Hierarchical Models II

Ben Goodrich

StanCon Helsinki: August 29, 2018

Goals for the Second Session

- Before, we left off by estimating a hierarchical model with the **rstanarm** R package
- Richard McElreath argues that these hierarchical models should be the default approach to modeling
- Learn about how to estimate hierarchical models with the **brms** R package
- Learn about transformations to improve the efficiency of Stan's sampling
- Start to write hierarchical priors in the Stan language

A Smooth Nonlinear Model with **brms**

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

- In a Poisson model, $\mu_i = e^{\eta_i}$ and η_i is a function of covariates
- 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 $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Zb}$ form
- Simon Wood has a new edition of his **mgcv** book out

Results of Nonlinear Model

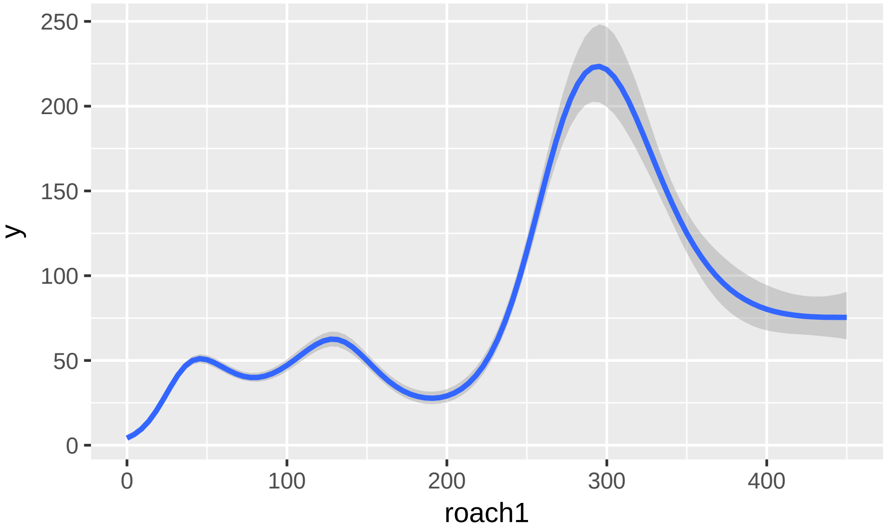
```
 Family: poisson
  Links: mu = log
Formula: y ~ s(roach1) + treatment
   Data: rstanarm::roaches (Number of observations: 262)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
    ICs: LOO = NA; WAIC = NA; R2 = NA

Smooth Terms:
               Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sds(sroach1_1)    11.51      3.02     7.22    19.12        598 1.00

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     2.96      0.02     2.91     3.01       3201 1.00
treatment    -0.66      0.03    -0.71    -0.61       2793 1.00
sroach1_1     3.48      0.15     3.17     3.78       1973 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
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 = rstanarm::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 = rstanarm::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, 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);
  }
}
```

Matt Trick / Non-centered (Re)Parameterization

- Let's simplify to the case where only the intercept varies across groups, i.e. $\alpha_j \sim \mathcal{N}(\alpha, \sigma) \, \forall j$
- $\sigma = e^{\omega}$ is unknown and ω has an improper uniform prior
- $\mathcal{N}(\alpha, \sigma) \stackrel{d}{=} \alpha + \sigma \times \mathcal{N}(0, 1)$ and similarly for other distributions in the location-scale family
- You can often help Stan sample efficiently via transformations

$$\begin{aligned} u_j &\sim \mathcal{N}(0,1) \implies \\ \alpha_j = \alpha + e^{\omega} u_j \forall j &\sim \mathcal{N}(\alpha, e^{\omega}) \end{aligned}$$

  - `vector[J] u` would be declared in the parameters block
  - `vector[J] alpha` would be declared in the transformed parameters block
  - The second derivative with respect to each u_j is constant
- Look at the bivariate prior for α_j, ω vs. that of u_j, ω

Comparison of Bivariate Priors

```r
library(rgl)

kernel <- function(alpha, omega) {
  dnorm(alpha, sd = exp(omega), log = TRUE)
}
LIM <- c(-2,2)
persp3d(kernel, xlim = LIM,
        ylim = LIM, zlab = "log kernel")

reparameterized_kernel <- function(u, omega) {
  dnorm(u, log = TRUE)
}
persp3d(reparameterized_kernel, xlim = LIM,
        ylim = LIM, zlab = "log kernel")
```

Coefficients Depending on Other Coefficients Again

Recall our Stan program where the coefficient on age is a **noisy** linear function of the person's income:

```
data {
  int<lower=1> N; vector[N] age;
  vector[N] income; int<lower=0,upper=1>[N] vote;
}
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 log-odds
}
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 += binomial_logit_lpmf(vote | eta);
  target += normal_lpdf(noise | 0, 1);
}  // priors on lambda, sigma, and beta
```

Centered Parameterization

The following is conceptually the same but often problematic:

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

Multivariate Matt Trick

- If $\boldsymbol{\beta}_j \sim \text{MultiNormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, Stan can have difficulty drawing efficiently from the joint posterior distribution
  - When Σ_{kk} is small, β_{kj} must fall in a narrow range, which entails a small stepsize for NUTS
  - When Σ_{kk} is large, β_{kj} can fall in a wide range, which requires a large stepsize or else many small steps
- You can help Stan with this problem via transformations

$$\begin{aligned} u_{kj} &\sim \text{Normal}(0,1)\,\forall k,j \implies \\ \boldsymbol{\beta}_j = \boldsymbol{\mu} + \sigma \mathbf{L}\mathbf{u}_j &\sim \text{MultiNormal}\left(\boldsymbol{\mu}, \sigma^2 \mathbf{L}\mathbf{L}^\top\right) \end{aligned}$$

  where $\sigma\mathbf{L}$ is the Cholesky factor of $\boldsymbol{\Sigma} = \sigma^2\mathbf{L}\mathbf{L}^\top$ and σ is the standard deviation of the errors
- Both **rstanarm** and **brms** do things like this

Decomposing a Covariance Matrix

- Suppose $\boldsymbol{\beta}_j \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\beta}_j$ is a K-vector for group j
- With Stan, you are free to do what makes sense, such as

$$\begin{aligned}
\boldsymbol{\Sigma} &= \boldsymbol{\Delta\Lambda\Delta} \text{ [sds x correlation x sds]} \\
\Delta_k &\sim \text{Exponential}(r_k)\,\forall k \\
\boldsymbol{\Lambda} &\sim \text{LKJ}(\eta)
\end{aligned}$$

- There is an easy and possibly non-informative prior for a correlation matrix $\boldsymbol{\Lambda}$, $f(\boldsymbol{\Lambda}|\eta) = \frac{1}{c(\eta,K)}|\boldsymbol{\Lambda}|^{\eta-1}$ called “LKJ”
- η acts like the shape parameter of a Beta distribution
  - if $\eta = 1$, $f(\boldsymbol{\Lambda}|\eta) = \frac{1}{c(\eta,K)}$ is constant
  - if $\eta > 1$, $\mathbf{I}$ is the modal correlation matrix and the only correlation matrix with positive density as $\eta \uparrow \infty$
  - if $\eta < 1$, $\mathbf{I}$ is at the trough of the distribution of correlation matrices, which is a weird thing to believe
- Can specify a prior on $\mathbf{L}$ such that $\boldsymbol{\Lambda} = \mathbf{L}\mathbf{L}^\top$ has the LKJ prior

A Multivariate Matt Trick with **brms**

```r
library(brms)
post <- brm(Reaction ~ Days + (Days | Subject),
            data = lme4::sleepstudy)
```

```r
str(make_standata(Reaction ~ Days + (Days | Subject),
                  data = lme4::sleepstudy), give.attr = FALSE)
```

```
## List of 11
##  $ N         : int 180
##  $ Y         : num [1:180(1d)] 250 259 251 321 357 ...
##  $ K         : int 2
##  $ X         : num [1:180, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Z_1_1     : num [1:180(1d)] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Z_1_2     : num [1:180(1d)] 0 1 2 3 4 5 6 7 8 9 ...
##  $ J_1       : int [1:180(1d)] 1 1 1 1 1 1 1 1 1 1 ...
##  $ N_1       : int 18
##  $ M_1       : int 2
##  $ NC_1      : num 1
##  $ prior_only: int 0
```

Data and Transformed Data Blocks

```
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  // data for group-level effects of ID 1
  int<lower=1> J_1[N];
  int<lower=1> N_1;
  int<lower=1> M_1;
  vector[N] Z_1_1;
  vector[N] Z_1_2;
  int<lower=1> NC_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
  real<lower=0> sigma;  // residual SD
  vector<lower=0>[M_1] sd_1;  // group-level standard deviations
  matrix[M_1, N_1] z_1;  // unscaled group-level effects
  // cholesky factor of correlation matrix
  cholesky_factor_corr[M_1] L_1;
}
transformed parameters {
  // group-level effects
  matrix[N_1, M_1] r_1 = (diag_pre_multiply(sd_1, L_1) * z_1)';
  vector[N_1] r_1_1 = r_1[, 1];
  vector[N_1] r_1_2 = r_1[, 2];
}
model {
  vector[N] mu = Xc * b + temp_Intercept;
  for (n in 1:N) {
    mu[n] += r_1_1[J_1[n]] * Z_1_1[n] + r_1_2[J_1[n]] * Z_1_2[n];
  }
  // priors including all constants
  target += student_t_lpdf(temp_Intercept | 3, 289, 56);
  target += student_t_lpdf(sigma | 3, 0, 56)
    - 1 * student_t_lccdf(0 | 3, 0, 56);
  target += student_t_lpdf(sd_1 | 3, 0, 56)
```

Hierarchical Shrinkage Priors

- Piironen and Vehtari (2017) derives the Finnish Horseshoe prior for regression coefficients where

$$
\begin{aligned}
\beta_j &\sim \mathcal{N}\left(0, \tau\widetilde{\lambda}_j\right) \\
\widetilde{\lambda}_j^2 &= \frac{c^2\lambda_j^2}{\frac{\sigma^2}{n s_j^2} + c^2 + \tau^2\lambda_j^2} \\
\lambda_j &\sim \text{Half-Cauchy}\left(0, 1\right) \\
\tau &\sim \text{Half-t}\left(v, 0, s \times \sigma\right) \\
c^2 &\sim \text{Inverse-Gamma}\left(\frac{d}{2}, \frac{d}{2}\right) \\
\sigma &\sim \,?
\end{aligned}
$$

- What would be declared in which block of a Stan program?
- Write a Stan program for linear regresison with K predictors that each have a Finnish Horseshoe prior

Conclusion

- Should use hierarchical modeling unless there is reason not to
- Hierarchical models are straightforward from a Bayesian perspective
- NUTS does a better job with hierarchical modeling that does Gibbs
- But the parameterization can make a big difference to NUTS