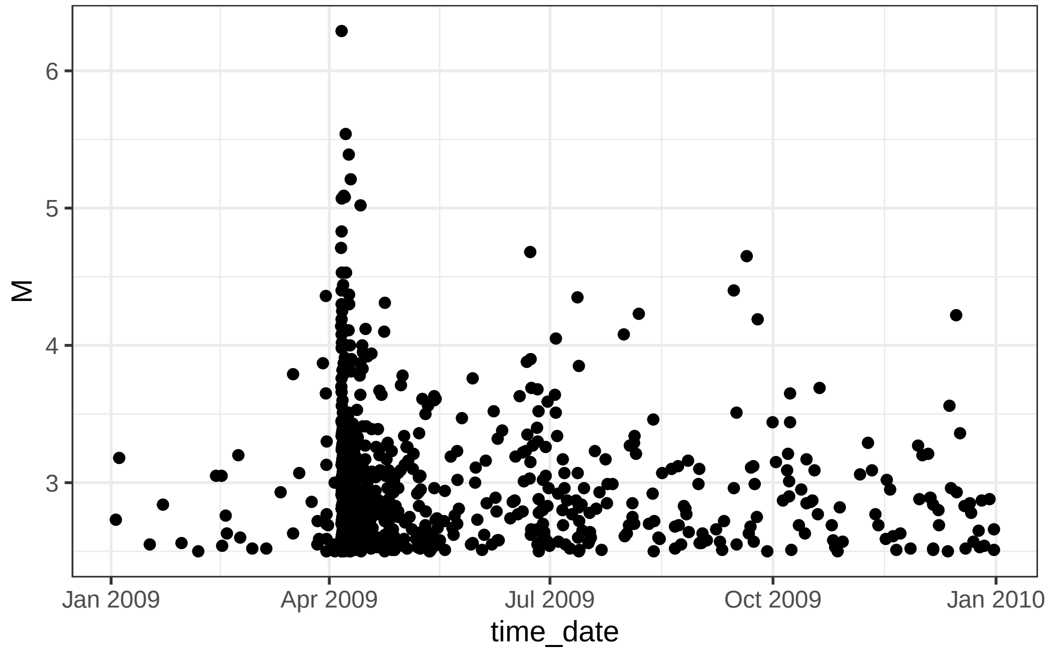


Figure 1: L'Aquila seismic sequence, times versus magnitudes

```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 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
```

```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
ETAS <- function(data = aquila.bru, m0 = M0, t1 = T1, t2 = T2,
                 ncore = num.cores, n.samp = 1000, max.batch = 1000,
                 mag = 5, n.breaks = 100, t.end.tri.post = 5,
                 t.end.tri.prior = 10, t.end.omori.post = 5,
                 t.end.omori.prior = 5){

    # maximum likelihood estimator for beta
    beta.p <- 1 / (mean(data$magnitudes) - m0)

    # fit the model
    model.fit <- Temporal.ETAS(
      total.data = data,
      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
    )

    # create input list to explore model output
```

```r
input_list <- list(
  model.fit = model.fit,
  link.functions = link.f
)

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

# plot marginal posteriors
postplot <- post.list$post.plot

# posterior sampling
post.samp <- post_sampling(
  input.list = input_list,
  n.samp = n.samp,
  max.batch = max.batch,
  ncore = num.cores
)

# taking the averages of the posterior parameter estimates
post.par <- apply(post.samp, 2, mean)

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

# set additional elements of the list
input_list$T12 <- c(t1, t2)
input_list$M0 <- m0
input_list$catalog.bru <- data

# posterior number of events
N.post <- get_posterior_N(input.list = input_list)
Npostplot <- N.post$post.plot
Npostmean <- N.post$post.df[which.max(N.post$post.df$mean), 1]

```

```r
  # number of large events
  large_events <- data[data$magnitudes >= mag,]
  Nlarge <- nrow(large_events)

  # mean absolute distance of the differences in magnitudes
  diff_mag <- diff(data$magnitudes)
  abs_dist_mag <- mean(abs(diff_mag))

  # mean absolute distance of the inter-arrival time
  interarrival <- diff(data$ts)
  abs_dist_int <- mean(abs(interarrival))

  # check if overdispersion occurs
  m_int_time <- mean(interarrival)
  v_int_time <- var(interarrival)
  overdisp <- m_int_time ^ 2 < v_int_time

  # triggering function plots
  # posterior
  triplotpost <- triggering_fun_plot(
    input.list = input_list,
    post.samp = post.samp,
    n.samp = NULL, magnitude = mag,
    t.end = t.end.tri.post, n.breaks = n.breaks
  )

  # prior
  triplotprior <- triggering_fun_plot_prior(input.list = input_list,
                          magnitude = mag, n.samp = n.samp,
                          t.end = t.end.tri.prior)

  # omori plots
  # posterior
  omoripost <- omori_plot_posterior(input.list = input_list,
                     post.samp = post.samp,
                     n.samp = NULL, t.end = t.end.omori.post)

  # prior
  omoriprior <- omori_plot_prior(input.list = input_list, n.samp = n.samp,
                   t.end = t.end.omori.prior)

  # returns the whole environment
```

```r
    envir <- as.list(environment())
    return(tibble::lst(envir))
}
etas <- ETAS()
```

```
Start creating grid...
Finished creating grid, time  4.450089
```

Synthetic catalogues generation

```r
mult.synth.ETAS <- function(t1 = NULL, t2 = NULL, n.cat = 8,
                     ht = etas$envir$data[which.max(
                         etas$envir$data$magnitudes), ],
                     events_threshold = 300){

    # inherits the environment from function `ETAS`
    envir <- etas$envir

    # updates environments if specified by users
    envir$t1 <- ifelse(!is.null(t1), t1, envir$t1)
    envir$t2 <- ifelse(!is.null(t2), t2, envir$t2)

    # generate catalogues as list of lists
    multi.synth.cat.list <- lapply(seq_len(n.cat), \(x)
    generate_temporal_ETAS_synthetic(
      theta = envir$post.samp[x, ], beta.p = envir$beta.p,
      M0 = envir$m0, T1 = envir$t1,
      T2 = envir$t2, Ht = ht, ncore = envir$ncore))

    # 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
    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
```

```r
  cat.df.for.plotting <- rbind(
    multi.synth.cat.df,
    cbind(envir$data[, c("ts", "magnitudes")],
      gen = NA, cat.idx = "observed"
    )
  )

  # plot them
  multi.synth.cat.plot <- ggplot(cat.df.for.plotting,
                                 aes(ts, magnitudes)) +
    geom_point(size = 0.5) +
    geom_point(
      data = ht, mapping = aes(ts, magnitudes), color = "red"
    ) +
    facet_wrap(facets = ~cat.idx)

  # modelling
  post.par <- matrix(rep(0, n.cat * 5), ncol = 5)
  Npostmean <- rep(0, n.cat)
  Nlarge <- rep(0, n.cat)
  abs_dist_int <- rep(0, n.cat)
  abs_dist_mag <- rep(0, n.cat)
  overdisp <- rep(0, n.cat)
  j <- rep(0, n.cat)

  i <- 1
  while(i <= n.cat){
      Nevents <- nrow(multi.synth.cat.df[
          multi.synth.cat.df$cat.idx == i,])
      if(Nevents >= events_threshold){
          multi.synth.etas <- ETAS(data = multi.synth.cat.df[
          multi.synth.cat.df$cat.idx == i,],
                           t1 = envir$t1, t2 = envir$t2)
          post.par[i,] <- multi.synth.etas$envir$post.par
          Npostmean[i] <- multi.synth.etas$envir$Npostmean
          Nlarge[i] <- multi.synth.etas$envir$Nlarge
          abs_dist_int[i] <- multi.synth.etas$envir$abs_dist_int
          abs_dist_mag[i] <- multi.synth.etas$envir$abs_dist_mag
          overdisp[i] <- multi.synth.etas$envir$overdisp
      }else{
          j[i] <- 1
```

```
        }
        i <- i + 1
    }

    # remove the elements not evaluated
    k <- which(j == 1)
    post.par <- post.par[-k,]
    Npostmean <- Npostmean[-k]
    Nlarge <- Nlarge[-k]
    abs_dist_int <- abs_dist_int[-k]
    abs_dist_mag <- abs_dist_mag[-k]
    overdisp <- overdisp[-k]

    # returns the whole environment
    environ <- as.list(environment())
    return(tibble::lst(environ))
}
mult.synth.fit <- mult.synth.ETAS(n.cat = 10, ht = NULL)
```

Forecasting

```
ETAS.forecast <- function(){

    # inherits the environment from function `ETAS`
    envir <- etas$envir

    # express 1 minute in days
    min.in.days <- 1 / (24 * 60)
    # find time of the event with the greatest magnitude
    t.max.mag <- envir$data$ts[which.max(envir$data$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 <- envir$data[envir$data$ts < T1.fore, ]

    # produce forecast
    daily.fore <- Temporal.ETAS.forecast(
```

```
      post.samp = envir$post.samp, # ETAS parameters posterior samples
      n.cat = nrow(envir$post.samp), # number of synthetic catalogues
      beta.p = envir$beta.p, # magnitude distribution parameter
      M0 = envir$m0, # cutoff magnitude
      T1 = T1.fore, # forecast starting time
      T2 = T2.fore, # forecast end time
      Ht = Ht.fore, # known events
      ncore = envir$ncore # number of cores
    )

    # 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(envir$data$ts >= T1.fore & envir$data$ts <= T2.fore)
    # plot the distribution
    histfore <- ggplot() +
      geom_histogram(aes(x = N.fore, y = after_stat(density)),
                     binwidth = 1) +
      geom_vline(xintercept = N.obs) +
      xlim(100, 500)

    return(tibble::lst(N.fore, N.obs, histfore))
}
fore <- ETAS.forecast()
```

```
# save.image()
```