Jump-point PLP (JPLP) simulation

Miao Cai miao.cai@slu.edu

2020-04-06

Contents

1 Power law process (PLP)

1.1 PLP intensity function

Power law process (PLP): When the intensity function of a NHPP is:

$$\lambda(t) = \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1} = \beta\theta^{-\beta}t^{\beta},$$

where $\beta > 0$ and $\theta > 0$, the process is called the power law process (PLP). The mean function $\Lambda(t)$ is the integral of the intensity function:

$$\Lambda(t) = \int_0^t \lambda(t)dt = \int_0^t \frac{\beta}{\theta}\left(\frac{t}{\theta}\right)^{\beta-1} = \left(\frac{t}{\theta}\right)^{\beta}$$

1.2 PLP simulation

```r
# simulating PLP - failure truncated case
sim_plp_n = function(mean_n, beta, theta){
  N = rpois(1, mean_n)
  u = runif(N, 0, 1)
  n_logu = -log(u)
  s = cumsum(n_logu)
  Delta_t = theta*s^(1/beta)
  return(Delta_t)
}

# simulate multiple NHPPs - time truncated case
sim_mul_plp_tau = function(n_shift = 20,
                           shift_len_mean = 20, shift_len_sd = 5,
                           theta = 10, beta = 2, mean_n = 5){
  tau_vector = rnorm(n_shift, shift_len_mean, shift_len_sd)#difference1

  t_list = list()
  len_list = list()
  # end_time1 = list() # not needed for time truncated case


  for (i in 1:n_shift) {
    t_list[[i]] = sim_plp_tau(tau_vector[i], beta, theta)
    len_list[[i]] = length(t_list[[i]])
  }

  event_dat = data.frame(
    shift_id = rep(1:n_shift, unlist(len_list)),
    event_time = Reduce(c, t_list)
  )

  start_end_dat = data.frame(
    shift_id = 1:n_shift,
    start_time = rep(0, n_shift),
    end_time = tau_vector #difference2
  )

  return(list(event_dat = event_dat,
              start_end_dat = start_end_dat,
              shift_length = unlist(len_list)))
}


sim_hier_plp_tau = function(N, beta = 1.5, theta){
  t_list = list()
  len_list = list()
  tau_vector = rnorm(N, 10, 1.3)

  for (i in 1:N) {
    t_list[[i]] = sim_plp_tau(tau_vector[i], beta = beta, theta = theta[i])
    len_list[[i]] = length(t_list[[i]])
  }

  event_dat = data.frame(
    shift_id = rep(1:N, unlist(len_list)),
    event_time = Reduce(c, t_list)
  )

  start_end_dat = data.frame(
```

```r
    shift_id = 1:N,
    start_time = rep(0, N),
    end_time = tau_vector #difference2
  )

  return(list(event_dat = event_dat,
              start_end_dat = start_end_dat,
              shift_length = unlist(len_list)))
}

plot_est = function(data, var = "beta", hline_var = 1.5){
  p = data %>%
    filter(term == var) %>%
    ggplot(aes(id, est_mean)) +
    geom_point() +
    geom_line(linetype = "dashed", color = "red")+
    geom_errorbar(aes(ymax = est_mean + 1.96*est_sd,
                      ymin = est_mean - 1.96*est_sd),
                  width = 1)+
    geom_segment(aes(x = 10, xend = 100,
                     y = hline_var, yend = hline_var),
                 color = "green")+
    scale_x_continuous(breaks = c(0, 10, 25, 50, 75, 100),
                       labels = c("0", "10", "25", "50", "75", "100")) +
    labs(x = "The number of drivers (random effects)",
         y = var) +
    theme_bw()
  return(p)
}
```

```r
# plot events
plot_events = function(event_dat, start_end_dat, cross_size = 2){
  p = event_dat %>%
    ggplot(aes(x = event_time, y = shift_id)) +
    geom_point(alpha = 0.8, shape = 4, color = 'red', size = cross_size) +
    scale_y_continuous("shift ID",
                       labels = as.character(start_end_dat$shift_id),
                       breaks = start_end_dat$shift_id)+
    xlab('Time to event (minutes)') +
    geom_segment(data = start_end_dat,
                 aes(x = start_time, xend = end_time,
                     y = shift_id, yend = shift_id),
                 lineend = 'butt',
                 arrow = arrow(length = unit(0.2, "cm"))) +
    theme_classic()
  return(p)
}
```

2 Jump Power Law Process (JPLP)

2.1 JPLP intensity function

A Bayesian hierarchical JPLP has the following intensity function:

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

where the introduced parameter κ is the percent of intensity function recovery once the driver takes a break. We assume that this κ is constant across drivers and shifts.