Bayesian Hierarchical Bernoulli Regression

JQT paper with 200 drivers - Model 1

Miao Cai miao.cai@slu.edu

2018-10-25

1 Bernoulli Distribution

2 Bernoulli Regression

Here we model the probability of a critical event occurred using a Bayesian hierarchical Bernoulli regression. We categorize the number of safety events during a trip into a binary variable Y of either 0 or 1, where 0 indicates that no critical event occurred during that trip while 1 indicates that at least 1 critical event occurred during the trip. Since each trip i has a different travel time t_i, we derived the Bernoulli distribution parameter p_i using a Poisson distribution, with the parameter λ_i represented by a linear combination of β_i and x_i.

$$
\begin{aligned}
P_i &= P(\text{at least one event in trip i}) \\
&= 1 - P(\text{no event in trip i}) \\
&= 1 - \frac{e^{-t_i\lambda_i}(t_i\lambda_i)^0)}{0!} \\
&= 1 - \exp(-t_i\lambda_i) \\
&= 1 - \exp(-t_i e^{\beta_0+\beta_i x_i})
\end{aligned} \tag{1}
$$

Transform that into a linear function of β_i, x_i and t_i

$$
\begin{aligned}
1 - P_i &= \text{EXP}(-t_i e^{\beta_0+\beta_i x_i}) \\
\log(1 - P_i) &= -t_i e^{\beta_0+\beta_i x_i} \\
\log\frac{1}{1-P_i} &= e^{\beta_0+\beta_i x_i + \log(t_i)} \\
\log\Big(\log\frac{1}{1-P_i}\Big) &= \beta_0 + \beta_i x_i + \log(t_i)
\end{aligned} \tag{2}
$$

Then, the random effects logistic model is

$$
\begin{aligned}
Y_i &\sim \text{Bern}(P_i) \\
\log\Big(\log\frac{1}{1-P_i}\Big) &= \beta_{0,d(i)} + \beta_{1,d(i)} \cdot \text{CT}_i + \xi \cdot \mathbf{W} + \nu \cdot \mathbf{D_i} + \log(t_i)
\end{aligned} \tag{3}
$$

Here the trip is indexed by i, Y_i is the binary outcome variable of whether at least one critical event occurred in trip i; $d(i)$ is the driver for trip i, $\beta_{0,d(i)}$ is the random intercept for driver $d(i)$; $\beta_{1,d(i)}$ is the random slope for the cumulative time (CTi) of driving in the shift (the sum of driving time for all previous trips) for driver $d(i)$; $\mathbf{W}$ is a vector of external environment fixed effects, including precipitation intensity and probability, visibility, and whether it was sunrise or sunset time; $\mathbf{D}_i$ are driver level fixed effects, including age group and business unit; t_i is the travel time for the trip i.

We assume that the drivers are random effects, and we assume exchangeable priors of the form

$$\beta_{0,d(1)}, \beta_{0,d(2)}, \ldots, \beta_{0,d(n)} \sim \text{i.i.d.} N(\mu_0, \sigma_0^2)$$

and

$$\beta_{1,d(1)}, \beta_{1,d(2)}, \ldots, \beta_{1,d(n)} \sim \text{i.i.d.} N(\mu_1, \sigma_1^2)$$

The parameters μ_0, σ_0, μ_1, and σ_1 are hyperparameters with priors. Since we do not have much prior knowledge on the hyperparameters, we assigned diffuse priors for these hyperparameters.

$$\begin{aligned}\mu_0 &\sim N(0, 10^2)\\ \mu_1 &\sim N(0, 10^2)\\ \sigma_0 &\sim \text{GAMMA}(1,1)\\ \sigma_1 &\sim \text{GAMMA}(1,1)\end{aligned} \tag{4}$$

Since μ_0 and μ_1 can be any real number, so we assigned two normal distributions with mean of 0 and standard deviation of 10 as the priors for these two hyperparameters. In comparison, σ_0 and σ_1 must be strictly positive, so we assigned $\text{GAMMA}(1,1)$ with wide distribution on positive real numbers as their priors.

3 Stan code

3.1 Centered parameterization

```
library(rstan)
library(rstanarm)
library(shinystan)
options(mc.cores=parallel::detectCores())









load("w2wdriver.Rdata")
w2wdriver = w2wdriver[!is.na(w2wdriver$Age),]

w1000 = w2wdriver

w1000$JBI00 = 0 # DCS00 as reference
w1000$JBI00[w1000$BUSINESS_UNIT == "JBI00"] = 1
w1000$VAN00 = 0
w1000$VAN00[w1000$BUSINESS_UNIT == "VAN00"] = 1 # DCS00 as reference

w1000$driver_num = as.integer(factor(w1000$driver_num)) # reorder driver number
w1000$visibility[is.na(w1000$visibility)] = mean(w1000$visibility, na.rm = T)

datstan = list(n = nrow(w1000),
               k = max(w1000$driver_num),
               driver_num = w1000$driver_num,
               travelTime = w1000$travelTime,
               event_i = w1000$event_i,
               drivetime_cum = w1000$drivetime_cum,
               Age = w1000$Age,
               JBI00 = w1000$JBI00,
               VAN00 = w1000$VAN00,
               visibility = w1000$visibility,
               precipIntensity = w1000$precipIntensity,
               precipProbability = w1000$precipProbability)
```

```
codestan = '
data {
  int<lower=0> n; // total # of obs
  int<lower=0> k; // # of drivers

  int<lower=0> driver_num[n]; //driver id
  int<lower=0> event_i[n]; //binary outcome
  real<lower=0> drivetime_cum[n]; //cumulative time of driving
  real<lower=0> travelTime[n];
  int<lower=0> Age[n]; //precipitation
  int<lower=0> JBI00[n];
  int<lower=0> VAN00[n];
  real<lower=0> visibility[n];
  real<lower=0> precipIntensity[n];
  real<lower=0> precipProbability[n];
}
parameters{
  vector[k] beta0;
  vector[k] beta1;
  real b_age;
  real b_JBI;
  real b_VAN;
  real b_visibility;
  real b_prec_inten;
  real b_prec_prob;
  real mu0;
  real mu1;
  real<lower=0> sigma0;
  real<lower=0> sigma1;
}

model{
//LIKELIHOOD
  vector[n] theta;

  for(i in 1:n){
  theta[i] = 1 - exp(-travelTime[i]* exp(beta0[driver_num[i]] + beta1[driver_num[i]]*drivetime_cum[i] +
  }
  event_i ~ bernoulli(theta);
  //HYPERPRIORS
  mu0 ~ normal(0, 10);
  mu1 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
  sigma1 ~ gamma(1, 1);
  //PRIORS
  b_age ~ normal(0, 10);
  b_JBI ~ normal(0, 10);
  b_VAN ~ normal(0, 10);
  b_visibility ~ normal(0, 10);
  b_prec_inten ~ normal(0, 10);
  b_prec_prob ~ normal(0, 10);
  beta0 ~ normal(mu0, sigma0);
  beta1 ~ normal(mu1, sigma1);
}
'



hfitnonstandlogit <- stan(model_code=codestan, model_name="hospitals1", data=datstan, iter=200,warmup =


#doctoralsym2018 = hfitnonstandlogit
#save(doctoralsym2018, file = "doctoralsym2018.Rdata")

#save(hfitnonstandlogit, file = "hfitnonstandlogit.Rdata")
shinystan::launch_shinystan(doctoralsym2018)

#shinystan::launch_shinystan(hfitnonstandlogit)
```

3.2 Non-centered parameterization

```
library(rstan)
library(rstanarm)
library(shinystan)
options(mc.cores=parallel::detectCores())



load("w2wdriver.Rdata")
w2wdriver = w2wdriver[!is.na(w2wdriver$Age),]

w1000 = w2wdriver

w1000$JBI00 = 0 # DCS00 as reference
w1000$JBI00[w1000$BUSINESS_UNIT == "JBI00"] = 1
w1000$VAN00 = 0
w1000$VAN00[w1000$BUSINESS_UNIT == "VAN00"] = 1 # DCS00 as reference

w1000$driver_num = as.integer(factor(w1000$driver_num)) # reorder driver number
w1000$visibility[is.na(w1000$visibility)] = mean(w1000$visibility, na.rm = T)
```

```r
datstan = list(n = nrow(w1000),
               k = max(w1000$driver_num),
               driver_num = w1000$driver_num,
               travelTime = w1000$travelTime,
               event_i = w1000$event_i,
               drivetime_cum = w1000$drivetime_cum,
               Age = w1000$Age,
               JBI00 = w1000$JBI00,
               VAN00 = w1000$VAN00,
               visibility = w1000$visibility,
               precipIntensity = w1000$precipIntensity,
               precipProbability = w1000$precipProbability)




codestan = '
data {
  int<lower=0> n; // total # of obs
  int<lower=0> k; // # of drivers

  int<lower=0> driver_num[n]; //driver id
  int<lower=0> event_i[n]; //binary outcome
  real<lower=0> drivetime_cum[n]; //cumulative time of driving
  real<lower=0> travelTime[n];
  int<lower=0> Age[n]; //precipitation
  int<lower=0> JBI00[n];
  int<lower=0> VAN00[n];
  real<lower=0> visibility[n];
  real<lower=0> precipIntensity[n];
  real<lower=0> precipProbability[n];
}
parameters{
  vector[k] beta0;
  vector[k] beta1;
  real b_age;
  real b_JBI;
  real b_VAN;
  real b_visibility;
  real b_prec_inten;
  real b_prec_prob;
  real mu0;
  real mu1;
  real<lower=0> sigma0;
  real<lower=0> sigma1;
}

model{
//LIKELIHOOD
  vector[n] theta;

  for(i in 1:n){
  theta[i] = 1 - exp(-1*travelTime[i]* exp(mu0 + beta0[driver_num[i]]*sigma0 + (mu1 + beta1[driver_num[
  }

  event_i ~ bernoulli(theta);
  //HYPERPRIORS
  mu0 ~ normal(0, 10);
  mu1 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
  sigma1 ~ gamma(1, 1);
  //PRIORS
  b_age ~ normal(0, 10);
  b_JBI ~ normal(0, 10);
  b_VAN ~ normal(0, 10);
  b_visibility ~ normal(0, 10);
  b_prec_inten ~ normal(0, 10);
  b_prec_prob ~ normal(0, 10);
  beta0 ~ normal(0, 1);
  beta1 ~ normal(0, 1);
}
'



doctoralsym2018 <- stan(model_code=codestan, model_name="hospitals1", data=datstan, iter=200,warmup = 1

save(doctoralsym2018, file = "doctoralsym2018.Rdata")


shinystan::launch_shinystan(doctoralsym2018)
```