(Not That) Advanced Hierarchical Models

Ben Goodrich

StanCon: January 10, 2018

Obligatory Disclosure

- Ben is an employee of Columbia University, which has received several research grants to develop Stan
- Ben is also a manager of GG Statistics LLC, which utilizes Stan for business purposes
- According to Columbia University policy, any such employee who has any equity stake in, a title (such as officer or director) with, or is expected to earn at least $5,000.00 per year from a private company is required to disclose these facts in presentations

Goals for the Tutorial

- Think about conditional distributions, the building blocks for hierarchical models
- Practice writing functions in the Stan language to draw from the prior predictive distribution
- Write simple Stan programs where some parameters are functions of other parameters
- Prepare for more advanced material tomorrow and Friday at 7AM

Hierarchical Data Generating Processes: Bowling

- How to model how person i does on the jth bowling frame?

Hierarchical Data Generating Processes: Bowling

- How to model how person *i* does on the *j*th bowling frame?
- You would need (at least) two probability distributions:
  1. Probability of knocking down $0, 1, \ldots, 10$ pins on the first roll
  2. Probability of knocking down $0, 1, \ldots, 10$ pins on the second roll, given what transpired on the first roll

```
first_roll <- sample(0:10, size = 1)
pins_left <- 10 - first_roll
second_roll <- sample(0:pins_left, size = 1)
first_roll + second_roll

## [1] 9
```

Hierarchical Data Generating Processes: IV

A generative model for an instrumental variable (IV) design is

$$\begin{aligned} \sigma_1 &\sim \text{Exponential}(r_1) \\ \text{Priors: } \sigma_2 &\sim \text{Exponential}(r_2) \\ \rho &\sim \text{Uniform}(-1,1) \\ \text{Errors: } \begin{bmatrix} \nu_i \\ \varepsilon_i \end{bmatrix} &\sim \mathcal{N}_2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}\right) \forall i \end{aligned}$$

$$\text{Priors: } \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \sim \mathcal{N}_3(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \qquad \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \sim \mathcal{N}_3(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$

$$\begin{aligned} \text{1st stage: } t_i &\equiv \alpha_0 + \alpha_1 x_i + \alpha_2 z_i + \nu_i \forall i \\ \text{2nd stage: } y_i &\equiv \beta_0 + \beta_1 x_i + \beta_2 t_i + \varepsilon_i \forall i \end{aligned}$$

Write a Stan function to draw *N* times from this DGP

```
vector[] IV_DGP_rng(int N, vector x, vector z, real r1,
                    real r2, vector mu_1, matrix Sigma_1,
                    vector mu_2, matrix Sigma_2) {
```

Write a Stan function to draw *N* times from this DGP

```
vector[] IV_DGP_rng(int N, vector x, vector z, real r1,
                    real r2, vector mu_1, matrix Sigma_1,
                    vector mu_2, matrix Sigma_2) {
  real sigma_1 = exponential_rng(r1);
  real sigma_2 = exponential_rng(r2);
  real cov_12 = uniform_rng(-1, 1) * sigma_1 * sigma_2;
  matrix[2,2] Sigma = [ [square(sigma_1), cov_12],
                        [cov_12, square(sigma_2)] ];
  vector[3] alpha = multi_normal_rng(mu_1, Sigma_1);
  vector[3] beta = multi_normal_rng(mu_2, Sigma_2);
  vector[N] ty[2] = {alpha[1] + alpha[2] * x + alpha[3] *
                     z, beta[1] + beta[2] * x};
  vector[2] zeros = rep_vector(0, 2);
  for (n in 1:N) {
    vector[2] errors = multi_normal_rng(zeros, Sigma);
    ty[1][n] += errors[1];
    ty[2][n] += beta[3] * ty[1][n] + errors[2];
  }
  return ty;
}
```

Exposing Stan Functions in R

- If you put the previous function inside the functions block of an otherwise empty Stan program, you can export it to R

```
rstan::expose_stan_functions("IV_DGP.stan")
args(IV_DGP_rng)
```

- At this point, you can call the `IV_DGP_rng` function with appropriate arguments and get back a list of two numeric vectors

Coefficients Depending on Other Coefficients

Write a simple Stan program where the coefficient on age is a linear function of the person's income, starting with

```
data {
  int<lower=1> N;
  vector[N] age;
  vector[N] income;
  int<lower=0,upper=1> vote[N];
}
parameters {
  vector[2] lambda; // intercept / slope for age's effect
  vector[2] beta;  // intercept / slope for outcome
}
```

Coefficients Depending on Other Coefficients

Write a simple Stan program where the coefficient on age is a linear function of the person's income, starting with

```
data {
  int<lower=1> N;
  vector[N] age;
  vector[N] income;
  int<lower=0,upper=1> vote[N];
}
parameters {
  vector[2] lambda; // intercept / slope for age's effect
  vector[2] beta;  // intercept / slope for outcome
}

model {
  vector[N] beta_age = lambda[1] + lambda[2] * income;
  vector[N] eta = beta[1] + beta[2] * income
                + beta_age .* age;
  target += bernoulli_logit_lpmf(vote | eta);
}  // priors on lambda and beta
```

Relation to Interaction Terms in R

- If

$$\begin{aligned} \eta_i &= \beta_1 + \beta_2 \times \text{Income}_i + \beta_{3i} \times \text{Age}_i \\ \beta_{3i} &= \lambda_1 + \lambda_2 \times \text{Income}_i \end{aligned}$$

  then by substituting & distributing:

$$\begin{aligned} \eta_i &= \beta_1 + \beta_2 \times \text{Income}_i + (\lambda_1 + \lambda_2 \times \text{Income}_i) \times \text{Age}_i \\ &= \beta_1 + \beta_2 \times \text{Income}_i + \lambda_1 \times \text{Age}_i + \lambda_2 \times \text{Income}_i \times \text{Age}_i \end{aligned}$$

  and β_1, β_2, λ_1, and λ_2 can be estimated (unregularized) via

```
glm(vote ~ income + age + income:age, family = binomial)
```

- Stan version is easier to interpret; R version is quick

Coefficients Depending on Other Coefficients Again

Write a simple Stan program where the coefficient on age is a **noisy** linear function of the person's income, starting with

```
data {
  int<lower=1> N; vector[N] age;
  vector[N] income; int<lower=0,upper=1> vote[N];
}
parameters {
  vector[2] lambda; // intercept / slope for age's effect
   vector[N] noise; // error in effect of age
   real<lower=0> sigma; // sd of error in beta_age
  vector[2] beta;  // intercept / slope for outcome
}
```

Coefficients Depending on Other Coefficients Again

Write a simple Stan program where the coefficient on age is a **noisy** linear function of the person's income, starting with

```
data {
  int<lower=1> N; vector[N] age;
  vector[N] income; int<lower=0,upper=1> vote[N];
}
parameters {
  vector[2] lambda; // intercept / slope for age's effect
  vector[N] noise; // error in effect of age
  real<lower=0> sigma; // sd of error in beta_age
  vector[2] beta;  // intercept / slope for outcome
}
model {
  vector[N] beta_age = lambda[1] + lambda[2] * income
                     + sigma * noise; // non-centering
  vector[N] eta = beta[1] + beta[2] * income
                + beta_age .* age;
  target += bernoulli_logit_lpmf(vote | eta);
  target += normal_lpdf(noise | 0, 1);
}  // priors on lambda, sigma, and beta
```

Relation to "Random Coefficient Models"

- Previous model cannot be estimated via `glm` in R
- In order for MLEs to be consistent as $N \uparrow \infty$, the number of parameters to estimate must remain fixed. So, `noise` couldn't be considered a parameter in the previous model.
- A "random coefficient model" (RCM) would consider `noise` to be "error" and integrate it out of the likelihood function
- For Gaussian outcomes, this can be done analytically; otherwise it must be done numerically using quadrature
- Can then use MLE to obtain parameter point estimates: $\widehat{\boldsymbol{\lambda}}$, $\widehat{\sigma}$, and $\widehat{\boldsymbol{\beta}}$
- Bayesians take `noise` to be a parameter, draw from the conditional distribution of all parameters given the data, and ignore posterior margins that are not interesting

Cluster Sampling Designs

- Classic example of cluster sampling:
  1. Randomly draw J schools from the population of schools
  2. For each selected school, randomly draw N_j students
  3. Collect data on these $N = \sum_{j=1}^{J} N_j$ students
- If one tried to replicate this study, both the schools and the students would be different than in the original study

$$
\begin{aligned}
\tau &\sim \text{Exponential}(r_\tau) \\
\alpha_j &\sim \mathcal{N}(0, \tau)\ \forall j \\
\beta &\sim \mathcal{N}(\mu_\beta, \sigma_\beta) \\
\sigma_\varepsilon &\sim \text{Exponential}(r_\sigma) \\
\varepsilon_{ij} &\sim \mathcal{N}(0, \sigma_\varepsilon) \\
y_{ij} &\equiv \alpha_j + \beta \times \text{class\_size}_i + \varepsilon_{ij}\ \forall i, j
\end{aligned}
$$

Write a Stan function to draw from this DGP

```
vector cluster_DGP_rng(int J, int[] N, vector class_size,
                       real r_tau, real r_sigma,
                       real mu_beta, real sigma_beta) {
```