

# Earthquake Forecasting

## Dissertation Project 2

Robin Lin s2435943

August 2023

```
1 require(tidyverse)
2 require(knitr)
3 require(kableExtra)
4 require(factoextra)
5 require(cluster)
6 require(ETAS.inlabru)
7 require(ggplot2)
8 require(dplyr)
9 require(magrittr)
10 require(tidyquant)
11 require(rnaturalearth)
12 require(terra)
13 require(sf)
14 require(ggspatial)
15 require(rnaturalearthdata)
16 require(lubridate)
17
18 # Increase/decrease num.cores if you have more/fewer cores on your computer.
19 # future::multisession works on both Windows, MacOS, and Linux
20 num.cores <- 1
21 future::plan(future::multisession, workers = num.cores)
22 INLA::inla.setOption(num.threads = num.cores)
23 # To deactivate parallelism, run
24 # future::plan(future::sequential)
25 # INLA::inla.setOption(num.threads = 1)
```

Copula transformation of the priors

```

1  # set copula transformations list
2  link.f <- list(
3    mu = \ (x) gamma_t(x, 0.3, 0.6),
4    K = \ (x) unif_t(x, 0, 10),
5    alpha = \ (x) unif_t(x, 0, 10),
6    c_ = \ (x) unif_t(x, 0, 10),
7    p = \ (x) unif_t(x, 1, 10)
8  )
9
10 # set inverse copula transformations list
11 inv.link.f <- list(
12   mu = \ (x) inv_gamma_t(x, 0.3, 0.6),
13   K = \ (x) inv_unif_t(x, 0, 10),
14   alpha = \ (x) inv_unif_t(x, 0, 10),
15   c_ = \ (x) inv_unif_t(x, 0, 10),
16   p = \ (x) inv_unif_t(x, 1, 10)
17 )

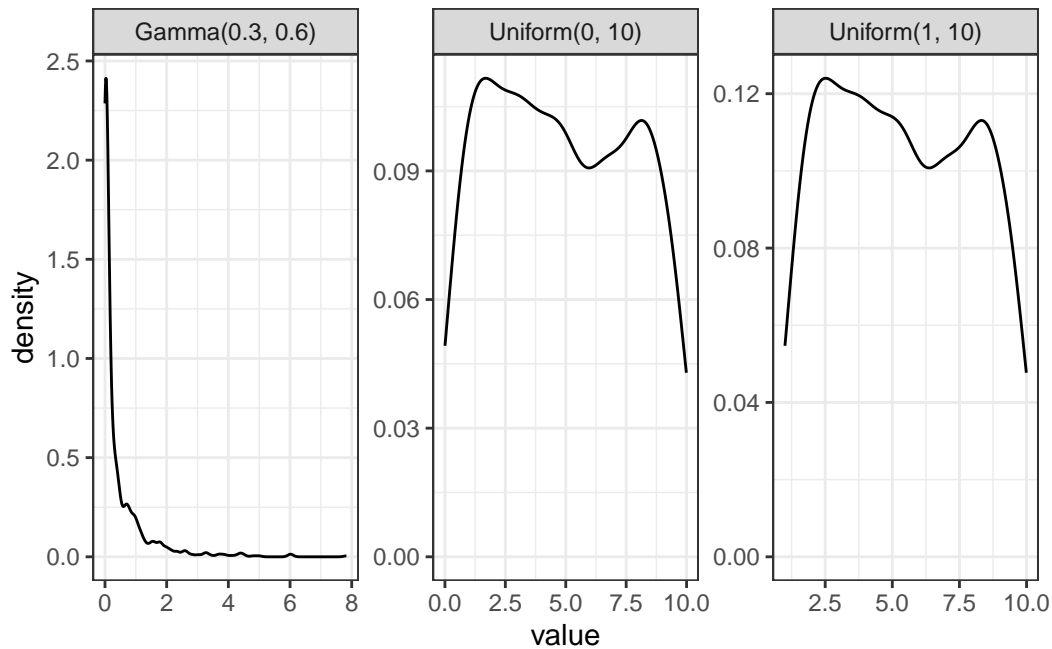
1  # obtain sample from standard normal distribution
2  X <- rnorm(1000)
3  # apply copula transformations
4  gamma.X <- gamma_t(X, 0.3, 0.6)
5  unif.X <- unif_t(X, 0, 10)
6  unif.X.2 <- unif_t(X, 1, 10)
7
8  # build data.frame for plotting
9  df.to.plot <- rbind(
10    data.frame(
11      value = gamma.X,
12      distribution = "Gamma(0.3, 0.6)"
13    ),
14    data.frame(
15      value = unif.X,
16      distribution = "Uniform(0, 10)"
17    ),
18    data.frame(
19      value = unif.X.2,
20      distribution = "Uniform(1, 10)"
21    )
22 )

```

```

1 # plot them
2 ggplot(df.to.plot, aes(value)) +
3   geom_density() +
4   theme_bw() +
5   facet_wrap(facets = ~ distribution, scales = "free")

```



Italy

```

1 # transform time string in Date object
2 horus$time_date <- as.POSIXct(
3   horus$time_string,
4   format = "%Y-%m-%dT%H:%M:%OS",
5   tz = "UTC"
6 )
7 # There may be some incorrectly registered data-times in the original data set,
8 # that as.POSIXct() can't convert, depending on the system.
9 # These should ideally be corrected, but for now, we just remove the rows that
10 # couldn't be converted.
11 # horus <- na.omit(horus)
12
13 # set up parameters for selection

```

```

14 start.date <- as.POSIXct("2009-01-01T00:00:00",
15                           format = "%Y-%m-%dT%H:%M:%OS")
16 end.date <- as.POSIXct("2010-01-01T00:00:00", format = "%Y-%m-%dT%H:%M:%OS")
17 min.longitude <- 10.5
18 max.longitude <- 16
19 min.latitude <- 40.5
20 max.latitude <- 45
21 M0 <- 2.5
22
23 # set up conditions for selection
24 aquila.sel <- (horus$time_date >= start.date) &
25   (horus$time_date < end.date) &
26   (horus$lon >= min.longitude) &
27   (horus$lon <= max.longitude) &
28   (horus$lat >= min.latitude) &
29   (horus$lat <= max.latitude) &
30   (horus$M >= M0)
31
32 # select
33 aquila <- horus[aquila.sel, ]

1 italy.map <- ne_countries(country = 'Italy', returnclass = "sf",
2                           scale = 'medium')
3
4 aquila.sf <- st_as_sf(aquila,
5                      coords = c("lon", "lat"),
6                      crs = st_crs('EPSG:4326'))

1 ggplot() +
2   geom_sf(data = aquila.sf[aquila$M > 3,], size = 0.8) +
3   geom_sf(data = italy.map, fill = alpha("lightgrey", 0), colour = 'green',
4           linewidth = 0.7) +
5   geom_sf(data = aquila.sf[aquila$M > 5,], size = 0.9, colour = 'orange') +
6   geom_sf(data = aquila.sf[aquila$M > 6,], size = 1, colour = 'red') +
7   ggtitle("Map of event locations")

```

Map of event locations

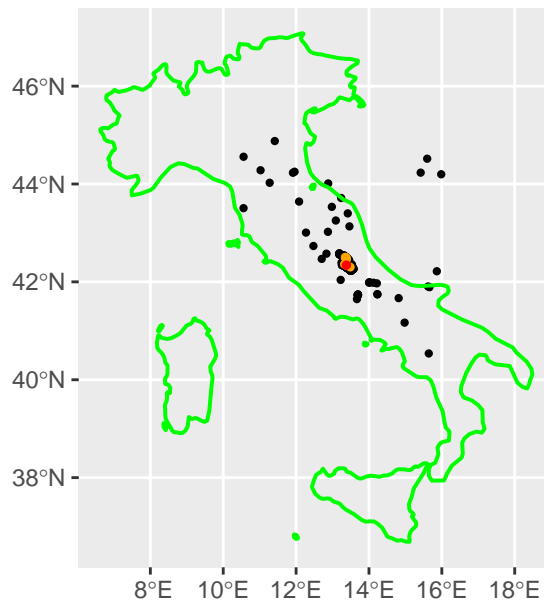


Figure 1: Italian earthquakes in 2009

```
1 ggplot(aquila, aes(time_date, M)) +  
2   geom_point() +  
3   theme_bw()
```

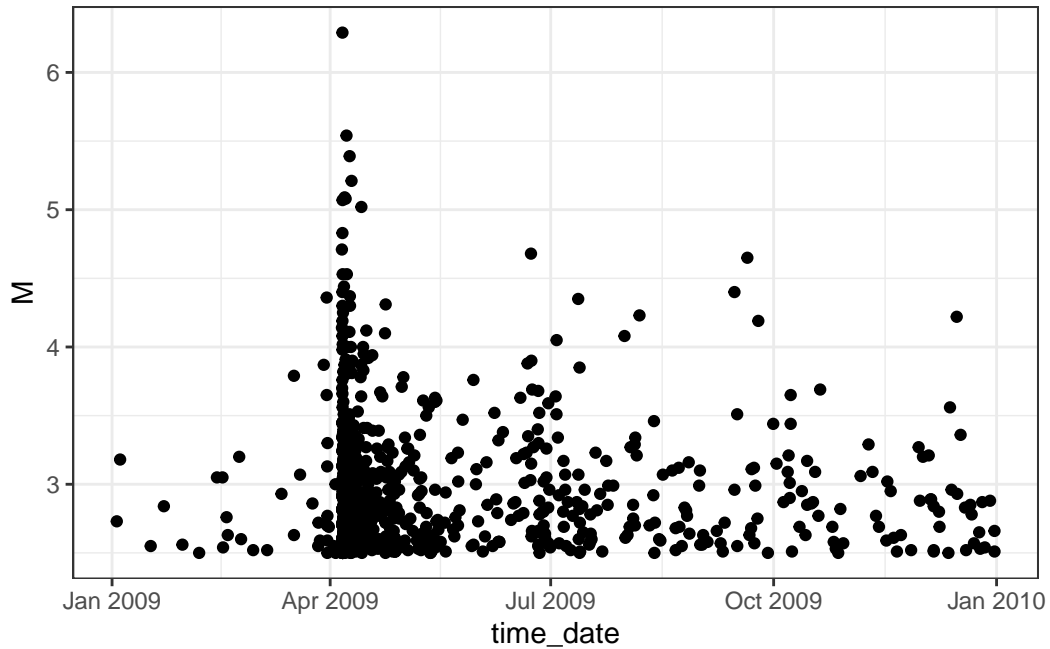


Figure 2: L'Aquila seismic sequence, times versus magnitudes

```

1  # set up data.frame for model fitting
2  aquila.bru <- data.frame(
3    ts = as.numeric(
4      difftime(aquila$time_date, start.date, units = "days")
5    ),
6    magnitudes = aquila$M,
7    idx.p = 1 : nrow(aquila)
8  )

1  # set up list of initial values
2  th.init <- list(
3    th.mu = inv.link.f$mu(0.5),
4    th.K = inv.link.f$K(0.1),
5    th.alpha = inv.link.f$alpha(1),
6    th.c = inv.link.f$c_(0.1),
7    th.p = inv.link.f$p(1.1)
8  )

```

```

1  # set starting and time of the time interval used for model fitting.
2  # In this case, we use the interval covered by the data.
3  T1 <- 0
4  T2 <- max(aquila.bru$ts) + 0.2 # Use max(..., na.rm = TRUE) if there may
5  # still be NAs here

1  # set up list of bru options
2  bru.opt.list <- list(
3    bru_verbose = 3, # type of visual output
4    bru_max_iter = 70, # maximum number of iterations
5    # bru_method = list(max_step = 0.5),
6    bru_initial = th.init # parameters' initial values
7  )

1  ETAS <- function(data = aquila.bru, m0 = M0, t1 = T1, t2 = T2,
2                    ncore = num.cores, Link.f = link.f,
3                    Bru.opt.list = bru.opt.list, n.samp = 1000,
4                    max.batch = 1000, mag = 4.5, n.breaks = 100,
5                    t.end.tri.post = 5, t.end.tri.prior = 10,
6                    t.end.omori.post = 5, t.end.omori.prior = 5){
7
8    # maximum likelihood estimator for beta
9    beta.p <- 1 / (mean(data$magnitudes) - m0)
10
11   # fit the model
12   model.fit <- Temporal.ETAS(
13     total.data = data,
14     M0 = m0,
15     T1 = t1,
16     T2 = t2,
17     link.functions = Link.f,
18     coef.t. = 1,
19     delta.t. = 0.1,
20     N.max. = 5,
21     bru.opt = Bru.opt.list
22   )
23
24   # create input list to explore model output
25   input_list <- list(
26     model.fit = model.fit,
27     link.functions = Link.f

```

```

28 )
29
30 # get marginal posterior information
31 post.list <- get_posterior_param(input.list = input_list)
32
33 # plot marginal posteriors
34 postplot <- post.list$post.plot
35
36 # posterior sampling
37 post.samp <- post_sampling(
38   input.list = input_list,
39   n.samp = n.samp,
40   max.batch = max.batch,
41   ncore = num.cores
42 )
43
44 # taking the averages of the posterior parameter estimates
45 post.par <- apply(post.samp, 2, mean)
46
47 # pair plot
48 pair.plot <- post_pairs_plot(
49   post.samp = post.samp,
50   input.list = NULL,
51   n.samp = NULL,
52   max.batch = max.batch
53 )
54 pairplot <- pair.plot$pair.plot
55
56 # set additional elements of the list
57 input_list$T12 <- c(t1, t2)
58 input_list$M0 <- m0
59 input_list$catalog.bru <- data
60
61 # posterior number of events
62 N.post <- get_posterior_N(input.list = input_list)
63 Npostplot <- N.post$post.plot
64 Npostmean <- N.post$post.df[which.max(N.post$post.df$mean), 1]
65
66 # number of large events
67 large_events <- data[data$magnitudes >= mag,]
68 Nlarge <- nrow(large_events)

```



```

69
70   # mean absolute distance of the differences in magnitudes
71   diff_mag <- diff(data$magnitudes)
72   abs_dist_mag <- mean(abs(diff_mag))
73
74   # mean absolute distance of the inter-arrival time
75   interarrival <- diff(data$ts)
76   abs_dist_int <- mean(abs(interarrival))
77
78   # check if overdispersion occurs
79   m_int_time <- mean(interarrival)
80   v_int_time <- var(interarrival)
81   overdisp <- m_int_time ^ 2 < v_int_time
82
83   # triggering function plots
84   # posterior
85   triplotpost <- triggering_fun_plot(
86     input.list = input_list,
87     post.samp = post.samp,
88     n.samp = NULL, magnitude = mag,
89     t.end = t.end.tri.post, n.breaks = n.breaks
90   )
91
92   # prior
93   triplotprior <- triggering_fun_plot_prior(input.list = input_list,
94     magnitude = mag, n.samp = n.samp,
95     t.end = t.end.tri.prior)
96
97   # omori plots
98   # posterior
99   omoripost <- omori_plot_posterior(input.list = input_list,
100     post.samp = post.samp,
101     n.samp = NULL, t.end = t.end.omori.post)
102
103   # prior
104   omoriprior <- omori_plot_prior(input.list = input_list,
105     n.samp = n.samp,
106     t.end = t.end.omori.prior)
107
108   # returns the whole environment
109   envir <- as.list(environment())
110   return(tibble::lst(envir))

```

```

111 }
112 etas <- ETAS()

```

Effect of mis-specifying parameters

```

1  ## set copula transformations list
2  # link.f1 <- list(
3  #   mu = \ (x) gamma_t(x, 0.3, 0.6),
4  #   K = \ (x) unif_t(x, 0, 10),
5  #   alpha = \ (x) unif_t(x, 0, 10),
6  #   c_ = \ (x) unif_t(x, 0, 10),
7  #   p = \ (x) unif_t(x, 1, 10)
8  # )

```

Synthetic catalogues generation

```

1  mult.synth.ETAS <- function(t1 = NULL, t2 = NULL, n.cat = 1000,
2                             ht = NULL){
3
4     # inherits the environment from function `ETAS`
5     envir <- etas$envir
6
7     # updates environments if specified by users
8     envir$t1 <- ifelse(!is.null(t1), t1, envir$t1)
9     envir$t2 <- ifelse(!is.null(t2), t2, envir$t2)
10
11    # Function to generate a synthetic catalogue
12    synth.gen <- function(i){
13        iteration <- i
14        synth <- generate_temporal_ETAS_synthetic(
15            theta = envir$post.par %>% as.list,
16            beta.p = envir$beta.p,
17            M0 = envir$m0, T1 = envir$t1,
18            T2 = envir$t2, Ht = ht, ncore = num.cores)
19        return(synth)
20    }
21
22    # generates catalogues as list of lists
23    multi.synth.cat.list <- lapply(seq_len(n.cat), \ (x)
24        synth.gen(x))
25
26    # stores catalogues as list of data.frames

```

```

27 multi.synth.cat.list.df <- lapply(multi.synth.cat.list,
28                                   \ (x) do.call(rbind, x))
29
30 # counts the number of events in each catalogue
31 Nevents <- lapply(seq_len(n.cat), \ (i) nrow(
32   multi.synth.cat.list.df[[i]])) %>% unlist
33
34 # counts the number of large events in each catalogue
35 mag <- etas$envir$mag
36 Nlarge <- lapply(seq_len(n.cat), \ (i) sum(
37   multi.synth.cat.list.df[[i]]$magnitudes >= mag)) %>% unlist
38
39 # extracts the highest magnitude in each catalogue
40 MaxMag <- lapply(seq_len(n.cat), \ (i) max(
41   multi.synth.cat.list.df[[i]]$magnitudes)) %>% unlist
42
43 # sets catalogue identifier
44 multi.synth.cat.list.df <- lapply(seq_len(n.cat),
45                                   \ (x) cbind(
46   multi.synth.cat.list.df[[x]],
47   cat.idx = x,
48   num_events = Nevents[x],
49   num_large = Nlarge[x],
50   max_mag = MaxMag[x]))
51
52 # merges catalogues in unique data.frame
53 multi.synth.cat.df <- do.call(rbind, multi.synth.cat.list.df)
54
55 # returns the whole environment
56 environ <- as.list(environment())
57 return(tibble::lst(environ))
58 }
59
60 mult.synth <- mult.synth.ETAS(ht = NULL)

1 # synth.imp <- function(data = etas$envir$data, impose_type = 1, n.cat = 5){
2 #
3 #   breaks <- data$magnitudes %>%
4 #     quantile(probs = seq(0, 1, length.out = 6))
5 #
6 #   classes <- data$magnitudes %>%

```

```

7 #         cut(breaks = breaks)
8 #
9 #     samps = switch(impose_type,
10 #
11 #         seq_len(5) %>%
12 #         lapply(\(i) which(classes == levels(classes)[i]) %>%
13 #             sample(2 * i)) %>% unlist,
14 #
15 #         c(5) %>%
16 #         lapply(\(i) which(classes == levels(classes)[i %% 2 + 1]) %>%
17 #             sample(i * (i + 1))) %>% unlist
18 #     )
19 #
20 #     ht <- data[samps,]
21 #
22 #     mult.synth.imp <- mult.synth.ETAS(ht = ht, n.cat = n.cat)
23 #
24 #     return(mult.synth.imp)
25 # }

```

## Fitting Models on the Synthetic Catalogues

```

1 hier_clu_samp <- function(syn = mult.synth){
2
3     # This function aims at performing hierarchical clustering
4     # for the synthetic catalogues specified in `syn`, as well as
5     # selecting 1 sample in each cluster.
6
7     # Extract the number of events, the number of large events, as well as
8     # the highest magnitude in each catalogue.
9     # Store them into a data frame.
10    multi.synth.cat.list.df <- syn$enviro$multi.synth.cat.list.df
11    syn.cat.info <- multi.synth.cat.list.df %>% length %>% seq_len %>%
12        lapply(\(i) multi.synth.cat.list.df[[i]][1, 5 : 7])
13    synth.df <- do.call(rbind, syn.cat.info)
14
15    # Calculate the agglomerative coefficients for different
16    # linkage methods, and select the method with the highest
17    # agglomerative coefficient.
18    link_m <- c('single', 'complete', 'average', 'ward')
19    agg_coef <- link_m %>%

```

```

20     supply(\(i) (synth.df %>% agnes(method = i))$ac)
21 method <- agg_coef %>% which.max %>% names
22 method <- ifelse(!(method == 'ward'), method, 'ward.D2')
23
24 # Determine the optimal number of clusters.
25 opt_num <- clusGap(synth.df, FUN = hcut,
26                   K.max = 30, B = 20)$Tab[, 3] %>% which.max
27
28 # Add a new column specifying the numbers of clusters.
29 synth.final <- synth.df %>%
30   cbind(
31     cluster = (
32       synth.df %>%
33         dist(method = 'euclidean') %>%
34         hclust(method = method) %>%
35         cutree(k = opt_num)
36     )
37   )
38
39 # Select 1 sample from each cluster, and return the sample id.
40 categories <- as.factor(synth.final$cluster)
41 samp.id <- (categories %>% levels %>% length %>% seq_len %>%
42   lapply(
43     \(i) (categories == levels(categories)[i]) %>%
44       which %>% sample(1))) %>% unlist
45
46 return(samp.id)
47 }

1 synth.model <- function(syn = mult.synth,
2                           slice = seq(1, 7, by = 1)){
3
4   Nevents <- syn$enviroN$Nevents
5   Nlar <- syn$enviroN$Nlarge
6   MaxMag <- syn$enviroN$MaxMag %>% round(2)
7   samp.id <- hier_clu_samp(syn)
8
9   # modelling
10  post <- rep(list(NULL), samp.id %>% length)
11  post.par <- matrix(rep(0, (samp.id %>% length) * 5),
12                    ncol = 5)

```

```

13
14   for(i in samp.id %>% length %>% seq_len){
15     multi.synth.etas <- ETAS(data =
16       syn$enviro$multi.synth.cat.list.df[[samp.id[i]]],
17       t1 = syn$enviro$envir$t1,
18       t2 = syn$enviro$envir$t2)
19     post[[i]] <- multi.synth.etas$envir$post.list
20     post.par[i,] <- multi.synth.etas$envir$post.par
21
22     post[[i]]$post.df$Catalogues <-
23     paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
24       'Events, with', Nlar[samp.id[i]], 'Large Events, ',
25       'and the Highest Magnitude is', MaxMag[samp.id[i]])
26   }
27
28   df.true.param <- data.frame(x = etas$envir$post.par,
29     param = names(etas$envir$post.par %>% as.list))
30
31   # bind marginal posterior data.frames
32   bind.post.df <- do.call(rbind,
33     lapply(samp.id %>% length %>% seq_len,
34       \(i) post[[i]]$post.df))
35
36   # plot them
37   post.par.plot <- ggplot(bind.post.df,
38     aes(x = x, y = y, colour = Catalogues)) +
39     geom_line() +
40     facet_wrap(facets = ~ param, scales = "free") +
41     xlab("param") +
42     ylab("pdf") +
43     geom_vline(
44       data = df.true.param,
45       mapping = aes(xintercept = x), linetype = 2
46     )
47
48   # returns the whole environment
49   environ <- as.list(environment())
50   return(tibble::lst(environ))
51 }
52 mult.synth.fit <- lapply(seq_len(3), \(i) synth.model(syn = mult.synth,
53   slice = seq(7 * (i - 1) + 1, 7 * i, by = 1)))
54 mult.synth.fit[[4]] <- synth.model(syn = mult.synth,

```

```

55         slice = seq(22, sampid %>% length, by = 1))

1  synth.fit <- function(samp.each.class = 5,
2
3         syn = mult.synth,
4         charac_id = 1,
5         bin_id = 1,
6         impose_type = 1){
7
8  if(is.null(syn)){
9      syn1 <- synth.imp(impose_type = impose_type)
10     syn <- syn1
11
12     # selecting catalogues
13     Nevents <- syn$enviroN$Nevents
14     Nlar <- syn$enviroN$Nlarge
15     MaxMag <- syn$enviroN$MaxMag %>% round(2)
16
17     # we need to bind the synthetics with the observed catalogue
18     # for plotting
19     cat.df.for.plotting <- rbind(
20         syn$enviroN$multi.synth.cat.df,
21         cbind(syn$enviroN$envir$data[, c("ts", "magnitudes")],
22             gen = NA, cat.idx = "observed",
23             num_events = nrow(etas$envir$data),
24             num_large = etas$envir$Nlarge,
25             max_mag = max(etas$envir$data$magnitudes) %>% round(2)
26         )
27     )
28
29 }else{
30     syn1 <- NULL
31     # selecting catalogues
32     Nevents <- syn$enviroN$Nevents
33     Nlar <- syn$enviroN$Nlarge
34     MaxMag <- syn$enviroN$MaxMag %>% round(2)
35
36     characteristic <- switch(charac_id, Nevents, Nlar)
37     breaks <- switch(charac_id,
38         quantile(characteristic,
39             probs =

```

```

40         seq(0, 17 / 20, length.out = 4)[
41             c(bin_id, bin_id + 1)]),
42         quantile(characteristic + .1,
43             probs =
44             seq(0, 17 / 20, length.out = 4)[
45                 c(bin_id, bin_id + 1)])
46     )
47
48
49     classes <- cut(characteristic, breaks = breaks)
50     samp.id <- rep(0, samp.each.class * (classes %>% levels %>% length))
51     for(i in classes %>% levels %>% length %>% seq_len){
52         samp.id[(samp.each.class * (i - 1) + 1) :
53             (samp.each.class * i)] <-
54             sample(which(classes == levels(classes)[i]),
55                 samp.each.class)
56     }
57
58     # we need to bind the synthetics with the observed catalogue
59     # for plotting
60     cat.df.for.plotting <- rbind(
61         syn$enviro$multi.synth.cat.df[
62             which(
63                 syn$enviro$multi.synth.cat.df$cat.idx %in% samp.id),
64             ],
65         cbind(syn$enviro$envir$data[, c("ts", "magnitudes")],
66             gen = NA, cat.idx = "observed",
67             num_events = nrow(etas$envir$data),
68             num_large = etas$envir$Nlarge,
69             max_mag = max(etas$envir$data$magnitudes) %>% round(2)
70         )
71     )
72 }
73
74 # plot them
75 multi.synth.cat.plot <- ggplot(cat.df.for.plotting,
76     aes(ts, magnitudes)) +
77     geom_point(size = 0.5) +
78     geom_point(
79         data = syn$enviro$ht,
80         mapping = aes(ts, magnitudes), colour = "black"

```



```

81 ) +
82 facet_wrap(facets =
83             vars(cat.idx, num_events, num_large, max_mag),
84             labeller = 'label_both', ncol = 2)
85
86 if(!(is.null(syn1))){
87   # modelling
88   input <- rep(list(NULL), syn$enviro$N.cat)
89   post <- rep(list(NULL), syn$enviro$N.cat)
90   post.par <- matrix(rep(0, (syn$enviro$N.cat) * 5),
91                      ncol = 5)
92   Npost <- rep(list(NULL), syn$enviro$N.cat)
93   Npostmean <- rep(0, syn$enviro$N.cat)
94   abs_dist_int <- rep(0, syn$enviro$N.cat)
95   abs_dist_mag <- rep(0, syn$enviro$N.cat)
96   overdisp <- rep(0, syn$enviro$N.cat)
97
98   for(i in syn$enviro$N.cat %>% seq_len){
99     multi.synth.etas <- ETAS(data =
100                             syn$enviro$multi.synth.cat.df,
101                             t1 = syn$enviro$envir$t1,
102                             t2 = syn$enviro$envir$t2)
103     post[[i]] <- multi.synth.etas$envir$post.list
104     post.par[i,] <- multi.synth.etas$envir$post.par
105     Npost[[i]] <- multi.synth.etas$envir$N.post
106     Npostmean[i] <- multi.synth.etas$envir$Npostmean
107     abs_dist_int[i] <- multi.synth.etas$envir$abs_dist_int
108     abs_dist_mag[i] <- multi.synth.etas$envir$abs_dist_mag
109     overdisp[i] <- multi.synth.etas$envir$overdisp
110
111     post[[i]]$post.df$Catalogues <-
112     paste('Random Catalogue', i, ': ', Nevents[i],
113           'Events, with', Nlar[i], 'Large Events, ',
114           'and the Highest Magnitude is', MaxMag[i])
115
116     Npost[[i]]$post.df$Catalogues <-
117     paste('Random Catalogue', i, ': ', Nevents[i],
118           'Events, with', Nlar[i], 'Large Events, ',
119           'and the Highest Magnitude is',
120           MaxMag[i])
121   }

```

```

122
123 df.true.param <- data.frame(x = etas$envir$post.par,
124                             param = names(etas$envir$post.par %>% as.list))
125
126 # bind marginal posterior data.frames
127 bind.post.df <- do.call(rbind,
128                         lapply(syn$environ$n.cat %>% seq_len,
129                               \ (i) post[[i]]$post.df))
130
131 # plot them
132 post.par.plot <- ggplot(bind.post.df,
133                          aes(x = x, y = y, colour = Catalogues)) +
134   geom_line() +
135   facet_wrap(facets = ~ param, scales = "free") +
136   xlab("param") +
137   ylab("pdf") +
138   geom_vline(
139     data = df.true.param,
140     mapping = aes(xintercept = x), linetype = 2
141   )
142
143 ##
144 df.true.N <- data.frame(N = etas$envir$Npostmean, param = 'N')
145
146 # bind marginal posterior data.frames
147 bind.post.N.df <- do.call(rbind,
148                           lapply(syn$environ$n.cat %>% seq_len,
149                                 \ (i) Npost[[i]]$post.df))
150
151 # plot them
152 post.N.plot <- ggplot(bind.post.N.df,
153                       aes(x = N, y = mean, colour = Catalogues)) +
154   geom_line() +
155   xlab("N") +
156   ylab("pdf") +
157   geom_vline(
158     data = df.true.N,
159     mapping = aes(xintercept = N), linetype = 2
160   )
161
162 }else{
163   # modelling

```

```

164 input <- rep(list(NULL), samp.id %>% length)
165 post <- rep(list(NULL), samp.id %>% length)
166 post.par <- matrix(rep(0, (samp.id %>% length) * 5),
167                   ncol = 5)
168 Npost <- rep(list(NULL), samp.id %>% length)
169 Npostmean <- rep(0, samp.id %>% length)
170 abs_dist_int <- rep(0, samp.id %>% length)
171 abs_dist_mag <- rep(0, samp.id %>% length)
172 overdisp <- rep(0, samp.id %>% length)
173
174 for(i in samp.id %>% length %>% seq_len){
175   multi.synth.etas <- ETAS(data =
176     syn$enviro$multi.synth.cat.list.df[[samp.id[i]]],
177     t1 = syn$enviro$envir$t1,
178     t2 = syn$enviro$envir$t2)
179   post[[i]] <- multi.synth.etas$envir$post.list
180   post.par[i,] <- multi.synth.etas$envir$post.par
181   Npost[[i]] <- multi.synth.etas$envir$N.post
182   Npostmean[i] <- multi.synth.etas$envir$Npostmean
183   abs_dist_int[i] <- multi.synth.etas$envir$abs_dist_int
184   abs_dist_mag[i] <- multi.synth.etas$envir$abs_dist_mag
185   overdisp[i] <- multi.synth.etas$envir$overdisp
186
187   post[[i]]$post.df$Catalogues <-
188     paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
189           'Events, with', Nlar[samp.id[i]], 'Large Events, ',
190           'and the Highest Magnitude is', MaxMag[samp.id[i]])
191
192   Npost[[i]]$post.df$Catalogues <-
193     paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
194           'Events, with', Nlar[samp.id[i]], 'Large Events, ',
195           'and the Highest Magnitude is',
196           MaxMag[samp.id[i]])
197 }
198
199 df.true.param <- data.frame(x = etas$envir$post.par,
200                             param = names(etas$envir$post.par %>% as.list))
201
202 # bind marginal posterior data.frames
203 bind.post.df <- do.call(rbind,
204                         lapply(samp.id %>% length %>% seq_len,

```

```

205         \ (i) post[[i]]$post.df))
206
207     # plot them
208     post.par.plot <- ggplot(bind.post.df,
209                             aes(x = x, y = y, colour = Catalogues)) +
210         geom_line() +
211         facet_wrap(facets = ~ param, scales = "free") +
212         xlab("param") +
213         ylab("pdf") +
214         geom_vline(
215             data = df.true.param,
216             mapping = aes(xintercept = x), linetype = 2
217         )
218
219     ##
220     df.true.N <- data.frame(N = etas$envir$Npostmean, param = 'N')
221
222     # bind marginal posterior data.frames
223     bind.post.N.df <- do.call(rbind,
224                               lapply(samp.id %>% length %>% seq_len,
225                                     \ (i) Npost[[i]]$post.df))
226
227     # plot them
228     post.N.plot <- ggplot(bind.post.N.df,
229                             aes(x = N, y = mean, colour = Catalogues)) +
230         geom_line() +
231         xlab("N") +
232         ylab("pdf") +
233         geom_vline(
234             data = df.true.N,
235             mapping = aes(xintercept = N), linetype = 2
236         )
237 }
238
239 # returns the whole environment
240 environ <- as.list(environment())
241 return(tibble::lst(environ))
242 }
243 # mult.synth.fit <- lapply(seq_len(6), \ (i) synth.fit(syn = mult.synth,
244 #                                                       charac_id = (i - 1) %% 3 + 1,
245 #                                                       bin_id = (i - 1) %% 3 + 1))

```

```
246 # test <- synth.fit(syn = NULL, impose_type = 1)
```

## Analysis on the Behaviours of the Time-between-Events

```
1 ECDF.interarrival <- function(j){
2
3   samp.id <- mult.synth.fit[[j]]$enviro$ Samp.id
4   Nevents <- mult.synth.fit[[j]]$enviro$Nevents
5   Nlar <- mult.synth.fit[[j]]$enviro$Nlar
6   MaxMag <- mult.synth.fit[[j]]$enviro$MaxMag
7
8   data.list <- lapply(samp.id %>% length %>% seq_len, \(i) data.frame(
9     Time_between_events =
10      mult.synth$enviro$multi.synth.cat.list.df[[samp.id[i]]]$ts %>%
11      sort %>% diff,
12     Catalogues = paste('Random Catalogue', i, ':', Nevents[samp.id[i]],
13      'Events, with', Nlar[samp.id[i]], 'Large Events, ',
14      'and the Highest Magnitude is',
15      MaxMag[samp.id[i]])
16   )
17 )
18
19 data.list[[length(samp.id) + 1]] <- data.frame(
20   Time_between_events = etas$envir$interarrival,
21   Catalogues = paste('Observed', ':', nrow(etas$envir$data),
22     'Events, with', etas$envir$Nlarge, 'Large Events, ',
23     'and the Highest Magnitude is',
24     max(etas$envir$data$magnitudes) %>% round(2)))
25
26 df <- do.call(rbind, data.list)
27
28 ECDF.plot <- ggplot(df, aes(x = Time_between_events,
29   colour = Catalogues)) +
30   stat_ecdf() +
31   xlab('Time between Events') +
32   ylab('Empirical Cumulative Probability')
33
34   return(ECDF.plot)
35 }
36
37 ecdf_interarrival <- lapply(seq_len(5), \(j) ECDF.interarrival(j))
```

## Forecasting

```
1 ETAS.forecast <- function(){
2
3   # inherits the environment from function `ETAS`
4   envir <- etas$envir
5
6   # express 1 minute in days
7   min.in.days <- 1 / (24 * 60)
8   # find time of the event with the greatest magnitude
9   t.max.mag <- envir$data$ts[which.max(envir$data$magnitudes)]
10  # set starting time of the forecasting period
11  T1.fore <- t.max.mag + min.in.days
12  # set forecast length
13  fore.length <- 1
14  # set end time of the forecasting period
15  T2.fore <- T1.fore + fore.length
16  # set known data
17  Ht.fore <- envir$data[envir$data$ts < T1.fore, ]
18
19  # produce forecast
20  daily.fore <- Temporal.ETAS.forecast(
21    post.samp = envir$post.samp, # ETAS parameters posterior samples
22    n.cat = nrow(envir$post.samp), # number of synthetic catalogues
23    beta.p = envir$beta.p, # magnitude distribution parameter
24    M0 = envir$m0, # cutoff magnitude
25    T1 = T1.fore, # forecast starting time
26    T2 = T2.fore, # forecast end time
27    Ht = Ht.fore, # known events
28    ncore = num.cores # number of cores
29  )
30
31  # find number of events per catalogue
32  N.fore <- vapply(
33    seq_len(daily.fore$n.cat),
34    \(x) sum(daily.fore$fore.df$cat.idx == x), 0
35  )
36  # find number of observed events in the forecasting period
37  N.obs <- sum(envir$data$ts >= T1.fore & envir$data$ts <= T2.fore)
38  # plot the distribution
39  histfore <- ggplot() +
40    geom_histogram(aes(x = N.fore, y = after_stat(density)),
```

```

41         binwidth = 1) +
42     geom_vline(xintercept = N.obs) +
43     xlim(100, 500)
44
45     return(tibble::lst(N.fore, N.obs, histfore))
46 }
47 # fore <- ETAS.forecast()

```

```

1 # save.image(file = 'Dissertation.RData')

```