36-463/663: Multilevel & Hierarchical Models
Spring 2022
HW06 – Due Tue Mar 29, 11:59pm

- Please turn the homework in, as a single pdf, online in GradeScope using the link provided on the Assignment page on canvas.cmu.edu. Upload *one* file per person.
- There are 2 main exercises, each with "parts"...
- These are not "interesting" exercises by themselves. Instead they are more like "finger exercises" to practice writing, debugging, and examining the output of, `stan()` programs.
  - Feel free to borrow ideas liberally from lecture notes, pdfs and R files shared in class, and/or from resources you find on the www, etc.

Exercises

1. **Chicks.** The `ChickWeight` data frame is included with R, and you can access it with `data(ChickWeight)`. It has 578 rows and 4 columns from an experiment on the effect of diet on early growth of chicks. The variables (Columns) are:
   - weight: a numeric vector giving the body weight of the chick (gm).
   - Time: a numeric vector giving the number of days since birth when the measurement was made.
   - Chick: an ordered factor with levels $18 < \ldots < 48$ giving a unique identifier for the chick. The ordering of the levels groups chicks on the same diet together and orders them according to their final weight (lightest to heaviest) within diet.
   - Diet: a factor with levels 1, ..., 4 indicating which experimental diet the chick received.

   For most of this exercise we will be working with the model

$$\begin{aligned}
\log(\text{weight}_i) &= \alpha_{0j[i]} + \alpha_{1j[i]} \cdot \text{Time}_i + \epsilon_i \\
\alpha_{0j} &= \beta_0 + \eta_{0j} \\
\alpha_{1j} &= \beta_1 + \eta_{1j} \\
\epsilon_i &\overset{iid}{\sim} N(0, \sigma^2) \\
\eta_{0j} &\overset{iid}{\sim} N(0, \tau_0^2) \\
\eta_{1j} &\overset{iid}{\sim} N(0, \tau_1^2)\,,
\end{aligned}$$

   where i is the observation number and j is the Chick number[1]. You will have to convert `Chick` from a factor variable to a numerical variable, to work with `stan()`.

   (a) In the same folder as this assignment sheet there is a file `chicks.stan` *with most of the program already written, but the model section left blank.* Fill in the blank model section[2] and get the program to run using

[1] Also note that in this model we are just lettng $\rho_{\eta_0,\eta_1} = 0$, to make the `stan()` coding less complicated.

[2] You should not have to change any of the code already in the file; just add to it. However if it is easier for you to change some of the code in the file, that is fine too.

`library(rstan)`, taking at least 500 MCMC steps per chain[3], and accepting the other `stan()` defaults, so that you end up with at least 1000 "good" MCMC samples from the four Markov chains produced. Turn in:

- The completed stan program `chicks.stan`.
- A printout of the fitted stan object but including only the lines for β_0, β_1, τ_0, τ_1, and σ.

(You are free to use `lmer()` to estimate the corresponding multi-level model, to make sure you are on the right track, but please don't turn in any `lmer()` code or output for this part of the exercise.)

(b) Use `shinystan` or `bayesplot` to generate and turn in the following plots, neatly organized:

- For β_0: (i) Histogram of the MCMC draws; (ii) Autocorrelation plot for at least one of the four Markov chains; (iii) trace plot of all four Markov chains on the same graph, with different colors to indicate the different Markov chains.
- For β_1: the same three things.
- For τ_0: the same three things.
- For τ_1: the same three things.
- For σ: the same three things.

(c) Set `y <- log(ChickWeight$weight)`, and use `shinystan` or `bayesplot` to produce graphical posterior predictive checks (ppc's) for

- `T(y) = mean(y)`
- `T(y) = sd(y)`
- At least one other test statistic `T(y)` that produces an interesting result.

(d) Now fit the corresponding `lmer()` model (the parameter estimates should be almost identical to your estimates from the `stan()` model). Turn in

- The plot of the conditional residuals for the `lmer()` model. If the model fits well, these should be approximately normally distributied.
- The plot of the "uniformized" conditional residuals[4] for the `lmer()` model, using `library(DHARMa)`. If the model fits, these should be approximately uniformly distributed.

Write a sentence or two identifying and commenting on any suggested improvements to the model, or any other useful information, that you can see in these plots.

2. **Toenails.** The `toenail` data frame comes from the `faraway` library, and you can access it by

```
install.packages("faraway")
library(faraway)
data(toenail)
```

There are 1908 observations from a study comparing two oral treatments for toenail infection. Patients were evaluated for the degree of separation of the nail. Patients were randomized into two treatments and were followed over seven visits—four in the first year and yearly thereafter. The patients have not been treated prior to the first visit so this should be regarded as the baseline. The variables (columns) are:

- ID: ID of patient
- outcome: 0=none or mild seperation, 1=moderate or severe

[3] If you need to take more steps to get convergence to the stationary distribution, that is fine.

[4] We could build these by hand from the `stan()` model, but `DHARMa` is faster and less prone to user errors.

- treatment: the treatment A=0 or B=1
- month: time of the visit (not exactly monthly intervals hence not round numbers)
- visit: the number of the visit

For most of this exercise we will be working with the model

$$\begin{aligned}
\text{outcome}_i &\sim \text{Bernoulli}(p_i) \\
\text{logit}(p_i) &= \alpha_{0j[i]} + \alpha_{1j[i]} \cdot \text{month}_i + \beta_2 \cdot \text{treatment}_{j[i]} + \epsilon_i \\
\alpha_{0j} &= \beta_0 + \eta_{0j} \\
\alpha_{1j} &= \beta_1 + \eta_{1j} \\
\eta_{0j} &\overset{iid}{\sim} N(0, \tau_0^2) \\
\eta_{1j} &\overset{iid}{\sim} N(0, \tau_1^2)
\end{aligned}$$

where i is the observation number and j is the patient ID number[5]. Note that there are some skips in the ID number sequence in the data set; it would be simplest for working with `stan()` to convert these ID numbers to successive integers with no skips. (Note that σ is missing from this model—why?)

(a) In the same folder as this assignment sheet there is a file `toenails.stan` *with the parameters and transformed parameters sections left blank*. Fill in the blank sections[6] and get the program to run using `library(rstan)`, taking at least 500 MCMC steps per chain[7], and accepting the other `stan()` defaults, so that you end up with at least 1000 "good" MCMC samples from the four Markov chains produced. Turn in:

- The completed stan program `toenails.stan`.
- A printout of the fitted stan object but including only the lines for $\beta_0, \beta_1, \beta_2, \tau_0$, and τ_1.

Some notes:

- You are free to use `glmer()` to estimate the corresponding multi-level model, to make sure you are on the right track, but please don't turn in any `glmer()` code or output for this part of the exercise. Note that, even if you have done everything right the magnitudes of the `glmer()` estimates may not agree very well with the magnitudes of the `stan()` estimates; see part (d) below also.
- You will struggle to get a "great" run from `stan()` on this model. If you can get a run where most or all of the difficulties are in `yrep`, that will be good enough: the various convergence and stability diagnostics like `Rhat` don't really apply to `yrep` since it is simulated from the posterior predictive distribution; moreover the discreteness of `yrep` makes n_{eff} less meaningful.

(b) Use `shinystan` or `bayesplot` to generate and turn in the following plots, neatly organized:

- For β_0: (i) Histogram of the MCMC draws; (ii) Autocorrelation plot for at least one of the four Markov chains; (iii) trace plot of all four Markov chains on the same graph, with different colors to indicate the different Markov chains.
- For β_1: the same three things.
- For β_2: the same three things.
- For τ_0: the same three things.

[5] Also note that in this model we are just lettng $\rho_{\eta_0,\eta_1} = 0$, to make the `stan()` coding less complicated.

[6] You should not have to change any of the code already in the file; just add to it. However if it is easier for you to change some of the code in the file, that is fine too.

[7] If you need to take more steps to get convergence to the stationary distribution, that is fine.

- For τ_1: the same three things.

(c) Set `y <- toenail$outcome`, and use `shinystan` or `bayesplot` to produce graphical posterior predictive checks (ppc's) for

- `T(y) = mean(y)`
- `T(y) = sd(y)`
- At least one other test statistic `T(y)` that produces an interesting result.

(d) Now fit the corresponding `glmer()` model (the parameter estimates should be almost identical to your estimates from the `stan()` model). When I did this, I thought that fixed effect estimates from `glmer()` were much less plausible than the fixed effect estimates from `stan()`, which made me trust the `stan()` results more. Do you agree? Why or why not?

(e) Turn in

- The plot of the conditional residuals for the `glmer()` model. If the model fits well, these should be approximately normally distributied.
- The plot of the "uniformized" conditional residuals[8] for the `lmer()` model, using `library(DHARMa)`. If the model fits, these should be approximately uniformly distributed.

Write a sentence or two identifying and commenting on any suggested improvements to the model, or any other useful information, that you can see in these plots (or if you think some or all of the plots are useless, explain that as well).

[8] We could build these by hand from the `stan()` model, and we probably should since the `stan()` results seem more plausible than the `glmer()` results, but `DHARMa` is faster and less prone to user errors; it will be fine for the purpose of just getting used to using and interpreting `DHARMa` stuff.