(Somewhat) Advanced Hierarchical Models

Ben Goodrich

StanCon: January 11, 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

- Yesterday, we talked a lot about models where the probability distribution for some variable depends on another variable
- Today, we are going to talk about the case where probability distribution for some parameter depends on another parameter
- Richard McElreath argues that these hierarchical models should be the default approach to modeling
- Learn about how to estimate hierarchical models with the **rstanarm** and **brms** R packages

Hierarchical Models

- A hierarchical model is one where a prior is specified on a parameter conditional on another unknown parameter
- Hierarchical models are often used in situations to allow parameters to vary by categorical group
- Suppose there are J groups & N_j observations in jth group
- Best way to think about such structures:
    - There is a likelihood contribution for the jth group
    - There are priors over how parameters vary across groups
    - There are priors on parameters common to all groups
- Relevant prior information pertains to how similar you believe the groups' data-generating processes to be

School Example

This is a data-generated process we talked about yesterday:

$$\begin{aligned}
\tau &\sim \text{Exponential}\left(r_\tau\right) \\
\alpha_j &\sim \mathcal{N}\left(0, \tau\right) \forall j \text{ [hierarchical]} \\
\beta &\sim \mathcal{N}\left(\mu_\beta, \sigma_\beta\right) \\
\sigma &\sim \text{Exponential}\left(r_\sigma\right) \\
\epsilon_{ij} &\sim \mathcal{N}\left(0, \sigma\right) \\
y_{ij} &\equiv \alpha_j + \beta \times \text{class_size}_i + \epsilon_{ij} \forall i, j
\end{aligned}$$

Stan Function for School Example

```
vector cluster_DGP_rng(int J, int[] N, vector class_size,
                       real r_tau, real r_sigma,
                       real mu_beta, real sigma_beta) {
  real tau = exponential_rng(r_tau);
  real sigma = exponential_rng(r_sigma);
  real beta = normal_rng(mu_beta, sigma_beta);
  vector[sum(N)] y;
  int pos = 1;
  for (j in 1:J) {
    real alpha_j = normal_rng(0, tau);
    for (i in 1:N[j]) {
      real mu = alpha_j + beta * class_size[pos];
      y[pos] = mu + normal_rng(0, sigma);
      pos += 1;
    }
  }
  return y;
}
```

Restatement of the Hierarchical Linear Model

- Generally, both intercepts & slopes can vary across groups
- Let $\boldsymbol{\beta}_j = \boldsymbol{\beta} + \mathbf{b}_j$ and $\mathbf{b}^\top = \begin{bmatrix} \mathbf{b}_1^\top & \mathbf{b}_2^\top & \cdots & \mathbf{b}_J^\top \end{bmatrix}$. Rewrite the model as $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon}$ where $\mathbf{Z}$ is a sparse matrix that basically interacts $\mathbf{X}$ with group-specific dummy variables.
- Bayesians: $\mathbf{b} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}\left(\boldsymbol{\theta}\right)\right)$ and $\mathbf{y} \sim \mathcal{N}\left(\mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b}, \sigma^2\mathbf{I}\right)$
- For frequentists, each $\mathbf{b}_j$ is not a fixed "parameter" but rather a random variable that is part of the error term
- Frequentists integrate out each $\mathbf{b}_j$ and then choose $\widehat{\boldsymbol{\beta}}, \widehat{\sigma}$, and $\boldsymbol{\Sigma}\left(\widehat{\boldsymbol{\theta}}\right)$ to maximize $\mathbf{y} \sim \mathcal{N}\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{Z}\boldsymbol{\Sigma}\left(\boldsymbol{\theta}\right)\mathbf{Z}^\top\right)$
- Technically, $\widehat{\mathbf{b}}_j$ is not "estimated" but rather "predicted" from the residuals $\mathbf{e} = \mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}$ by subsequently regressing $\mathbf{e}$ on $\mathbf{Z}$

Table 2 from the **lme4** Vignette (frequentist)

Formula	Alternative	Meaning
`(1 | g)`	`1 + (1 | g)`	Random intercept with fixed mean
`0 + offset(o) + (1 | g)`	`-1 + offset(o) + (1 | g)`	Random intercept with *a priori* means
`(1 | g1/g2)`	`(1 | g1)+(1 | g1:g2)`	Intercept varying among `g1` and `g2` within `g1`
`(1 | g1)+(1 | g2)`	`1 + (1 | g1) + (1 | g2)`	Intercept varying among `g1` and `g2`
`x + (x | g)`	`1 + x + (1 + x | g)`	Correlated random intercept and slope
`x + (x || g)`	`1 + x + (1 | g) + (0 + x | g)`	Uncorrelated random intercept and slope

Table 2: Examples of the right-hand sides of mixed-effects model formulas. The names of grouping factors are denoted `g`, `g1`, and `g2`, and covariates and *a priori* known offsets as `x` and `o`.

Bayesian Implementations with **lme4** / **mgcv** Syntax

- The **rstanarm** and **brms** packages accept the same | syntax as **`lme4`**
- Both also permit the same `s(...)` syntax as **mgcv** to use smooth, non-linear functions of parameters
- Add arguments for the priors on α, $\boldsymbol{\beta}$, σ, etc.
- Try **rstanarm** and / or **brms** first to make sure your data are amenable to a hierarchical model

Tadpole Example from McElreath, chapter 12

```r
GH <- "https://raw.githubusercontent.com/"
FILE <- "rmcelreath/rethinking/master/data/reedfrogs.csv"
reedfrogs <- read.table(paste0(GH, FILE), sep = ";",
                        header = TRUE)
reedfrogs$tank <- as.factor(1:nrow(reedfrogs)) # groups
library(rstanarm); options(mc.cores = parallel::detectCores())
post <- stan_glmer(cbind(surv, density - surv) ~ size +
                   (1 | tank), data = reedfrogs,
                   family = binomial('logit'))
```

```r
dim(as.matrix(post)) # raw draws from posterior distribution

## [1] 4000   51
```

Results of Tadpole Example from McElreath

```
## stan_glmer
##  family:       binomial [logit]
##  formula:      cbind(surv, density - surv) ~ size + (1 | tank)
##  observations: 48
## ------
##
## Estimates:
##             Median MAD_SD
## (Intercept) 1.2    0.4
## sizesmall   0.4    0.5
##
## Error terms:
##  Groups Name        Std.Dev.
##  tank   (Intercept) 1.6
## Num. levels: tank 48
##
## Sample avg. posterior predictive
## distribution of y (X = xbar):
##          Median MAD_SD
## mean_PPD 16.3   0.4
##
```

More Results of Tadpole Example from McElreath

```
fixef(post)

## (Intercept)   sizesmall
##   1.1787690   0.4306905

NROW(ranef(post)$tank)

## [1] 48

head(cbind(coef(post)$tank[,1],
           fixef(post)[1] + ranef(post)$tank))

##   coef(post)$tank[, 1] (Intercept)
## 1             2.040190    2.040190
## 2             2.921214    2.921214
## 3             0.952524    0.952524
## 4             2.922820    2.922820
## 5             1.712078    1.712078
## 6             1.713280    1.713280
```

A Smooth Nonlinear Model with **brms**

```r
library(brms)
post2 <- brm(y ~ s(roach1) + treatment,
             data = roaches, family = poisson)
```

- The `s(roach1)` says that the logarithm of the conditional number of roaches is a spline function of the previous number of roaches
- This can also be represented in $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b}$ form
- Simon Wood has a new edition of his book out

Results of Nonlinear Model

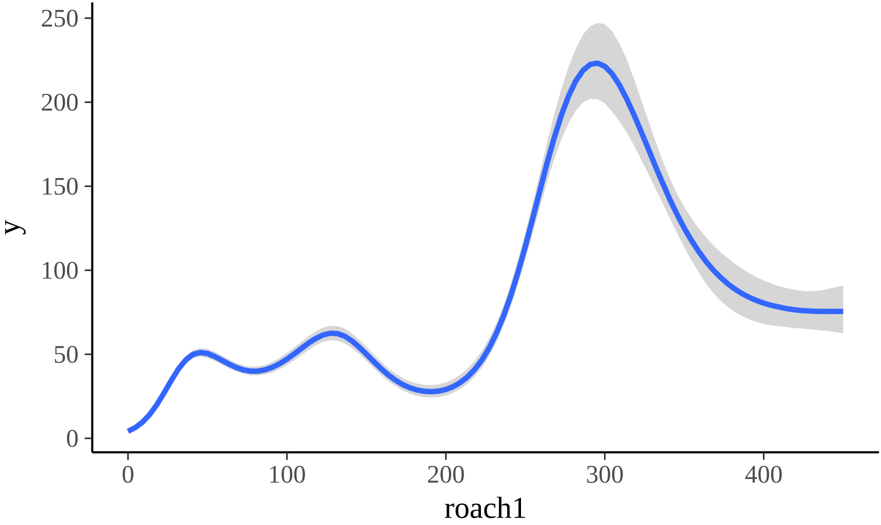
```
 Family: poisson
  Links: mu = log
Formula: y ~ s(roach1) + treatment
   Data: roaches (Number of observations: 262)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
    ICs: LOO = NA; WAIC = NA; R2 = NA

Smooth Terms:
               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rha
sds(sroach1_1)    11.53      2.99     7.31    18.95        552 1.0

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     2.96      0.02     2.91     3.01       2948 1.00
treatment    -0.66      0.02    -0.71    -0.62       2341 1.00
sroach1_1     3.48      0.15     3.20     3.77       1832 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.S
is a crude measure of effective sample size, and Rhat is the poter
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Plot of Nonlinear Model



Using **brms** to Generate Stan Programs

```
str(make_standata(y ~ s(roach1) + treatment,
                  data = roaches, family = poisson),
    give.attr = FALSE)

## List of 8
##  $ N         : int 262
##  $ Y         : int [1:262(1d)] 153 127 7 7 0 0 73 24 2 2 ...
##  $ nb_1      : int 1
##  $ knots_1   : int [1(1d)] 8
##  $ Zs_1_1    : num [1:262, 1:8] 0.0427 0.07292 -0.00439 -0.0051
##  $ K         : int 3
##  $ X         : num [1:262, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
##  $ prior_only: int 0
```

```
make_stancode(y ~ s(roach1) + treatment,
              data = roaches, family = poisson)
```

Data and Transformed Data Blocks

```
data {
  int<lower=1> N;  // total number of observations
  int Y[N];  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  // data of smooth s(roach1)
  int nb_1;  // number of bases
  int knots_1[nb_1];
  matrix[N, knots_1[1]] Zs_1_1;
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, K - 1] Xc;  // centered version of X
  vector[K - 1] means_X;  // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}
```

Remaining Blocks

```
parameters {
  vector[Kc] b;  // population-level effects
  real temp_Intercept;  // temporary intercept
  // parameters of smooth s(roach1)
  vector[knots_1[1]] zs_1_1;
  real<lower=0> sds_1_1;
}
transformed parameters {
  vector[knots_1[1]] s_1_1 = sds_1_1 * zs_1_1;
}
model {
  vector[N] mu = Xc * b + Zs_1_1 * s_1_1 + temp_Intercept;
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, 1.1, 10);
  target += normal_lpdf(zs_1_1 | 0, 1);
  target += student_t_lpdf(sds_1_1 | 3, 0, 10)
    - 1 * student_t_lccdf(0 | 3, 0, 10);
  // likelihood including all constants
  if (!prior_only) {
    target += poisson_log_lpmf(Y | mu);
  }
}
```