Problem Set #3:

1. Partial pooling in hierarchical models. The 8 Schools model (considered in class), can be expressed as follows:
   1. yi ~ Normal(mui, sigmai)
   2. mui ~ Normal(nu, tau)

In the basic setup, the data are {yi,sigmai} (i.e. the treatment effects and variances are considered as observed quantities for each school), and the parameters to be estimated are mui (the partially pooled estimates of treatment for each school), nu (the overall average treatment effect), and tau (the uncertainty of the global effect). The nu and tau parameters are given prior distributions, which if left unspecified in Stan amount to an “uninformative” or uniform prior distribution.

1. For the first part of this problem, rather than estimate “tau”, fix it to a constant value and then run the model. Repeat this for tau = c(0.1, 0.5, 1, 5, 10, 50, 100). For each model run, save the estimates of the mui, and then plot their posterior mean as a function of tau, compared to the posterior mean of nu (the average treatment effect). Your plots should thus have 8 curves (one for each school), with tau on the x-axis.
2. In this part of the problem, you will again be treating tau as a parameter to be estimated rather than fixed to a constant value. Specify 3 different half-Normal priors as follows: 1) make sure that you declare real<lower = 0> tau; in the parameters block, and then 2) in model block, write down tau ~ normal(0,scale) for the following choices of scale = c(1,5,10,50). Once again, compare how the mu change by plotting 8 different curves with the prior scale on the x-axis.

**Teaching note**: we normally say that the posterior “averages over” the prior. Comparing the two kinds of answers you get above, is a concrete way of understanding what this means in practice. In this way, you can think about your specification of a prior distribution as an alternative to committing to one fixed value. Meanwhile, the hierarchical model itself is a way of compromising between no pooling and complete pooling, and will almost always outperform either alternative. In the hierarchical model where you *estimate* tau, you are letting the model “learn” the among-group variance, but “tuning” that learning process with the prior (for my machine learning students ☺).

1. Fitting a logistic state space model. Using your homework from last problem set, and the material covered in class, set up a simulation of a logistic growth process, where you are fixing the carrying capacity parameter K to some standard value (100 is an easy choice), but letting the growth rate parameter alpha vary c(0.1,0.5,1).
   1. First create a vector of the true values of your latent variable ‘z’ assuming no process error for the 3 different values of alpha. Plot out the trajectories to make sure that you are picking up a nice range of sigmoidal behavior. (Note, you will need to choose a reasonable initial condition for each).
   2. Simulate fake “observed” values by simulating random noise for each observation of z using z + rnorm(N,0,sigma), for some reasonable choice of sigma. Plot your new ‘noisy’ trajectories.
   3. Flesh out the model discussed in class in Stan code, and try to fit your fake data to it. Use the “extract(stan\_model, pars = c(alpha,sigma))” to extract the MCMC samples for the alpha and sigma parameters. Plot these as histograms, with a solid vertical red line indicating the “true” value used to simulate the data. Refit 3 times (one for each different alpha).
   4. **Challenge**: Repeat the simulation and model fitting only adding process error to your simulation of the latent variable ‘z’. For the time being, keep your process error variance modestly low. Also, rather than one observation at each time point, draw 3-5. You will need to pass in your data y as a matrix to do this. One way is to declare ‘real<lower = 0> y[N,M];’ in the data block, where you have already placed ‘int<lower = 0> N;’ and ‘int<lower = 0> M;’ above it to define then rows and columns. Once again, compare your parameter estimates (in terms of histograms of posterior draws) to the ‘true’ simulated values.

**Teaching Note:** This problem takes you right up to some solidly intermediate hierarchical modeling. Take your time with it, and feel free to email me questions when/if you get stuck. As you develop models, this procedure of first doing simulations, and then fitting to simulated values is your first line of defense against model mis-specification and error. We will build off of this in the next class using a physics motivated example. If you think of model building as expressing a probabilistic story of how your data were generated, this step amounts to making sure that your methods are capable of returning correct answers when you get to make the story yourself. It will act as your trustworthy friend and guide when you are unsure of yourself and your models.