**STAT 578 – Advanced Bayesian Modeling – Spring 2020**

# Assignment 3

1. (a) [2 pts] R script `FlintGibbs.R` implements a Gibbs sampler for the partially conjugate Flint data model. It is very similar to the example R code in Lesson 7.2: Gibbs Sampling. Use the script to simulate from the posterior for $\mu$ and $\sigma^2$. Then use R function `acf` to produce an autocorrelation plot for the successive $\mu$ variates and an autocorrelation plot for the successive $\sigma^2$ variates.

   (b) R script `FlintMetropolis.R` implements a Metropolis sampler for the same Flint data model. It is very similar to the example R code in Lesson 7.3: Metropolis and Metropolis-Hastings.

      (i) [1 pt] Experiment with different settings for the proposal variance `rho` (by uncommenting its line and changing the value). Find a value of `rho` that gives an overall (average) acceptance rate of about 0.35.[1] What value of `rho` did you find?

      [Warning: Using a value of `rho` that is too large may cause the R code to produce an error, due to numerical problems. Start with smaller values of `rho`.]

      (ii) [2 pts] With the value of `rho` that you found, use the script to simulate from the posterior for $\mu$ and $\sigma^2$. Then use R function `acf` to produce an autocorrelation plot for the successive $\mu$ variates and an autocorrelation plot for the successive $\sigma^2$ variates.

   (c) [1 pt] Compare the autocorrelation plots from the previous two parts. Which method exhibited faster mixing: the Gibbs sampler or the Metropolis sampler?

2. Use the 2016 US presidential polls data in `polls2016.txt` to answer the following, running all parts using JAGS and R (`rjags`). Remember that you will have to create a variable `sigma` in R to represent the standard deviation of the polls, defined to be half of the margin of error. Refer to Lesson 4.2: Normal Hierarchical Model in R/JAGS.

   (a) Use the model in `polls20161.bug` for the following:

      (i) [2 pts] Create an initialization list (in R) supporting 4 chains, with a different initialization for each chain. Set initial values for `mu` to $\pm 100$ and values for `tau` to 100 or 0.01. Then use `jags.model` to create the JAGS model R object with these initializations. List all of the R code you used.

      (ii) [2 pts] Perform a burn-in of 2500 iterations, then monitor the `mu` and `tau` nodes for 5000 iterations (for each chain). List all of the R code you used.

      (iii) [3 pts] For the iterations you monitored, produce trace plots of `mu` and `tau`. Do there appear to be any convergence problems? Display the plots and R code that produced them.

      (iv) [2 pts] For the iterations you monitored, compute Gelman-Rubin statistics (potential scale reduction factors) for `mu` and `tau`. Do there appear to be any convergence problems? Show your R code and its output.

[1] This is approximately the "optimal" acceptance rate for a two-dimensional parameter. See Roberts, G. O., & Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, *16*, 351–367.

(v) [2 pts] For the iterations you monitored, display autocorrelation plots for `mu` and `tau` for one of the chains. (Hint: For example, to reference the first chain of an `mcmc.list` object named `x`, use `x[[1]]`.) Comment on the apparent speed of mixing.

(vi) [2 pts] For the iterations you monitored, compute effective sample sizes (over all chains) for `mu` and `tau`. Would they be considered adequate? Show your R code and its output.

(b) Now consider a new model that uses an almost flat prior for `tau` on the *log* scale, as follows: Create a new JAGS model by modifying `polls20161.bug` to eliminate the current prior for `tau`, create a new parameter `logtau` with a $\mathrm{U}(-100, 100)$ prior distribution, and define `tau` to be equal to `exp(logtau)`.

(i) [2 pts] Display all of the code for your new JAGS model.

(ii) [2 pts] Create an initialization list (in R) supporting 4 chains, with a different initialization for each chain. Set initial values for `mu` to $\pm 100$ and values for `logtau` to $\log 100$ or $\log 0.01$. Then use `jags.model` to create the JAGS model R object with these initializations. List all of the R code you used.

(iii) [2 pts] Perform a burn-in of 2500 iterations, then monitor the `mu` and `tau` nodes for 5000 iterations (for each chain). List all of the R code you used.

(iv) [3 pts] For the iterations you monitored, produce trace plots of `mu` and `tau`. Do there appear to be any convergence problems? Display the plots and R code that produced them.

(v) [2 pts] For the iterations you monitored, compute Gelman-Rubin statistics (potential scale reduction factors) for `mu` and `tau`. Do there appear to be any convergence problems? Show your R code and its output.

(vi) [2 pts] For the iterations you monitored, display autocorrelation plots for `mu` and `tau` for one of the chains.[2] Comment on the apparent speed of mixing.

(vii) [2 pts] What is wrong with this model that could explain any problems you noted? (Hint: What would happen if you used an improper flat prior on $\log \tau$?)

Total: 34 pts

[2] Ordinarily, autocorrelation plots are not used for chains that have not converged, but you are asked to produce them here even if there was no convergence.