Hierarchical Jump-point PLP (JPLP) simulation

Miao Cai miao.cai@slu.edu

2020-05-14

Contents

1 Bayesian Hierarchical Jump Power Law Process (JPLP)

1.1 Model setting

The Bayesian hierarchical JPLP model is parameterized as

$$
\begin{aligned}
t_{d,s,1}, t_{d,s,2}, \cdots, t_{d,s,n_{d,s}} &\sim \text{JPLP}(\beta, \theta_{d,s}, \kappa)\\
\beta &\sim \text{Gamma}(1,1)\\
\log \theta_{d,s} &= \gamma_{0d} + \gamma_1 x_{d,s,1} + \gamma_2 x_{d,s,2} + \cdots + \gamma_k x_{d,s,k}\\
\kappa &\sim \text{Uniform}(0,1)\\
\gamma_{01}, \gamma_{02}, \cdots, \gamma_{0D} &\sim \text{i.i.d. } N(\mu_0, \sigma_0^2)\\
\gamma_1, \gamma_2, \cdots, \gamma_k &\sim \text{i.i.d. } N(0, 10^2)\\
\mu_0 &\sim N(0, 5^2)\\
\sigma_0 &\sim \text{Gamma}(1,1),
\end{aligned}
\tag{1}
$$

where the introduced parameter κ is the percent of intensity function recovery once the driver takes a break. By definition, $a_{d,s,0} = 0$. We assume that this κ is constant across drivers and shifts.

1.2 Intensity function of JPLP

Since the Bayesian hierarchical PLP does not account for the rests within a shift and associated potential reliability repairment. In this subsection, we proposes a Bayesian hierarchical JPLP, with the following intensity function:

$$
\begin{aligned}
\lambda_{\text{JPLP}}(t|d, s, r, \beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}) &= \begin{cases} \kappa^0 \lambda(t|\beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}) & 0 < t \leq a_{d,s,1} \\ \kappa^1 \lambda(t|\beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}) & a_{d,s,1} < t \leq a_{d,s,2} \\ \cdots & \cdots \\ \kappa^{R-1} \lambda(t|\beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}) & a_{d,s,R-1} < t \leq a_{d,s,R} \end{cases} \\
&= \kappa^{r-1} \lambda(t|d, s, r, \kappa, \beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}) \quad a_{d,s,r-1} < t \leq a_{d,s,r},
\end{aligned} \tag{2}
$$

The notations are identical with those in PLP except for the extra κ parameter.

1.3 The likelihood function of JPLP

The likelihood function for driver d on shift s is

$$
\begin{aligned}
L_{s,d}(\kappa, \beta, \gamma_{0,d}, \gamma|\text{Data}_{d,s}) = &\Big(\prod_{i=1}^{c_{d,s}} \lambda\big(t_{i,d,s}|d, s, r, k, \beta, \gamma_{0,d}, \gamma, \mathbf{X}_d, \mathbf{W}\big) \Big) \\
&\times \exp\Big(- \int_0^{a_{d,s,r}\lambda\big(u|d,s,r,k,\beta,\gamma_{0,d},\gamma,\mathbf{X}_d,\mathbf{W}\big)du} \Big)
\end{aligned} \tag{3}
$$

The overall likelihood function is

$$
L = \prod_d \prod_{s \in d} L_{s,d} \tag{4}
$$

2 Simulating parameters and data

- Parameters needed: $\kappa, \beta, \theta, \gamma_{0,d}, \gamma$
  - $\theta \leftarrow \gamma_{0,d}, \gamma, \mathbf{X}$
- Data needed: $\mathbf{X}$
  - x_1, x_2, x_3

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

sim_hier_JPLP = function(
  beta = 1.2,
  kappa = 0.8,
  D = 10, # the number of drivers
  K = 3, # the number of predictor variables
  group_size_lambda = 10, # the mean number of shifts for each driver
  mu0 = 0.2, # hyperparameter 1
  sigma0 = 0.5, # hyperparameter 2
  R_K = c(1, 0.3, 0.2) # Fixed-effects parameters
)
{
  # 1. Random-effect intercepts
  r_0D = rnorm(D, mean = mu0, sd = sigma0)
```

```r
# 3. The number of observations (shifts) in the $d$-th driver: $N_{d}$
N_K = rpois(D, group_size_lambda)
N = sum(N_K) # the total number of shifts for all D drivers
id = rep(1:D, N_K)

# 4. Generate data: x_1, x_2, .. x_K
simX = function(group_sizes = N_K)
  {
    ntot = sum(group_sizes)

    int1 = rep(1, ntot)
    x1 =  rnorm(ntot, 1, 1)
    x2 = rgamma(ntot, 1, 1)
    x3 =  rpois(ntot, 2)

    return(data.frame(int1, x1, x2, x3))
  }
X = simX(N_K)

# 5. Scale parameters of a JPLP
# 5a. parameter matrix: P
P = cbind(r0 = rep(r_0D, N_K), t(replicate(N, R_K)))
M_logtheta = P*X
theta = exp(rowSums(M_logtheta))

# Initialization of lists
t_shift_vec = list()
n_stop_vec = list()
t_stop_vec = list()
n_event_vec = list()
t_event_vec = list()

for (i in 1:N)
  {
    sim_tau = rnorm(1, 10, 1.3)
    n_stop = get_n_stop()
    sim_t_trip = round((1:n_stop)*sim_tau/(n_stop + 1) +
                         rnorm(n_stop, 0, sim_tau*0.15/n_stop), 2)
    t_events = sim_jplp(tau0 = sim_tau,
                        kappa0 = kappa,
                        t_trip0 = sim_t_trip,
                        beta0 = beta,
                        theta0 = theta[i])
    t_shift_vec[[i]] = sim_tau
    n_stop_vec[[i]] = n_stop
    t_stop_vec[[i]] = sim_t_trip
```

```r
      n_event_vec[[i]] = length(t_events)
      t_event_vec[[i]] = t_events
    }



  # shifts data
  shift_dt = data.frame(
    driver_id = rep(1:D, N_K),
    shift_id = 1:N,
    start_time = rep(0, N),
    end_time = Reduce(c, t_shift_vec),
    n_stop = Reduce(c, n_stop_vec),
    n_event = Reduce(c, n_event_vec)
  )

  # trips data set
  trip_dt = data.frame(
    driver_id = rep(shift_dt$driver_id, shift_dt$n_stop),
    shift_id = rep(1:N, unlist(n_stop_vec)),
    trip_time = Reduce(c, t_stop_vec)
  )

  # TEMPORARY vector: a temporary vector for events per driver
  n_event_driver = shift_dt %>%
    group_by(driver_id) %>%
    summarise(n_event = sum(n_event)) %>%
    pull(n_event)

  # events data set
  event_dt = data.frame(
    driver_id = rep(1:D, n_event_driver),
    shift_id = rep(1:N, Reduce(c, n_event_vec)),
    event_time = Reduce(c, t_event_vec)
  )

  return(list(event_time = event_dt,
              trip_time = trip_dt,
              shift_time = shift_dt))
}

set.seed(123)
df = sim_hier_JPLP()
```

Simulated parameters:

- κ:
- β:

- θ:
- $\gamma_{0,d}$:
- γ:

Simulated data:

```r
str(df)
```

```
## List of 3
##  $ event_time:'data.frame':  162 obs. of  3 variables:
##   ..$ driver_id : int [1:162] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ shift_id  : int [1:162] 1 1 1 1 1 2 2 2 3 3 ...
##   ..$ event_time: num [1:162] 2.71 4.57 5.7 6.27 6.66 ...
##  $ trip_time :'data.frame':  236 obs. of  3 variables:
##   ..$ driver_id: int [1:236] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ shift_id : int [1:236] 1 1 1 1 2 2 2 3 3 3 ...
##   ..$ trip_time: num [1:236] 1.66 4.71 6.44 9.16 2.94 5.29 6.79 2.42 5.47 7.5 ...
##  $ shift_time:'data.frame':  95 obs. of  6 variables:
##   ..$ driver_id : int [1:95] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ shift_id  : int [1:95] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ start_time: num [1:95] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ end_time  : num [1:95] 11.09 10.01 10.07 10.07 9.26 ...
##   ..$ n_stop    : int [1:95] 4 3 3 2 4 3 4 4 1 1 ...
##   ..$ n_event   : int [1:95] 5 3 5 0 2 7 3 0 1 8 ...
```

3 Generate data pass on to Stan

```r
# This version is a template
# Need trip time data
sim_hier_nhpp = function(group_size_lambda = 10, D = 10, K = 3, beta = 1.5)
{
  # 1. Random-effect intercepts
  # hyperparameters
  mu0 = 0.2
  sigma0 = 0.5
  r_0D = rnorm(D, mean = mu0, sd = sigma0)

  # 2. Fixed-effects parameters
  R_K = c(1, 0.3, 0.2)

  # 3. The number of shifts in the $d$-th driver: $N_{d}$
  N_K = rpois(D, group_size_lambda)
  N = sum(N_K) # the total number of obs
  id = rep(1:D, N_K)

  # 4. Generate data: x_1, x_2, .. x_K
  sim1 = function(group_sizes = N_K)
  {
```

```r
    ntot = sum(group_sizes)

    int1 = rep(1, ntot)
    x1 = rnorm(ntot, 1, 1)
    x2 = rgamma(ntot, 1, 1)
    x3 = rpois(ntot, 2)

    return(data.frame(int1, x1, x2, x3))
  }
  X = sim1(N_K)

  # 5. Scale parameters of a NHPP
  # 5a. parameter matrix: P
  P = cbind(r0 = rep(r_0D, N_K),
            t(replicate(N, R_K)))
  M_logtheta = P*X

  # returned parameter for each observed shift

  theta_vec = exp(rowSums(M_logtheta))

  df = sim_hier_plp_tau(N = N, beta = beta, theta = theta_vec)

  hier_dat = list(
    N = nrow(df$event_dat),
    K = K,
    S = nrow(df$start_end_dat),
    D = max(id),
    id = id, #driver index
    tau = df$start_end_dat$end_time,
    event_time = df$event_dat$event_time,
    group_size = df$shift_length, #the number of events in each shift
    X_predictors = X[,2:4]
  )

  true_params = list(
    mu0 = mu0, sigma0 = sigma0,
    r0 = r_0D, r1_rk = R_K,
    beta = beta,
    theta = theta_vec
  )

  return(list(hier_dat = hier_dat, true_params = true_params))
}
```