ST440/540 Applied Bayesian Analysis
Lab activity for 2/24/2025

Announcements

- Homework 4 due Friday
- Quiz 7 due Friday
- There is a group formation survey due next Friday on Moodle

For 540 students, I posted the project description the course assignments page.
https://st540.wordpress.ncsu.edu/assignments/
The next step is an abstract with a brief project description due in a few weeks.

For 440 students, your final exam will be a group analysis of a problem I assign you. Please still fill out the group formation survey! However, you do not need to start thinking of a research topic because I will send the project after the second mid-term.

A. STUDENT QUESTIONS

(1) Should we know what rejection sampling is? The term is new to me.

We won't use it in this class. You probably have enough algorithms in your head at the moment, but if you'd like an explanation let me know.

(2) How much unique/custom code do you expect us to write to perform these sampling types?

After the homework assignment this week you won't have to write any custom code. The idea is that you've done it a few times so you will understand what's happening inside JAGS.

(3) I'm still kind of foggy on the way Metropolis sampling works. Here is how I'm interpreting it at a very oversimplified level (please let me know if my understanding is accurate): you come up with a parameter estimate, generate a new estimate based on that previous estimate, calculate the ratio between the two estimates, "accept" the new one based on the ratio, and then use it to come up with a new estimate (with the acceptance probability saved somewhere?). And then you proceed thousands of times to draw out the posterior, and that's why you don't just stop when you come up with a peak in the distribution. If I'm correct, where exactly does the power of this sampling come from? How can we rely on the idea that this algorithm will actually draw out the correct posterior?

This is a fair description of the algorithm. The "power" comes from the fact that you can sample from *any* distribution as long as you can write the likelihood and prior. This is actually really hard, it just looks easy for simple cases like rnorm and rpois.

The proof that the algorithm converges to the correct posterior is beyond the scope of the class. You need to use a bunch of terms from stochastic process theory like ergodicity and reversibility. For now, we will take the probabilists at their word 😊

(4) Can you show the derivation for the Posterior Distribution for an Inverse Gamma Likelihood and Gamma Prior. I am getting Gamma (a,b) it must be wrong because it is not even close to what is shown in the Conjugate Prior Table on Wikipedia.

Sure. If Y|b ~ InvGamma(a,b) with a fixed and b ~ Gamma(c,d), then the posterior for b is

$$p(b|Y)\alpha\, f(Y|b)\pi(b)\;\alpha\; [b^{a}\exp\left(-\frac{b}{Y}\right)][b^{c-1}\exp(-bd)] = \; [b^{C-1}\exp(-bD)]$$

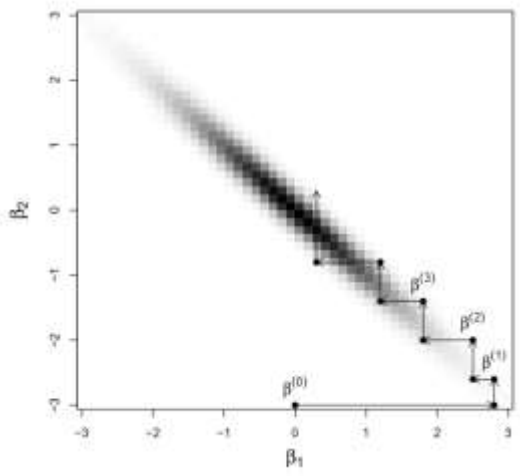
so b|Y ~ Gamma(C,D) for C=a+c and D = 1/Y+d.

(5) Given a sufficient number of runs, will a poorly-tuned candidate standard deviation cause the Metropolis algorithm not to converge?

Theoretically it will converge, practically it will not. In other words, you would have to run the chain for too long to be useful.

(6) I was really lost on slide 25 of section 3.2, the graph depicting "Posterior correlation leads to slow convergence" - I wasn't following what exactly this graph was showing, or what your explanation of it meant. Could you please revisit it and describe what it is depicting? Or direct me to some more reading on it that I could read directly?

The most basic MCMC algorithms update parameters one at a time from their joint distribution. The shading in the plot below is a made-up posterior distribution for (beta1, beta2) that shows negative correlation. Updating the parameters one at a time from their full conditional distributions limits the available moves to the horizonal and vertical lines/arrows. To stay in the shaded region where the posterior is non-zero using only these horizonal and vertical moves means you can only take small steps. "Small steps" is another way to say "slow convergence". If you could move diagonally you could move through the shaded region more quickly. This is what blocked sampling does because you updates the parameters simultaneously instead of sequentially.



(6) Could you review the exam for the maximum speed wind analysis?

We went over it last class on the solution is posted here

http://st540.wordpress.ncsu.edu/files/2025/02/E1_sol.pdf

Of course, this is just one solution, others fit Pareto or log-normal distributions and came to similar conclusions. I don't want to repeat myself too much, so if you have any specific questions it's better to email me.

(7) I just want to make sure that I understand how R functions in metropolis sampling. From my understanding, R is a probability that will indicate whether your candidate has a greater posterior probability then the previous candidate given the rest of the data. If so, you always move in the direction of the new candidate, but if not you use this probability to determine whether to move in the direction of the new candidate or stay where you are. Is this correct? If so is the smaller probability used for decreasing directions so that your posterior probability does not stretch across too wide of a range?

You got it. Small R means the candidate is a much worst fit to the posterior than the previous value, so the algorithm is less likely to move in this direction in order to stay in the range of values with high posterior probability.

(8) How do you know/decide when the burnout period should end and that you are actually drawing from the posterior distribution?

This the topic of the second video this week. Usually, you just prespecify a burn-in period that is, say, 20% of the total chain length and then just visually check that after this point the chains have stabilized.

(9) When we get JAGS, will we still write the models that we talked about this week? And, will JAGS make anything we write faster?

JAGS will write the code for you behind the scenes. It will make writing the code faster, although the actually time to execute the code is usually a bit slower.

(10) Among the different types of MCMC sampling (Gibbs, Metropolis. Metropolis-Hastings, Hamiltonian), are there any similarities or difference in the criteria and diagnostics for analyzing convergence?

Most the convergence diagnostics are the same for all MCMC sampling algorithms. Of course, Gibbs doesn't have an acceptance probability to monitor, but the other methods we will discuss in the next video are the same.

(11) What are some good examples of pitfalls you can walk into when doing MH sampling?

Usually, the hardest problems stem from posterior correlation between parameters. This can be hard to diagnose because the only sure-fire way to study the posterior is MH sampling. For this reason, it's good to start with simple models, like say fixing some parameters, and slowly allow the model to be more complex until it's either the model you want to fit and all is good or the MCMC breaks in which case you should consider a simpler model.

B. HOMEWORK AND CLASS PARTICIPATION SOLUTIONS

Q6: The optimal acceptance rate for Metropolis sampling is 20-40%.

(a) Why is a lower acceptance rate suboptimal?

Because the chain gets stuck and long chains are needed to explore the posterior.

(b) Why is a higher acceptance rate suboptimal?

Because this usually means the candidate is only a small deviation from the previous value, and so the steps are incremental and long chains are needed to explore the posterior.

C. DISCUSSION QUESTIONS

(1) Recall that Bayes' rule is $p(\theta|Y) = f(Y|\theta)\pi(\theta)/m(Y)$. Explain why we never need to compute m(Y) to perform Metropolis sampling. Your answer must include a formula!

Say θ_1 is the candidate and θ_0 is the previous value. The metropolis ratio is

$$R = \frac{p(\theta_1|Y)}{p(\theta_0|Y)} = \frac{f(Y|\theta_1)\pi(\theta_1)/m(Y)}{f(Y|\theta_0)\pi(\theta_0)/m(Y)} = \frac{f(Y|\theta_1)\pi(\theta_1)}{f(Y|\theta_0)\pi(\theta_0)}$$

and the constant m(Y) cancels so we never need to compute it.

(2) Assume the model $Y|\ \theta \sim$ Gamma(θ,1) and prior $\theta \sim$ Uniform(0,10). This is not a conjugate prior and so you will use Metropolis-Hastings sampling.

(a) What is a reasonable candidate distribution for θ?

Normal(θ_0, c^2) is fine, although other distributions with support (0,10) might be more efficient.

(b) Give a formula for the acceptance probability (preferably in R code)

$$R = \frac{dgamma(Y, \theta_1, 1) * dunif(\theta_1, 0,10)}{dgamma(Y, \theta_0, 1) * dunif(\theta_0, 0,10)}$$

(c) What would you do if a candidate was outside the prior range (0,10)?

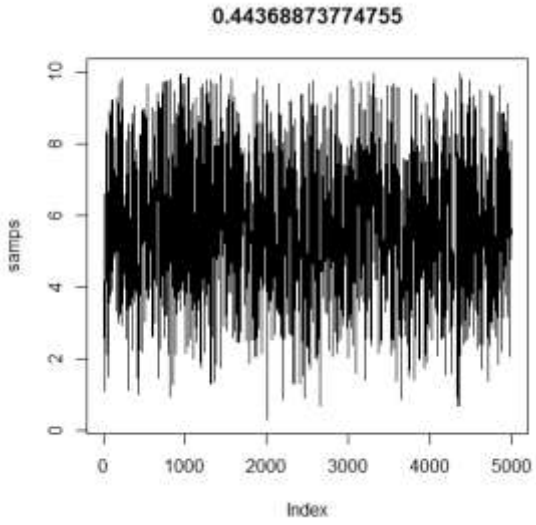
The prior PDF $dunif(\theta_1, 0,10)$ is zero and so the candidate is automatically rejected.

(d) How would you tune the candidate distribution? Be specific.

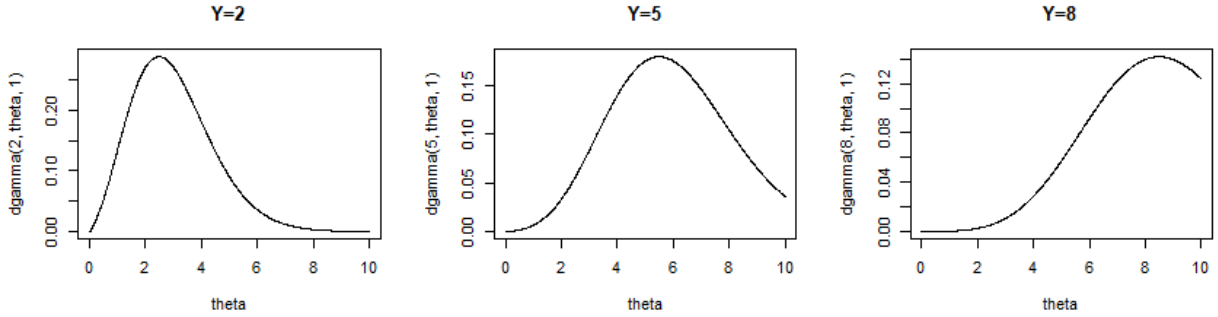
I would pick c until the acceptance probability is around 0.4.

Here is code if you're interested

```
Y      <- 5    # Data (I just picked something to illustrate the code)
can_sd <- 5    # Tuning parameter
iters  <- 5000 # Number of iters
theta  <- 1    # Initial value
samps  <- rep(theta,iters)
for(iter in 1:iters){
  can <- rnorm(1,theta,can_sd)
  if(can>0 & can<10){
     R   <- (dgamma(Y,can,1)*dunif(can,0,10))/
            (dgamma(Y,theta,1)*dunif(theta,0,10))
     if(runif(1)<R){theta <- can}
  }
  samps[iter] <- theta
}
acc_prob <- mean(samps[2:iters]!=samps[2:iters -1])
plot(samps,type="l",main=acc_prob)
```



These plots might help you visualize the problem:



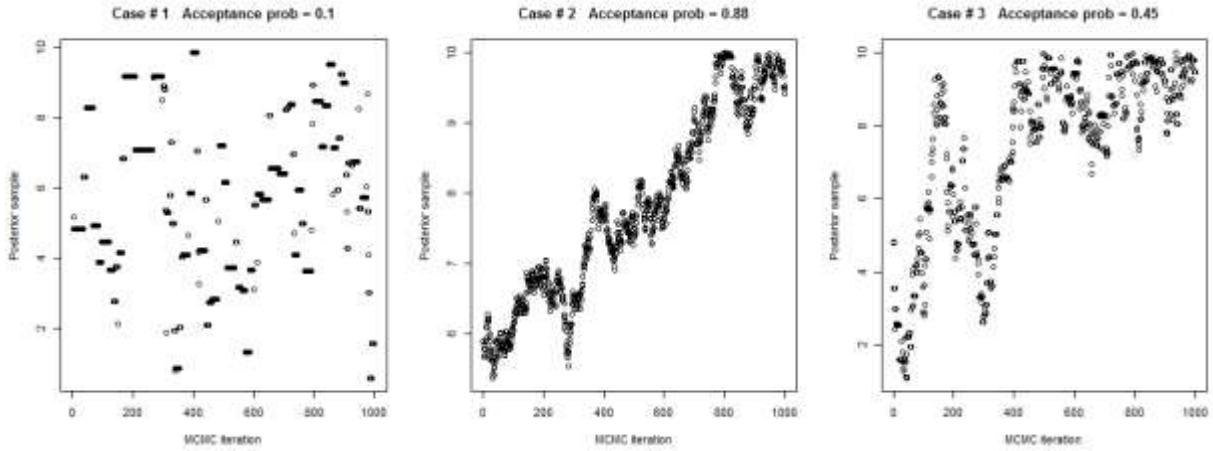

```
par(mfrow=c(1,3))
theta <- seq(0,10,.01)
plot(theta,dgamma(2,theta,1),type="l",main="Y=2")
plot(theta,dgamma(5,theta,1),type="l",main="Y=5")
plot(theta,dgamma(8,theta,1),type="l",main="Y=8")
```

(3) Referring to the model in (2), assume that we used a Gaussian candidate distribution with mean set to the previous value of θ and standard deviation c. The chains are run for 1000 iterations. For each of these plots, how would you modify the value c?

Case #1 – Lower c because acceptance is too low (maybe try c = c*0.8)

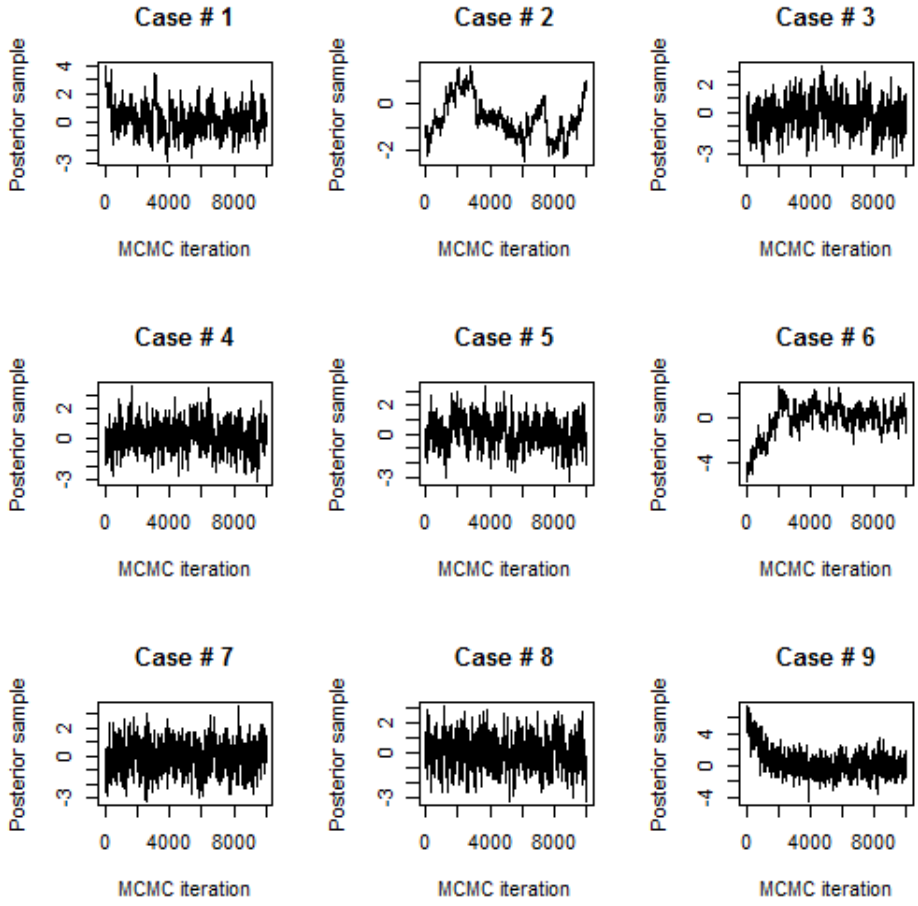
Case #2 – Increase c because acceptance is too high (maybe try c = c*1.2)

Case #3 – Looks pretty good, run it longer



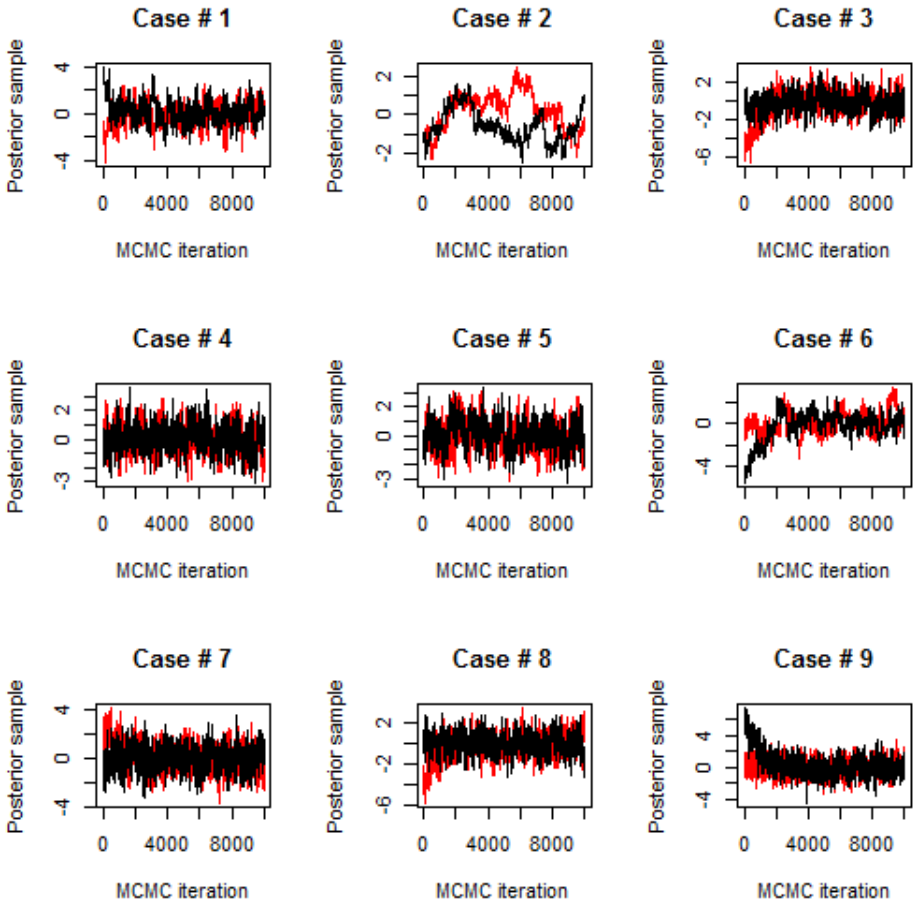
(4) For each trace plot below, at which iteration would you say the chain has converged?

Case #1 - 1000 Case #2 -10K Case #3 - 5K? Case #4 - 1K? Case #5 – 1K?
Case #6 -5K Case #7 - 1 Case #8 - 1 Case #9 – 5K



(5) Now instead of a single chain, two separate chains (one in red, one in black) are run with different starting values. For each case, select an iteration number T so that all samples after T from both chains can be kept and comment on how using multiple chains helped in this decision.

Case #1 - 1000 Case #2 -10K+ Case #3 - 5K Case #4 - 1 Case #5 – 1
Case #6 -5K Case #7 - 1 Case #8 - 1K Case #9 – 1K



(6) Consider the model Y|θ,b ~ Binomial(n,θ), θ|b ~ Beta(b,1-b) and b ~ Uniform(0,1).

(a) Specify initial values of θ and b.

See step 0 below

(b) What is the full conditional distribution of θ?

See step 1 below

(c) The full conditional distribution of b does not have a nice form and therefore can't be updated using Gibbs sampling. Sketch a Metropolis-within-Gibbs sampler for the joint posterior of (θ,b).

(0) Set theta = ½ and b = ½ (or theta = b= Y/n)

(1) Update theta given b as theta|b,Y ~ Beta(Y+b,n-Y+1-b)

(2) Update b given theta as

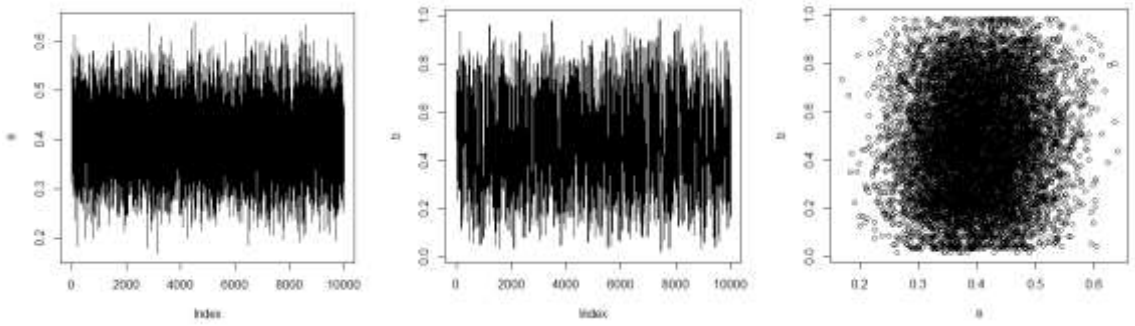
- Propose b_new |b_old ~ beta with mean b_old

Repeat steps (1) and (2) S times.

Here are code and results

```
n      <- 50
Y      <- 20
iters  <- 10000
tuning <- 1       # MH tuning parameter (see plots at the end)

theta  <- 0.5 # Initial values
b      <- 0.5

samps           <- matrix(0,iters,2)
samps[1,]       <- c(theta,b)
colnames(samps) <- c("theta","b")

for(iter in 1:iters){

  # Gibbs for theta
  theta <- rbeta(1,Y+b,n-Y+1-b)

  # MH for b
  can <- rbeta(1,tuning*b,tuning*(1-b))
  R1  <- dbeta(theta,can,1-can)*
         dunif(can,0,1)*
         dbeta(b,tuning*can,tuning*(1-can))
  R2  <- dbeta(theta,b,1-b)*
         dunif(b,0,1)*
         dbeta(can,tuning*b,tuning*(1-b))
  R   <- R1/R2
  if(runif(1)<R){b<-can}

  samps[iter,] <- c(theta,b)
}
```

```
acc_prob <- colMeans(samps[2:iters,]!=samps[2:iters -1, ])
> acc_prob
    theta         b
1.0000000 0.3163316
>
> colMeans(samps)
    theta         b
0.4004075 0.4765602
>
> apply(samps,2,quantile,c(0.025,0.975))
          theta          b
2.5%  0.2698998 0.07935664
97.5% 0.5384169 0.89343295

plot(samps[,1],ylab=expression(theta),type="l")
plot(samps[,2],ylab=expression(b),type="l")
plot(samps,xlab=expression(theta),ylab=expression(b))
```



```
# Plots of the candidate distribution
b <- seq(0,1,.01)
tuning <- 10; old <- .3
plot(b,dbeta(b,tuning*old,tuning*(1-old)),type="l")
tuning <- 10; old <- .7
plot(b,dbeta(b,tuning*old,tuning*(1-old)),type="l")
tuning <- 100; old <- .7
plot(b,dbeta(b,tuning*old,tuning*(1-old)),type="l")
```