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 First Session

• 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
• Estimate a hierarchical model using \texttt{rstanarm::stan_glmer}
Hierarchical Data Generating Processes: Bowling

• How to model what person $i$ does on the $j$th bowling frame?

```r
x_1 <- sample(0:10, size = 1)
pins_left <- 10 - x_1
x_2 <- sample(0:pins_left, size = 1)
x_1 + x_2
## [1] 4
```

• All Stan does is draw from a conditional probability distribution
Hierarchical Data Generating Processes: Bowling

- How to model what person $i$ does on the $j$th bowling frame?

- You would need (at least) two probability distributions:
  1. Probability of knocking down $x_1 \in \{0, 1, \ldots, 10\}$ pins on the first roll
  2. Probability of knocking down $x_2 \in \{0, 1, \ldots, 10 - x_1\}$ pins on the second roll, given that $x_1$ pins were knocked down on the first roll

```r
x_1 <- sample(0:10, size = 1)
pins_left <- 10 - x_1
x_2 <- sample(0:pins_left, size = 1)
x_1 + x_2
## [1] 4
```

- All Stan does is draw from a conditional probability distribution
Hierarchical Data Generating Processes: IV

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

\[ \sigma_1 \sim \text{Exponential}(r_1) \]

**Priors:**

\[ \sigma_2 \sim \text{Exponential}(r_2) \]

\[ \rho \sim \text{Uniform}(-1, 1) \]

**Errors:**

\[
\begin{bmatrix}
\nu_i \\
\varepsilon_i
\end{bmatrix}
\sim \mathcal{N}_2 \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \rho \sigma_1 \sigma_2 \\ \rho \sigma_1 \sigma_2 & \sigma_2^2 \end{bmatrix} \right) \quad \forall i
\]

**Priors:**

\[
\begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\alpha_2
\end{bmatrix}
\sim \mathcal{N}_3(\mu_1, \Sigma_1)
\]

\[
\begin{bmatrix}
\beta_0 \\
\beta_1 \\
\beta_2
\end{bmatrix}
\sim \mathcal{N}_3(\mu_2, \Sigma_2)
\]

1st stage: 
\[ t_i \equiv \alpha_0 + \alpha_1 x_i + \alpha_2 z_i + \nu_i \quad \forall i \]

2nd stage: 
\[ y_i \equiv \beta_0 + \beta_1 x_i + \beta_2 t_i + \varepsilon_i \quad \forall i \]
Coefficients Depending on Other Coefficients

Write a simple Stan program where the coefficient on age is a linear function of the person’s income, which both affect the probability of voting in a logit model, starting with

```stan
data {
  int<lower=1> N;
  vector[N] age;
  vector[N] income;
  int<lower=0,upper=1> vote[N]; // outcome
}
parameters {
  vector[2] lambda; // intercept / slope for age’s effect
  vector[2] beta;  // intercept / slope for log-odds
}
model {
  target += bernoulli_logit_lpmf(vote | eta);
} // priors on lambda, beta, and sigma
```
Coefficients Depending on Other Coefficients

Write a simple Stan program where the coefficient on age is a linear function of the person’s income, which both affect the probability of voting in a logit model, starting with

```stan
data {
  int<lower=1> N;
  vector[N] age;
  vector[N] income;
  int<lower=0,upper=1> vote[N]; // outcome
}
parameters {
  vector[2] lambda; // intercept / slope for age’s effect
  vector[2] beta;  // intercept / slope for log-odds
}
model {
          + beta_age .* age;
  target += bernoulli_logit_lpmf(vote | eta);
} // priors on lambda, beta, and sigma
```
Relation to Interaction Terms in R

- If

\[ \eta_i = \beta_1 + \beta_2 \times \text{Income}_i + \beta_3 \times \text{Age}_i \]

\[ \beta_3 = \lambda_1 + \lambda_2 \times \text{Income}_i \]

then by substituting and distributing:

\[ \eta_i = \beta_1 + \beta_2 \times \text{Income}_i + (\lambda_1 + \lambda_2 \times \text{Income}_i) \times \text{Age}_i \]

\[ = \beta_1 + \beta_2 \times \text{Income}_i + \lambda_1 \times \text{Age}_i + \lambda_2 \times \text{Income}_i \times \text{Age}_i \]

and \(\beta_1, \beta_2, \lambda_1, \) and \(\lambda_2\) can be estimated (unregularized) via

\texttt{glm(vote ~ income + age + income:age, family = binomial)}

- Stan version is easier to interpret; R version is quick
- Many hierarchical models are just interactions w/ group indicators
Coefficients Depending on Other Coefficients Again

Write a Stan program where the coefficient on age is a noisy linear function of the person’s income with standard deviation $\sigma$, starting with

```stan
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 log-odds
}
model {
  target += bernoulli_logit_lpmf(vote | eta);
  target += normal_lpdf(noise | 0, 1);
}
```
Coefficients Depending on Other Coefficients Again

Write a Stan program where the coefficient on age is a noisy linear function of the person’s income with standard deviation $\sigma$, starting with

```stan
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 log-odds
}
model {
                        + sigma * noise; // non-centering
                  + beta_age .* age;
  target += bernoulli_logit_lpmf(vote | eta);
  target += normal_lpdf(noise | 0, 1);
} // priors on lambda, sigma, and beta
```
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

\[
\begin{align*}
\tau &\sim \text{Exponential} \left( r_\tau \right) \\
\alpha_j &\sim \mathcal{N} \left( 0, \tau \right) \forall j \\
\beta &\sim \mathcal{N} \left( \mu_\beta, \sigma_\beta \right) \\
\sigma_\varepsilon &\sim \text{Exponential} \left( r_\sigma \right) \\
\varepsilon_{ij} &\sim \mathcal{N} \left( 0, \sigma_\varepsilon \right) \\
y_{ij} &\equiv \alpha_j + \beta \times \text{class\_size}_{ij} + \varepsilon_{ij} \forall i, j
\end{align*}
\]
functions {
vector cluster_DGP_rng(int J, int[] N, vector class_size,
    real r_tau, real r_sigma,
    real mu_beta, real sigma_beta) {

    \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 j + \varepsilon_{ij} \ \forall i, j
}
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];
      real epsilon = normal_rng(0, sigma)
      y[pos] = mu + epsilon;
      pos += 1;
    }
  }
  return y;
}
Exposing Stan Functions to R

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

```r
rstan::expose_stan_functions("schools.stan")
```

```r
args(cluster_DGP_rng)
```

```r
## function (J, N, class_size, r_tau, r_sigma, mu_beta, sigma_beta,
## seed = 0L)
## NULL
```

• Now you can call the `cluster_DGP_rng` function with those arguments and get back one vector of prior predictions

• Doing so repeatedly is a good way to judge whether your priors make sense
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 $j$th group.

• Best way to think about such structures:
  • There is a likelihood contribution for the $j$th 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.
### Table 2 from the **lme4** Vignette (frequentist)

<table>
<thead>
<tr>
<th>Formula</th>
<th>Alternative</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1</td>
<td>g)</td>
<td>1 + (1</td>
</tr>
<tr>
<td>0 + offset(o) + (1</td>
<td>g)</td>
<td>-1 + offset(o) + (1</td>
</tr>
<tr>
<td>(1</td>
<td>g1/g2)</td>
<td>(1</td>
</tr>
<tr>
<td>(1</td>
<td>g1) + (1</td>
<td>g2)</td>
</tr>
<tr>
<td>x + (x</td>
<td>g)</td>
<td>1 + x + (1 + x</td>
</tr>
<tr>
<td>x + (x</td>
<td></td>
<td>g)</td>
</tr>
</tbody>
</table>

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 \textit{a priori} known offsets as x

- The **lme4** parser converts statements like $x + (x | g)$ to a sparse matrix $Z$ that interacts (some columns of) $X$ with group-specific dummy variables, one for each level of $g$. 

---

Ben Goodrich

Hierarchical Models I

StanCon Helsinki
Restatement of the Hierarchical Linear Model

- Generally, both intercepts and slopes can vary across groups
- Let $\beta_j = \beta + b_j$ and $b^\top = [b_1^\top \ b_2^\top \ \cdots \ b_J^\top]$. Then:

  $\begin{align*}
  y &= \underbrace{X\beta}_{\text{Frequentist } \mu} + Zb + \varepsilon = \underbrace{X\beta + ZL(\theta)u\sigma}_{\text{Frequentist error}} + \underbrace{\varepsilon}_{\text{Bayesian error}} \\
  \end{align*}$

  $\begin{align*}
  \text{Bayesian } \mu &= X\beta \\
  \text{Frequentist } \mu &= X\beta + ZL(\theta)u\sigma \\
  \Rightarrow \varepsilon &= \text{Bayesian error} \\
  \end{align*}$

  where $L(\theta)$ is a Cholesky factor of $\text{cov}(b) = \Sigma(\theta) = L(\theta)L(\theta)^\top$

- Bayesians: $b \sim \mathcal{N}(0, \Sigma(\theta))$ and $y \sim \mathcal{N}(X\beta + Zb, \sigma^2 I)$

- For frequentists, each $b_j$ is not a fixed “parameter” but rather a random variable that is part of the error term that gets integrated out to choose $\hat{\beta}, \hat{\sigma}$, and $\Sigma(\theta)$ to maximize a multivariate normal likelihood with mean $X\beta$ and covariance matrix $\sigma^2 Z\Sigma(\theta)Z^\top$

- Technically, $\hat{b}_j$ is not “estimated” but rather “predicted” from the residuals $\varepsilon = y - X\hat{\beta}$ by subsequently regressing $\varepsilon$ on $Z$
Bayesian Implementations with \texttt{lmee4} / \texttt{mgcv} Syntax

- The \texttt{rstanarm} and \texttt{brms} packages accept \texttt{lmee4} syntax
- Both also permit the same $s(\ldots)$ syntax as \texttt{mgcv} to use smooth, non-linear functions of parameters
- Add arguments for the priors on $\alpha$, $\beta$, $\sigma$, etc.
- Try \texttt{rstanarm} and / or \texttt{brms} first to make sure your data are amenable to a hierarchical model
Tadpole Example from McElreath, chapter 12

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'))

dim(as.matrix(post)) # raw draws from posterior distribution
## [1] 4000  51
Results of Tadpole Example from McElreath

```r
## stan_glmer
## family: binomial [logit]
## formula: cbind(surv, density - surv) ~ size + (1 | tank)
## observations: 48
##
## Median MAD_SD
## (Intercept)  1.2   0.4
## sizesmall   0.4   0.5
##
## Error terms:
## Groups Name Std.Dev.
## tank (Intercept)  1.7
## Num. levels: tank 48
##
## Sample avg. posterior predictive distribution of y:
## Median MAD_SD
## mean_PPD      16.3 0.4
##
## For info on the priors used see help('prior_summary.stanreg').
```
More Results of Tadpole Example from McElreath

```r
fixef(post)
## (Intercept)  sizesmall
## 1.1524158  0.4427099

NROW(ranef(post)$tank)
## [1] 48

head(cbind(coef(post)$tank[,1],
          fixef(post)[1] + ranef(post)$tank))
##     coef(post)$tank[, 1] (Intercept)
## 1  2.018290     2.018290
## 2  2.914472     2.914472
## 3  0.946066     0.946066
## 4  2.899101     2.899101
## 5  1.758918     1.758918
## 6  1.731015     1.731015
```