Conquer zero-event issue in hierarchical NHPP

Miao Cai miao.cai@slu.edu

2019-07-24

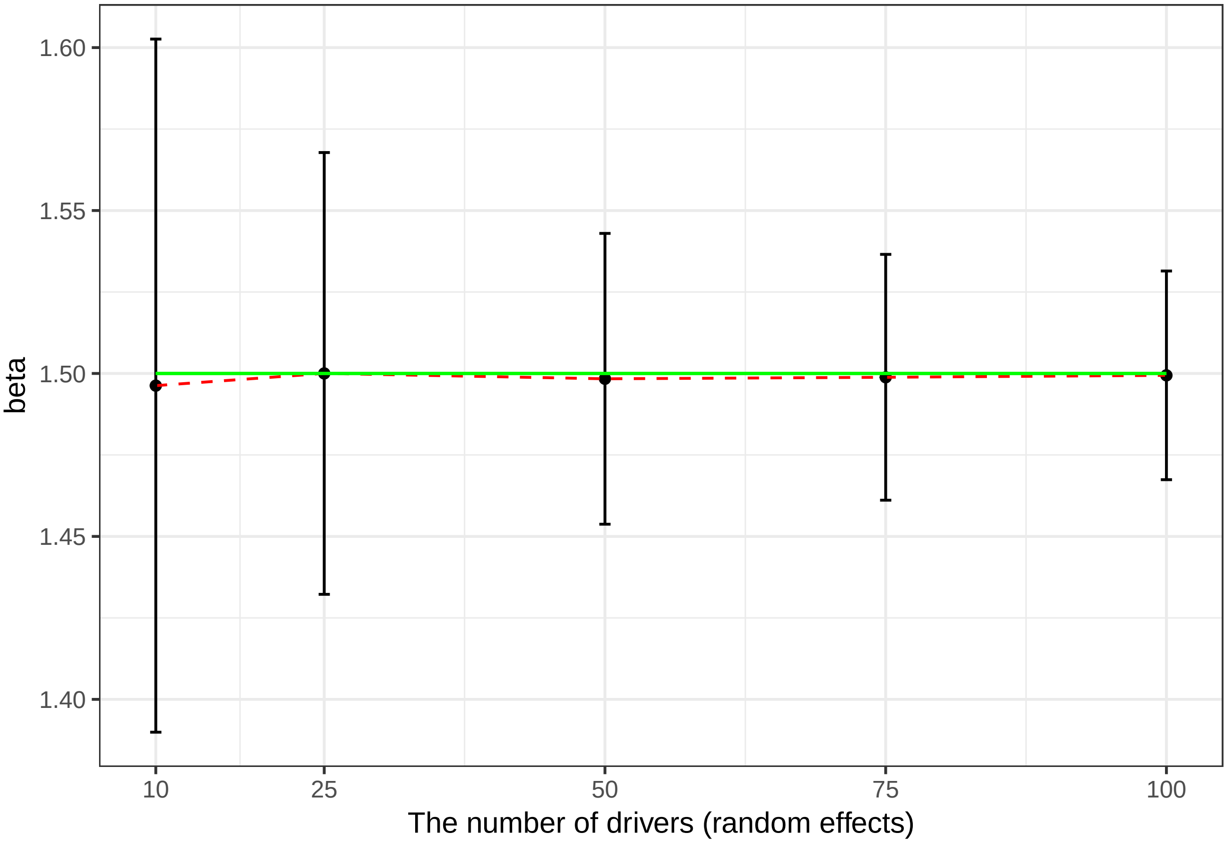
Contents

1 Revised code

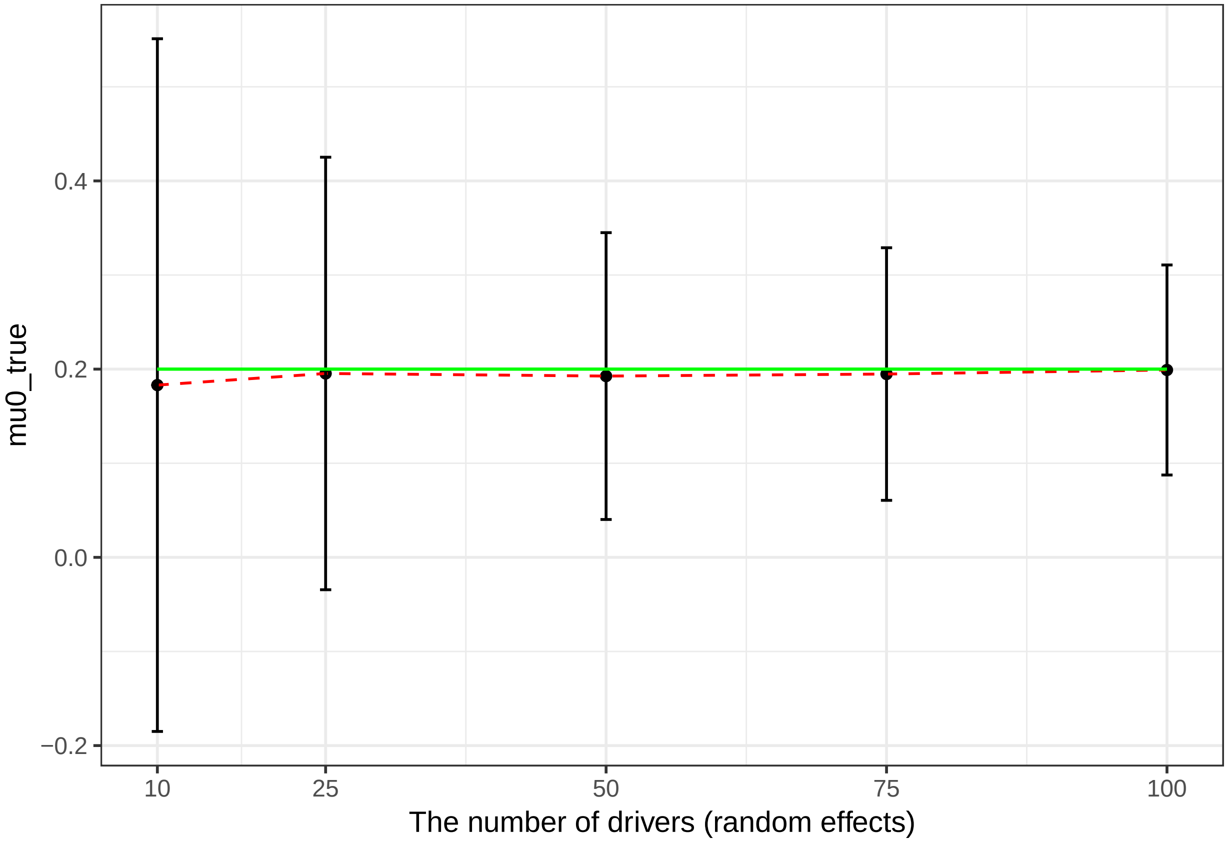
```
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
  real nhppnoevent_lp(real tau, real beta, real theta){
    real loglikelihood = - (tau/theta)^beta;
    return(loglikelihood);
  }
}
data {
  int<lower=1> N; //total # of failures
  int<lower=1> K; //number of predictors
  int<lower=1> S; //total # of shifts
  int<lower=1> D; //total # of drivers
  int<lower=1> id[S];//driver index, must be an array
  vector<lower=0>[S] tau;//truncated time
  vector<lower=0>[N] event_time; //failure time
  int group_size[S]; //group sizes
  matrix[S, K] X_predictors;//predictor variable matrix
}
transformed data{
  matrix[S, K] X_centered;
  vector[K] X_means;
  for(k0 in 1:K){
    X_means[k0] = mean(X_predictors[, k0]);
    X_centered[,k0] = X_predictors[, k0] - X_means[k0];
  }
}
parameters{
  real mu0; // hyperparameter
  real<lower=0> sigma0;// hyperparameter
  real<lower=0> beta;
  vector[K] R1_K; // fixed parameters
  vector[D] R0; // random intercept
}
transformed parameters{
  //vector[S] r_1_1 = ()
}
model{
  int position = 1;
  vector[S] theta_temp;

  for (s0 in 1:S){
    theta_temp[s0] = exp(R0[id[s0]] + X_centered[s0,]*R1_K);
  }

  for (s1 in 1:S){
    if(group_size[s1] == 0) {
      target += nhppnoevent_lp(tau[s1], beta, theta_temp[s1]);
    }else{
      segment(event_time, position, group_size[s1]) ~ nhpp(beta, theta_temp[s1], tau[s1]);
      position += group_size[s1];
    }
  }
  beta ~ gamma(1, 1);
  R0 ~ normal(mu0, sigma0);
  R1_K  ~ normal(0, 10);
  mu0 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
  //theta_temp ~ gamma(1, 0.01);
}
generated quantities{
  real mu0_true = mu0 - dot_product(X_means, R1_K);
  vector[D] R0_true = R0 - dot_product(X_means, R1_K);
  //real theta_correct = theta_temp - dot_product(X_centered, R1_K);
}
```

2 Estimation results - 500 simulations for each sample size

2.1 beta

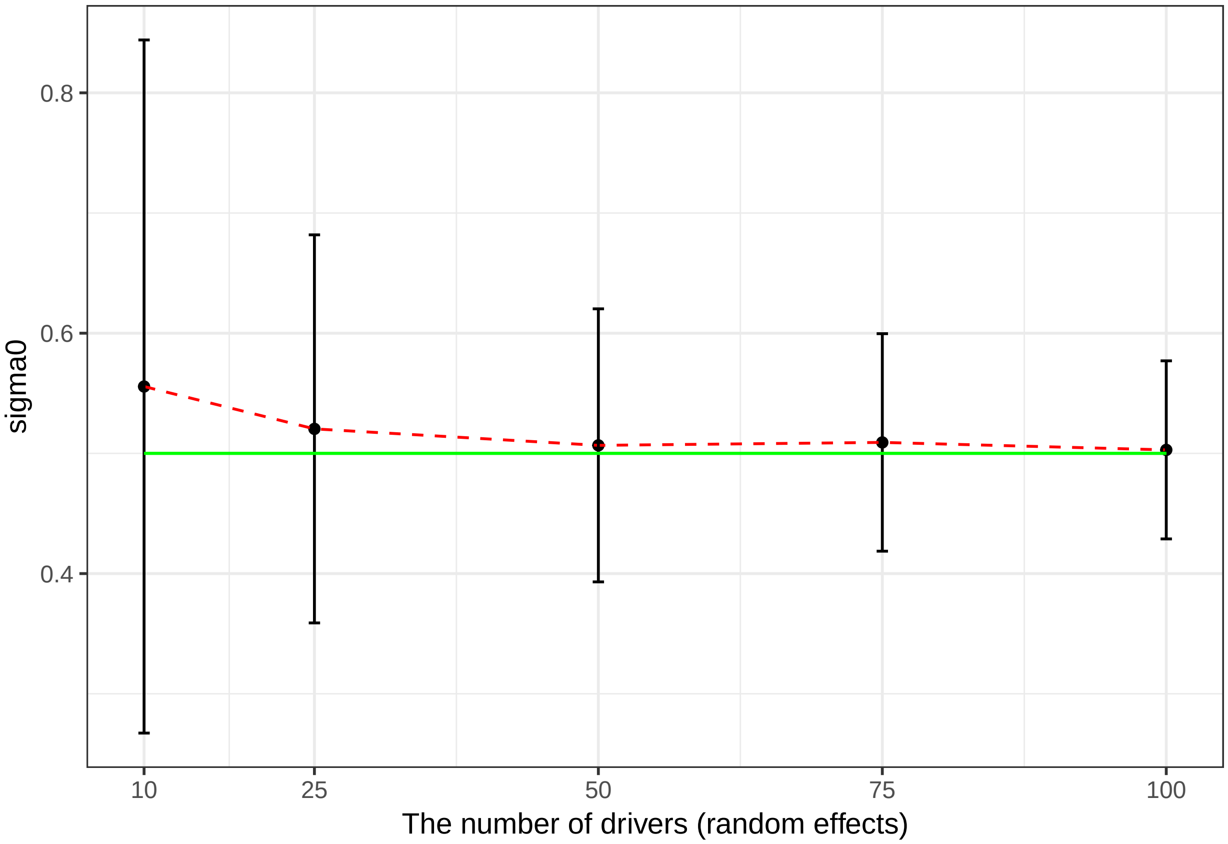


2.2 mu0

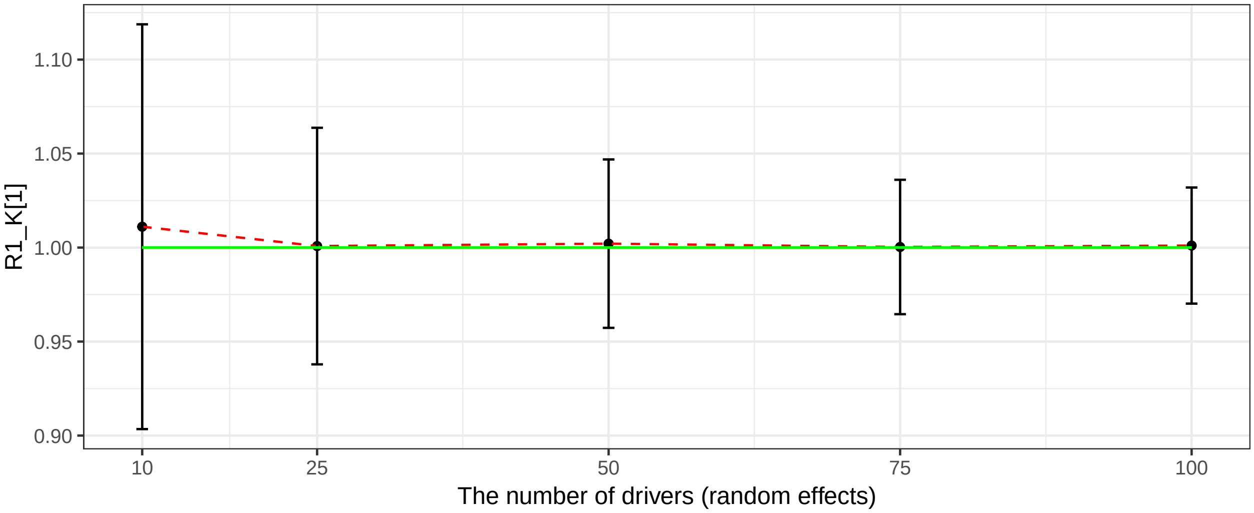


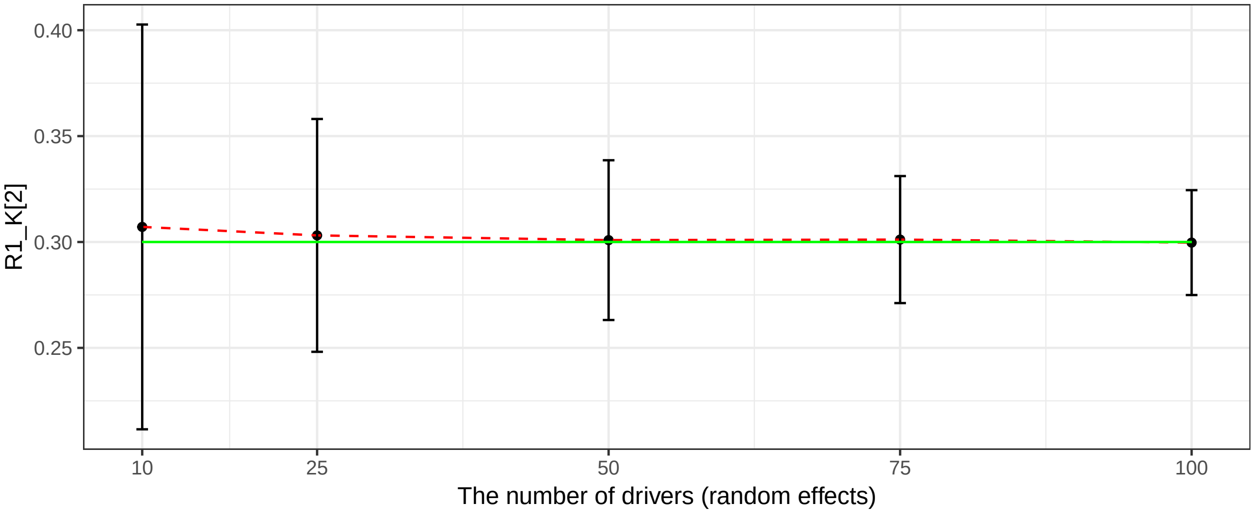
mu0_true
0.4
0.2
0.0
−0.2
10
25
50
75
100
The number of drivers (random effects)

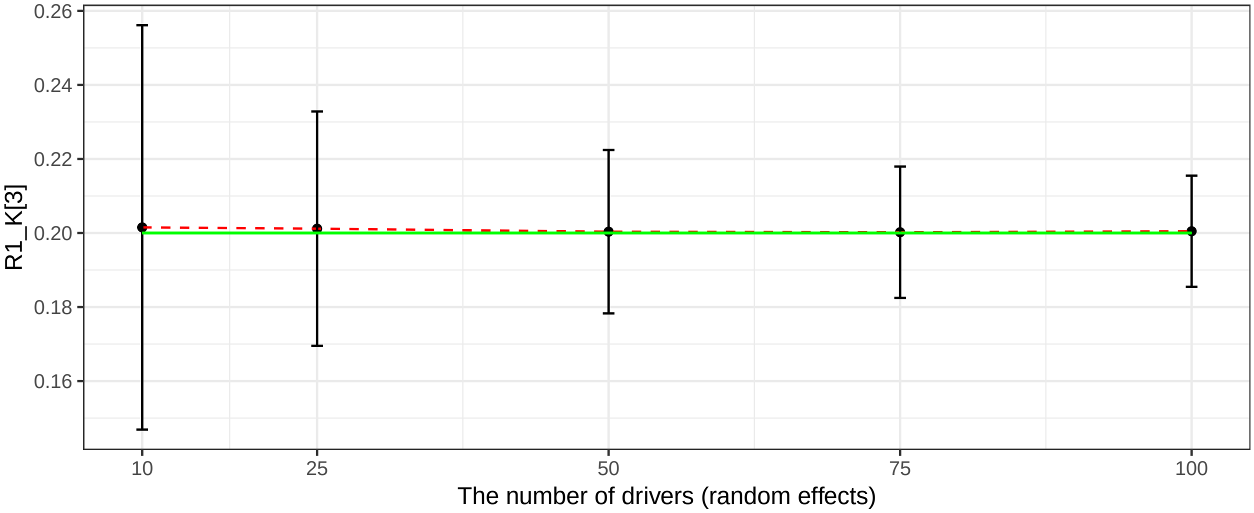
2.3 sigma0


0.8
0.6
0.4
sigma0
10
25
50
75
100
The number of drivers (random effects)

2.4 Fixed parameters


R1_K[1]
The number of drivers (random effects)


R1_K[2]
The number of drivers (random effects)


R1_K[3]
The number of drivers (random effects)