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 <- 1
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)
```

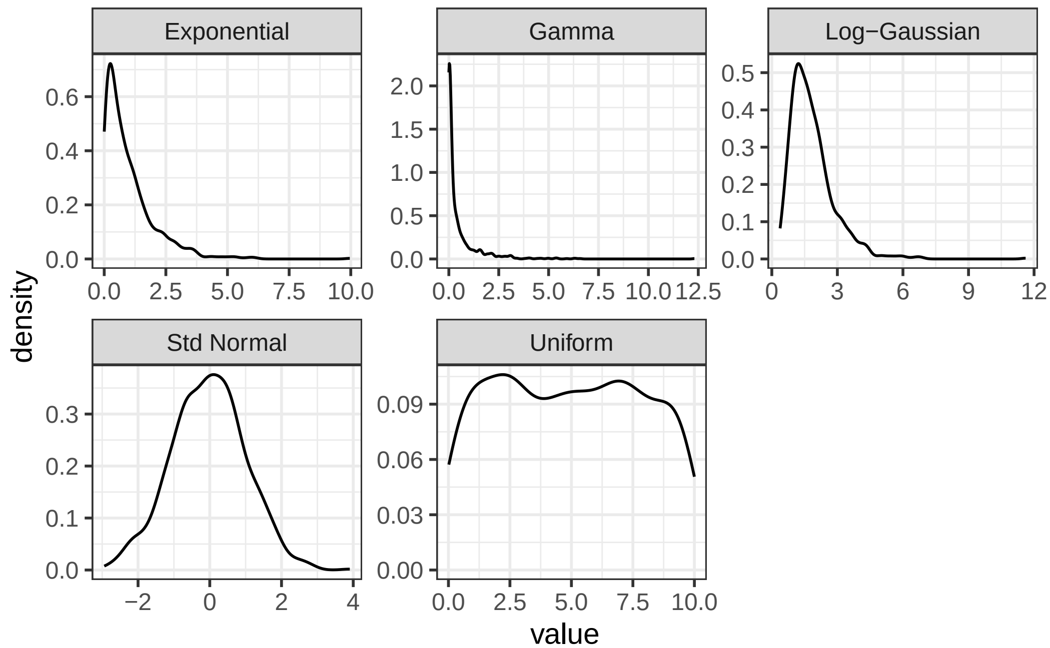
Copula transformation of the priors

```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),
```

```r
  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)
)
```

```r
# obtain sample from standard normal distribution
X <- rnorm(1000)
# apply copula transformations
gamma.X <- gamma_t(X, .3, .6)
unif.X <- unif_t(X, 0, 10)
exp.X <- exp_t(X, 1)
loggaus.X <- loggaus_t(X, .5, .5)

# build data.frame for plotting
df.to.plot <- rbind(
  data.frame(
    value = X,
    distribution = "Std Normal"
  ),
  data.frame(
    value = gamma.X,
    distribution = "Gamma"
  ),
  data.frame(
    value = unif.X,
    distribution = "Uniform"
  ),
  data.frame(
    value = exp.X,
    distribution = "Exponential"
  ),
  data.frame(
    value = loggaus.X,
    distribution = "Log-Gaussian"
```

```r
    )
)
# plot them
ggplot(df.to.plot, aes(value)) +
  geom_density() +
  theme_bw() +
  facet_wrap(facets = ~ distribution, scales = "free")
```

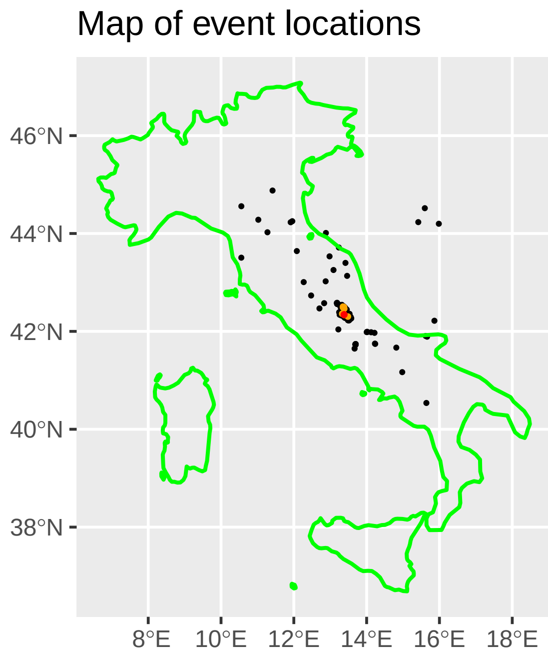


Italy

```r
# 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)
```

```r

# 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$lat >= min.latitude) &
  (horus$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$M > 3,], size = 0.4) +
  geom_sf(data = italy.map, fill = alpha("lightgrey", 0), colour = 'green',
          linewidth = 0.7) +
  geom_sf(data = aquila.sf[aquila$M > 5,], size = 0.5, colour = 'orange') +
  geom_sf(data = aquila.sf[aquila$M > 6,], size = 0.6, colour = 'red') +
  ggtitle("Map of event locations")
```




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

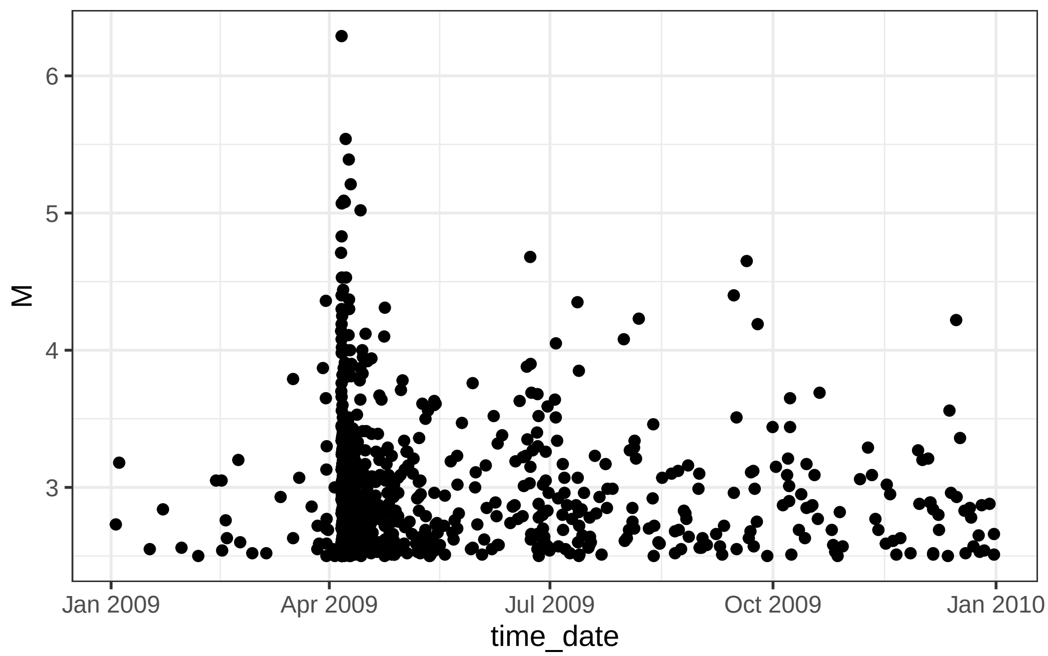


Figure 1: L'Aquila seismic sequence, times versus magnitudes

```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),
  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, Link.f = link.f,
                 Bru.opt.list = bru.opt.list, n.samp = 1000,
                 max.batch = 1000, mag = 4.5, 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
    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]

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

```r

  # 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
  envir <- as.list(environment())
  return(tibble::lst(envir))
```

```r
}
etas <- ETAS()
```

```
Start creating grid...
Finished creating grid, time  4.643445
```

Effect of mis-specifying parameters

```r
# # set copula transformations list
# link.f1 <- 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)
# )
```

Synthetic catalogues generation

```r
mult.synth.ETAS <- function(t1 = NULL, t2 = NULL, n.cat = 1000,
                    ht = etas$envir$data[which.max(
                        etas$envir$data$magnitudes), ]){

    # 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)

    # Function to generate a synthetic catalogue
    synth.gen <- function(i){
        iteration <- i
        synth <- generate_temporal_ETAS_synthetic(
                    theta = envir$post.par %>% as.list,
                    beta.p = envir$beta.p,
                    M0 = envir$m0, T1 = envir$t1,
                    T2 = envir$t2, Ht = ht, ncore = num.cores)
        return(synth)
    }

```

```r
    # generates catalogues as list of lists
    multi.synth.cat.list <- lapply(seq_len(n.cat), \(x)
        synth.gen(x))

    # stores catalogues as list of data.frames
    multi.synth.cat.list.df <- lapply(multi.synth.cat.list,
                                      \(x) do.call(rbind, x))

    # calculates the number of events in each catalogue
    Nevents <- unlist(lapply(seq_len(n.cat), \(i) nrow(
        multi.synth.cat.list.df[[i]])))

    # sets catalogue identifier
    multi.synth.cat.list.df <- lapply(seq_len(n.cat),
                                      \(x) cbind(
                                          multi.synth.cat.list.df[[x]],
                                            cat.idx = x,
                                          num_events = Nevents[x]))

    # merges catalogues in unique data.frame
    multi.synth.cat.df <- do.call(rbind, multi.synth.cat.list.df)

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

Fitting Models on the Synthetic Catalogues

```r
synth.fit <- function(breaks = c(0, 110, 130, 150, 170, 190, 210),
                      samp.each.class = 1){

    # selecting catalogues
    Nevents <- mult.synth$environ$Nevents
    classes <- cut(Nevents, breaks = breaks)
    samp.id <- rep(0, samp.each.class * (classes %>% levels %>% length))
    for(i in classes %>% levels %>% length %>% seq_len){
        samp.id[(samp.each.class * (i - 1) + 1) : (samp.each.class * i)] <-
            sample(which(classes == levels(classes)[i]), samp.each.class)
    }

```

```r
  # we need to bing the synthetics with the observed catalogue
  # for plotting
  cat.df.for.plotting <- rbind(
    mult.synth$environ$multi.synth.cat.df[
      which(
        mult.synth$environ$multi.synth.cat.df$cat.idx %in% samp.id),
      ],
    cbind(mult.synth$environ$envir$data[, c("ts", "magnitudes")],
      gen = NA, cat.idx = "observed", num_events = nrow(etas$envir$data)
    )
  )

  # plot them
  multi.synth.cat.plot <- ggplot(cat.df.for.plotting,
                                 aes(ts, magnitudes)) +
    geom_point(size = 0.5) +
    geom_point(
      data = mult.synth$environ$ht,
      mapping = aes(ts, magnitudes), colour = "black"
    ) +
    facet_wrap(facets = vars(cat.idx, num_events),
               labeller = 'label_both')

  # modelling
  input <- rep(list(NULL), classes %>% levels %>% length)
  post <- rep(list(NULL), classes %>% levels %>% length)
  post.par <- matrix(rep(0, (classes %>% levels %>% length) * 5),
                     ncol = 5)
  Npost <- rep(list(NULL), classes %>% levels %>% length)
  Npostmean <- rep(0, classes %>% levels %>% length)
  Nlarge <- rep(0, classes %>% levels %>% length)
  abs_dist_int <- rep(0, classes %>% levels %>% length)
  abs_dist_mag <- rep(0, classes %>% levels %>% length)
  overdisp <- rep(0, classes %>% levels %>% length)

  for(i in classes %>% levels %>% length %>% seq_len){
      multi.synth.etas <- ETAS(data =
          mult.synth$environ$multi.synth.cat.list.df[[samp.id[i]]],
                               t1 = mult.synth$environ$envir$t1,
                               t2 = mult.synth$environ$envir$t2)
      post[[i]] <- multi.synth.etas$envir$post.list
```

```r
        post.par[i,] <- multi.synth.etas$envir$post.par
        Npost[[i]] <- multi.synth.etas$envir$N.post
        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
    }

    for(i in samp.id %>% length %>% seq_len){
        post[[i]]$post.df$Catalogues <-
        paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
              'Events')
    }

    df.true.param <- data.frame(x = etas$envir$post.par,
                                param = names(etas$envir$post.par %>% as.list))

    # bind marginal posterior data.frames
    bind.post.df <- do.call(rbind,
                            lapply(samp.id %>% length %>% seq_len,
                                   \(i) post[[i]]$post.df))

    # plot them
    post.par.plot <- ggplot(bind.post.df,
                            aes(x = x, y = y, colour = Catalogues)) +
      geom_line() +
      facet_wrap(facets = ~ param, scales = "free") +
      xlab("param") +
      ylab("pdf") +
      geom_vline(
        data = df.true.param,
        mapping = aes(xintercept = x), linetype = 2
      )

    ##
    for(i in samp.id %>% length %>% seq_len){
        Npost[[i]]$post.df$Catalogues <-
        paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
              'Events')
    }
```

```r

    df.true.N <- data.frame(N = etas$envir$Npostmean, param = 'N')

    # bind marginal posterior data.frames
    bind.post.N.df <- do.call(rbind,
                              lapply(samp.id %>% length %>% seq_len,
                                     \(i) Npost[[i]]$post.df))

    # plot them
    post.N.plot <- ggplot(bind.post.N.df,
                          aes(x = N, y = mean, colour = Catalogues)) +
      geom_line() +
      xlab("N") +
      ylab("pdf") +
      geom_vline(
        data = df.true.N,
        mapping = aes(xintercept = N), linetype = 2
      )

    # returns the whole environment
    environ <- as.list(environment())
    return(tibble::lst(environ))
}
mult.synth.fit <- rep(list(NULL), 3)
mult.synth.fit[[1]] <- synth.fit()
```

```
Start creating grid...
Finished creating grid, time  0.2027328
Start creating grid...
Finished creating grid, time  0.2167971
Start creating grid...
Finished creating grid, time  0.344548
Start creating grid...
Finished creating grid, time  0.369961
Start creating grid...
Finished creating grid, time  0.79897
Start creating grid...
Finished creating grid, time  0.9608929
```

```r
mult.synth.fit[[2]] <- synth.fit(breaks =
                                   c(210, 240, 270, 300, 330, 360, 400))
```

```
Start creating grid...
Finished creating grid, time  0.6186411
Start creating grid...
Finished creating grid, time  0.6089611
Start creating grid...
Finished creating grid, time  0.5968621
Start creating grid...
Finished creating grid, time  0.6932831
Start creating grid...
Finished creating grid, time  0.8530071
Start creating grid...
Finished creating grid, time  0.8823462
```

```r
mult.synth.fit[[3]] <- synth.fit(breaks = c(400, 600, 800, 1600))
```

```
Start creating grid...
Finished creating grid, time  1.390939
Start creating grid...
Finished creating grid, time  2.207845
Start creating grid...
Finished creating grid, time  3.399027
```

Analysis on the Behaviours of the Time-between-Events

```r
ECDF.interarrival <- function(i){

    samp.id <- mult.synth.fit[[i]]$environ$samp.id
    Nevents <- mult.synth$environ$Nevents

    data.list <- lapply(samp.id %>% length %>% seq_len, \(i) data.frame(
        Time_between_events =
            mult.synth$environ$multi.synth.cat.list.df[[samp.id[i]]]$ts %>%
                sort %>% diff,
        Num_Events = paste(Nevents[samp.id[i]], '(synthetic)')
        )
    )

    data.list[[length(samp.id) + 1]] <- data.frame(
        Time_between_events = etas$envir$interarrival,
        Num_Events = paste(nrow(etas$envir$data), '(observed)'))

```

```r
    data.list[[length(samp.id) + 2]] <- data.frame(
        Time_between_events =
            rexp(length(etas$envir$interarrival),
                 1 / etas$envir$m_int_time),
        Num_Events = paste(nrow(etas$envir$data),
                           '(under exponential assumption)'))

    df <- do.call(rbind, data.list)

    ECDF.plot <- ggplot(df, aes(x = Time_between_events,
                                colour = Num_Events)) +
        stat_ecdf() +
        xlab('Time between Events') +
      ylab('Empirical Cumulative Probability')

    return(ECDF.plot)
}

ecdf_interarrival <- lapply(seq_len(3), \(i) ECDF.interarrival(i))
```

Forecasting

```r
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
```

```r
    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 = num.cores # 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()
```

```r
# save.image(file = 'Robin_new.RData')
```