Bayesian hierarchical models for NHPP using `rstan`

Miao Cai miao.cai@slu.edu

2019-07-09

Contents

1 Model setting

Let $T_{d,s,i}$ denote the time to the d-th driver's s-th shift's i-th critical event. The total number critical events of d-th driver's s-th shift is $n_{d,s}$. The ranges of these notations are:

- $i = 1, 2, \cdots, n_{d,S_d}$,
- $s = 1, 2, \cdots, S_d$,
- $d = 1, 2, \cdots, D$.

We assume the times of critical events within the d-th driver's s-th shift were generated from a non-homogeneous Poisson process (NHPP) with a power law process (PLP), with a fix rate parameter β and varying scale parameters $\theta_{d,s}$ across drivers. The data generating process is then:

$$
\begin{aligned}
T_{d,s,1}, T_{d,s,2}, \cdots, T_{d,s,n_{d,s}} &\sim \mathrm{PLP}(\beta, \theta_{d,s}) \\
\beta &\sim \mathrm{Gamma}(1,1) \\
\log \theta_{d,s} &= \gamma_{0d} + \gamma_1 x_{d,s,1} + \gamma_2 x_{d,s,2} + \cdots + \gamma_k x_{d,s,k} \\
\gamma_{01}, \gamma_{02}, \cdots, \gamma_{0D} &\sim \text{i.i.d. } N(\mu_0, \sigma_0^2) \\
\gamma_1, \gamma_2, \cdots, \gamma_k &\sim \text{i.i.d. } N(0, 10^2) \\
\mu_0 &\sim N(0, 10^2) \\
\sigma_0 &\sim \mathrm{Gamma}(1,1)
\end{aligned}
$$

2 Simulating data

1. Random intercepts $\gamma_{01}, \gamma_{02}, \cdots, \gamma_{0D}$. The standard deviation of μ_0 was intentionally set to small number 2 to make $\theta_{d,s}$ fall into a reasonably small range. If I otherwise set it as 10, $\theta_{d,s}$ may be more than 10^5 due to the exponentiation, which may not be realistic in real-life data.

$$\mu_0 = 1, \quad \sigma_0 = 1$$

$$\sigma_0 \sim \text{Gamma}(1, 1)$$

$$\gamma_{01}, \gamma_{02}, \cdots, \gamma_{0D} \sim \text{i.i.d. } N(\mu_0, \sigma_0^2)$$

2. fixed parameters: 3 fixed parameters $\gamma_1, \gamma_2, \gamma_3$.

$$\gamma_1, \gamma_2, \gamma_3 \sim \text{i.i.d. } N(0, 10^2)$$

3. The number of observations in the d-th driver: N_d.

$$N_d \sim \text{Poisson}(100)$$

4. Data: 3 predictor variables $x_{d,s,1}, x_{d,s,2}, x_{d,s,3}$.

$$x_{d,s,1} \sim \text{N}(0, 10)$$

$$x_{d,s,2} \sim \text{Gamma}(10, 2)$$

$$x_{d,s,3} \sim \text{Poisson}(3.5)$$

5. Scale parameters of a NHPP (random effects): $\theta_{d,s}$.

$$\theta_{d,s} = \text{EXP}(\gamma_{0d} + \gamma_1 x_{d,s,1} + \gamma_2 x_{d,s,2} + \gamma_k x_{d,s,3})$$

6. Shape parameter of a NHPP (fixed effect): $\beta \sim \text{Gamma}(1, 1)$. Set

$$\beta = 1.5$$

7. Simulate a NHPP based on β and $\theta_{d,s}$.

$$T_{d,s,1}, T_{d,s,2}, \cdots, T_{d,s,n_{d,s}} \sim \text{PLP}(\beta, \theta_{d,s})$$

```r
pacman::p_load(rstan, tidyverse, data.table)
source("functions/NHPP_functions.R")

#set.seed(123)
D = 10 # the number of drivers
K = 3 # the number of predictor variables

# 1. Random-effect intercepts
# hyperparameters
mu0 = 1
sigma0 = 1
r_0D = rnorm(D, mean = mu0, sd = sigma0)

# 2. Fixed-effects parameters
R_K = rnorm(K, mean = 0, sd = 1)

# 3. The number of observations in the $d$-th driver: $N_{d}$
N_K = rpois(D, 10)

# 4. Generate data: x_1, x_2, .. x_K
sim1 = function(n = 10){
  x1 = rnorm(n, 0, 5)
  x2 = rgamma(n, 5, 1)
  x3 = rpois(n, 3.5)
  return(data.frame(x1, x2, x3))
}
simXD = function(ndrivers = D){
  XD = rep(list(data.frame()), ndrivers)
  for (i in 1:D) {
    XD[[i]] = sim1(N_K[i])
  }
  return(data.table::rbindlist(XD, idcol = "driver"))
}
X = simXD()

# 5. Scale parameters of a NHPP
# 5a. parameter matrix: P
N_D = X[,.N,driver][["N"]]# N by driver
N_all = sum(N_D) # total N
P = cbind(r0 = rep(r_0D, N_D),
          t(replicate(N_all, R_K)))
M_logtheta = P*X

theta = exp(rowSums(M_logtheta))
beta = 1.5
```

3 Estimation

```
plptauML = '
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
}
data {
  int<lower=0> D; //driver index
  int<lower=0> N; //total # of obs
  int<lower=0> K; //total # of shifts
  vector<lower=0>[K] tau;//truncated time
  vector<lower=0>[N] event_time; //failure time
  int s[K]; //group sizes
}
parameters{
  real<lower=0> beta;
  vector[K] r0; // random intercept
  vector[3] r; // fixed parameters
  real mu0; // hyperparameter
  real<lower=0> sigma0;// hyperparameter
}
transformed parameters{
  vector<lower=0>[K] theta;

  for (k0 in 1:K)
    theta[k0] = r0[D[k0]] + x1[i]*r[1] + x2[i]*r[2] + x3[i]*r[3];
}
model{
  int position;
  position = 1;
  for (k in 1:K){
    segment(event_time, position, s[k]) ~ nhpp(beta, theta[k], tau[k]);
    position = position + s[k];
  }
//PRIORS
  beta ~ gamma(1, 1);
  r0 ~ normal(mu0, sigma0);
  r  ~ normal(0, 10);
  mu0 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
  theta ~ gamma(1, 0.01);
}
'
```

In Stan code:

- Need a data matrix X,
- Need matrix multiplication,

In data:

- Need a driver index $d = 1, 2, \cdots, K$ for each shift k
- Need a data matrix X

```r
cat(readLines("stan/nhpp_plp_tau_ML.stan"))
```

```
## functions{   real nhpp_log(vector t, real beta, real theta, real tau){     vector[num_elements(t)] lo
```

4 Inline Rmarkdown Stan chunk

```
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
}
data {
  int<lower=0> N; //total # of obs
  int<lower=0> K; //total # of shifts
  int<lower=0> D[K];//driver index
  vector<lower=0>[K] tau;//truncated time
  vector<lower=0>[N] event_time; //failure time
  vector[K] x1;
  vector[K] x2;
  vector[K] x3;
  int s[K]; //group sizes
}
parameters{
  real<lower=0> beta;
  vector[K] r0; // random intercept
  vector[3] r; // fixed parameters
  real mu0; // hyperparameter
  real<lower=0> sigma0;// hyperparameter
}
transformed parameters{
  vector<lower=0>[K] theta;
  for (k0 in 1:K){
    theta[k0] = r0[ D[k0] ] + x1[k0]*r[1] + x2[k0]*r[2] + x3[k0]*r[3];
  }
}
model{
  int position;
  position = 1;
  for (k in 1:K){
    segment(event_time, position, s[k]) ~ nhpp(beta, theta[k], tau[k]);
    position = position + s[k];
  }
//PRIORS
  beta ~ gamma(1, 1);
  r0 ~ normal(mu0, sigma0);
  r  ~ normal(0, 10);
  mu0 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
  theta ~ gamma(1, 0.01);
}
```

```r
fit <- sampling(ex1)

print(fit)
```

```r
N<-100 #sample size

J<-10 #number of plant species

id<-rep(1:J,each=10) #index of plant species

K<-3 #number of regression coefficients

#population-level regression coefficient

gamma<-c(2,-1,3)

#standard deviation of the group-level coefficient

tau<-c(0.3,2,1)

#standard deviation of individual observations

sigma<-1

#group-level regression coefficients

beta<-mapply(function(g,t) rnorm(J,g,t),g=gamma,t=tau)

#the model matrix

X<-model.matrix(~x+y,data=data.frame(x=runif(N,-2,2),y=runif(N,-2,2)))

y<-vector(length = N)

for(n in 1:N){

  #simulate response data

  y[n]<-rnorm(1,X[n,]%*%beta[id[n],],sigma)

}
```