Stat 532 Assignment 6

Kenny Flagg

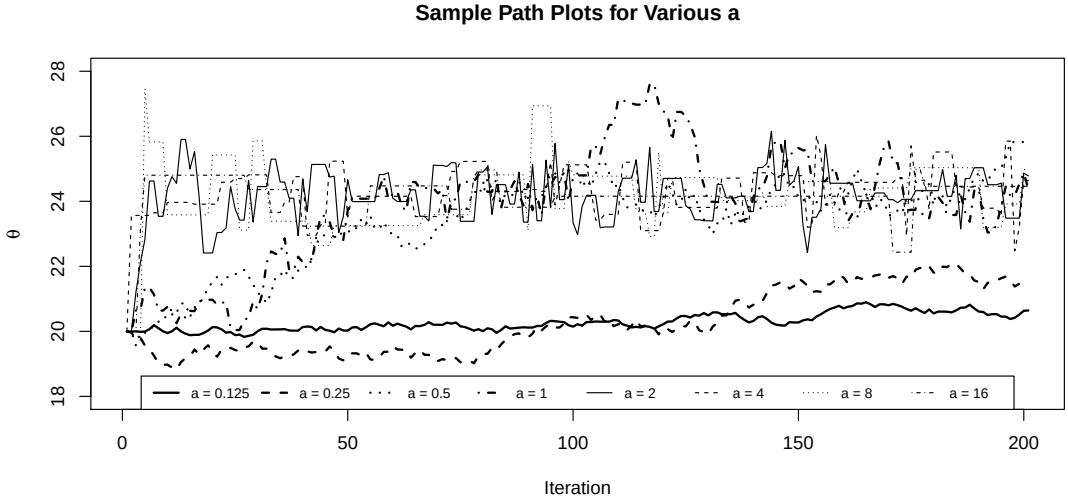
October 14, 2015

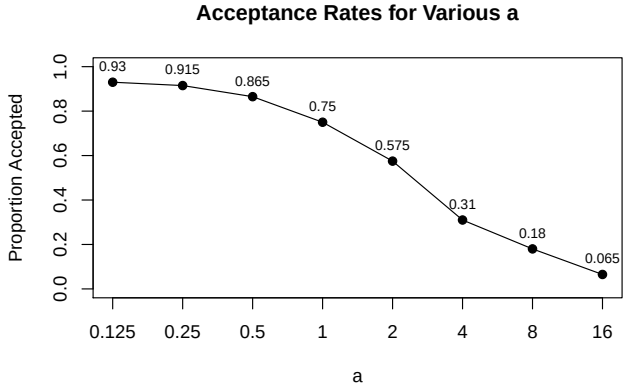
1. (a) I am finally getting in the habit of thoroughly commenting my code, so my Metropolis sampler goes on for nearly the next two pages. The results begin on page 3.

```
# Here's the data!
y.obs <- c(23, 24, 25, 26.5, 27.5)

# I like functions. Here's a function draw from J.
#  current  Current value of theta
#  a        Radius of support (tuning parameter)
draw.Unif.J <- function(current, a){return(runif(1, current - a, current + a))}

# Cauchy location parameter log-liklihood
loglik.Cauchy <- function(theta, y, scale = 1){
  # sapply allows theta to be a vector
  return(sapply(theta, function(x){sum(dcauchy(y, location = x, scale = scale, log = TRUE ))}))
}

# Normal prior log-density
logprior.Normal <- function(theta, mean = 20, v = 5){
  return(dnorm(theta, mean = mean, sd = sqrt(v), log = TRUE))
}

# Unscaled posterior log-density
logpost.theta <- function(theta, y = y.obs){
  return(loglik.Cauchy(theta, y) + logprior.Normal(theta))
}

# Single variable Metropolis sampler
#  theta.init  Starting value of theta
#  n.iter      Number of iterations
#  t           Log of target distribution
#              First argument must be theta (nothing else is passed)
#  J           Function to generate one draw from the jumping distrbution
#              First argument must be the current value of theta
#  ...         Additional arguments (tuning parameters) to pass to J
Metropolis.sampler <- function(theta.init, n.iter, t, J, ...){
  # Create a vector to store draws
  theta <- c(theta.init, numeric(n.iter))

  # I'm going to save everything for debugging/transarency/OCD purposes.
  # Create a vector to store candidates
  candidate <- numeric(n.iter)
```

```
  # Create a vector to store whether we accepted each candidate
  accept <- numeric(n.iter)

  # Create a vector to store Metropolis ratios
  r <- numeric(n.iter)

  # Create a vector of Unif(0, 1) draws
  u <- runif(n.iter)

  # Note: the theta vector index is ahead by 1.
  # The ith candidate is drawn from J(candidate[i]|theta[i]) because theta
  # has an extra entry, the initial value, at the begining.
  for(i in 1:n.iter){
    # Draw a candidate
    candidate[i] <- J(theta[i], ...)

    # Get the Metropolis ratio
    r[i] <- exp(t(candidate[i]) - t(theta[i]))

    # Accept the candidate if u < r
    # We'll always accept if the candidate has higher density (r > 1)
    if(u[i] < r[i]){
      theta[i+1] <- candidate[i]
      accept[i] <- TRUE
    }else{
          theta[i+1] <- theta[i]
          accept[i] <- FALSE
    }
  }

  # Calculate acceptance rate
  accept.rate <- mean(accept)

  # Make a list of all this stuff
  # Later on I might remove the initial value from theta, but for now
  # I want it on my path plots to see the whole convergence process.
  return(list('theta' = theta,
              'candidate' = candidate,
              'accept' = accept,
              'r' = r,
              'u' = u,
              'accept.rate' = accept.rate))
}
```

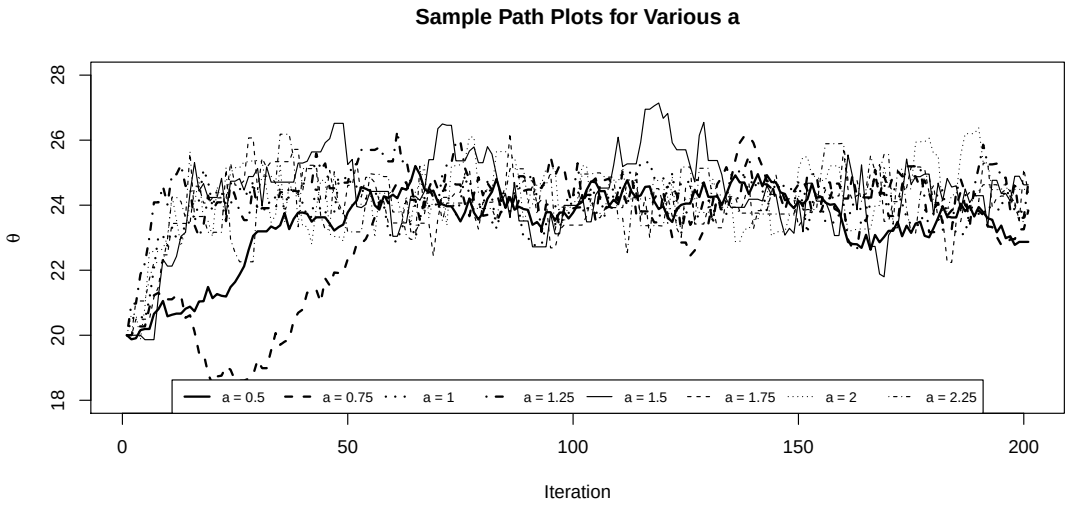
I initially had no I idea what values would be good choices for a, so I began by using powers of 2, from 2^{-3} to 2^4, to investigate different orders of magnitude. I set an initial value of $\theta = 20$ and ran my Metropolis sampler for 200 iterations with each a.

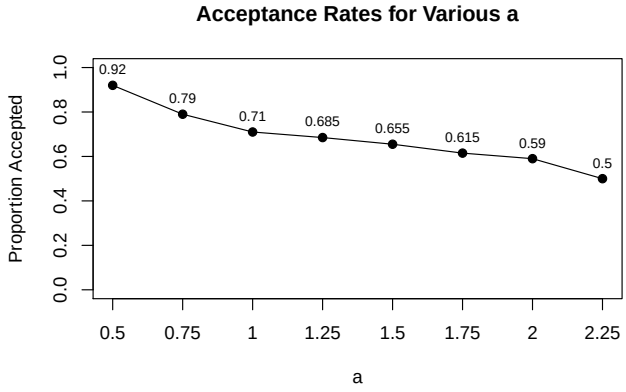




When a values of 2 and above were used, the sampler converged almost immediately, but the sample paths show many flat sections where draws were repeatedly rejected. The two smallest values of a, 0.125 and 0.25, resulted in the sampler failing to converge within 200 iterations. The other two values, 0.5 and 1, resulted in convergence after about 50 iterations.

Acceptance rate decreased as a increased. I would like to compromise between a high acceptance rate and fast convergence. It looks like a value around 1 would be acceptable, so I ran the sampler with additional values of a from 0.5 to 2.25 in steps of 0.25. Again, I used $\theta = 20$ as the initial value and ran algorithm for 200 iterations.

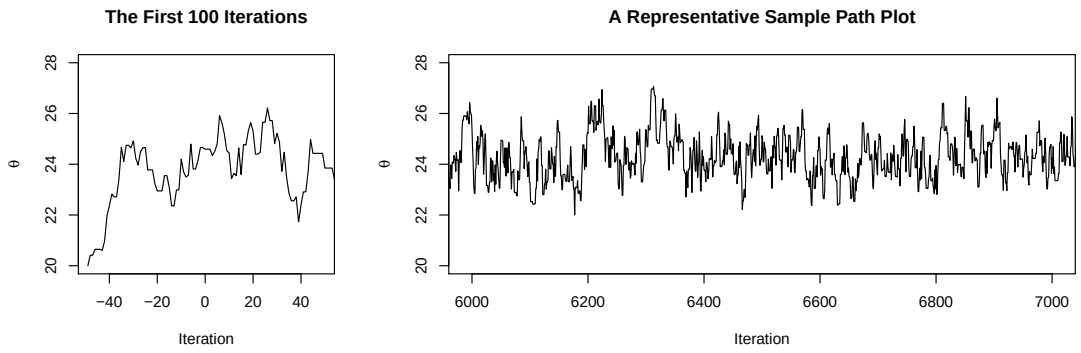


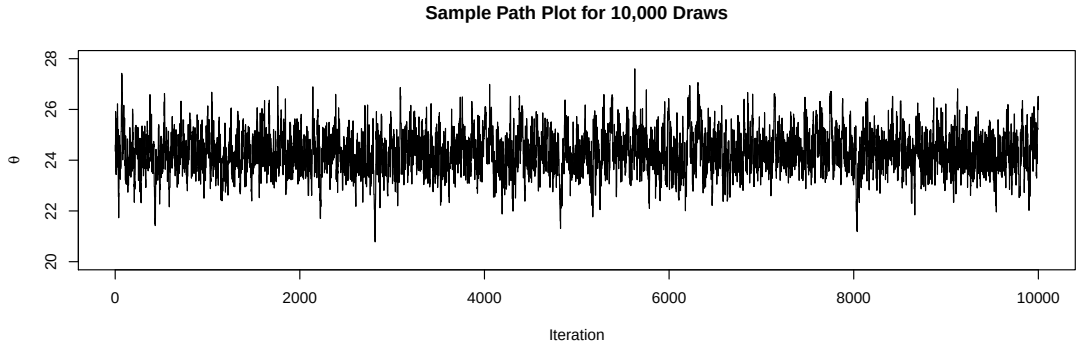


Values of 0.5 and 0.75 led to convergence around the 50th iteration. The values 1 and above all resulted in very quick convergence and the sample path plots all look similar. Again, the acceptance rate decreased as a increased.

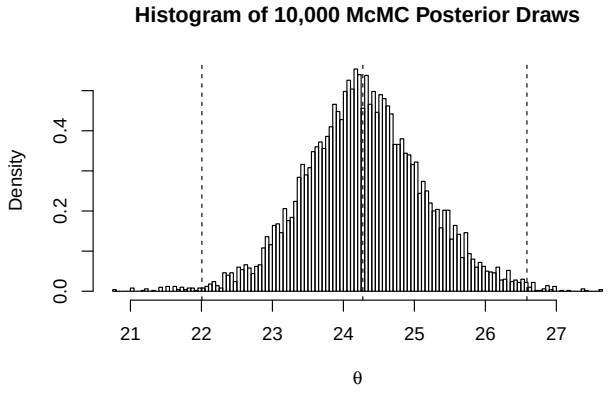
I chose $a = 1.25$ and ran the Metropolis sampler for 10,050 iterations, once again using $\theta = 20$ for the initial value. I then discarded the initial value and the first 50 draws as burn-in.

I used the sample path plots on the next page to assess convergence. The plot on the top left shows that the sampler converged well within 50 iterations. The path plot of 1,000 draws shows some minimal autocorrelation, but the sampler did move around the high-density area. The plot of the whole sequence shows that the sampler stayed in one area most of the time, but extreme values did occasionally occur.



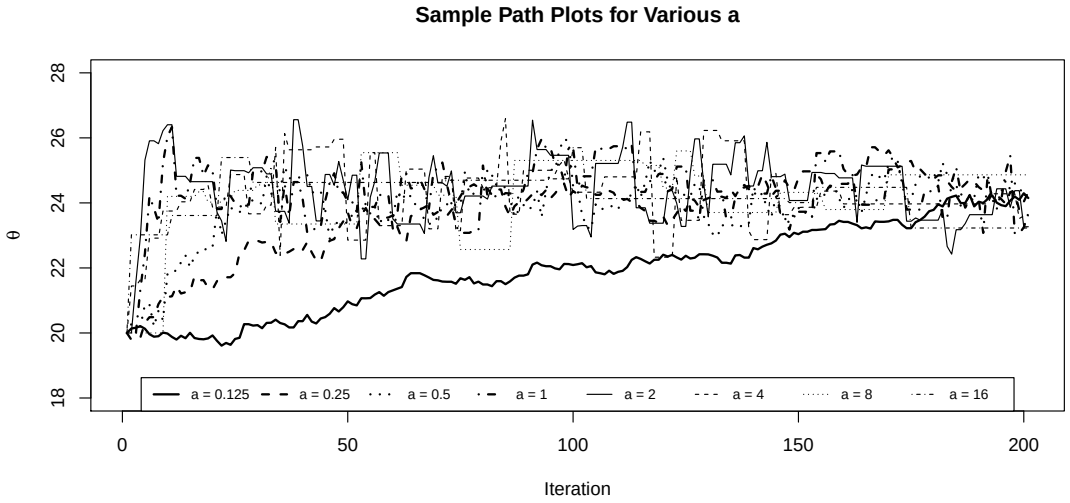


A histogram and summary statistics appear below. The vertical lines on the histogram show 99% confidence bounds and the median.



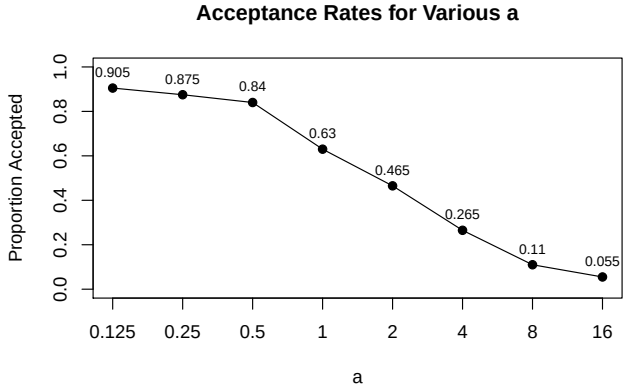
Posterior Mean	99% Posterior Interval	Posterior Median	$Pr(\theta > 25\|y)$	$Pr(\theta > 30\|y)$
24.3	(22.01, 26.59)	24.27	0.1919	0

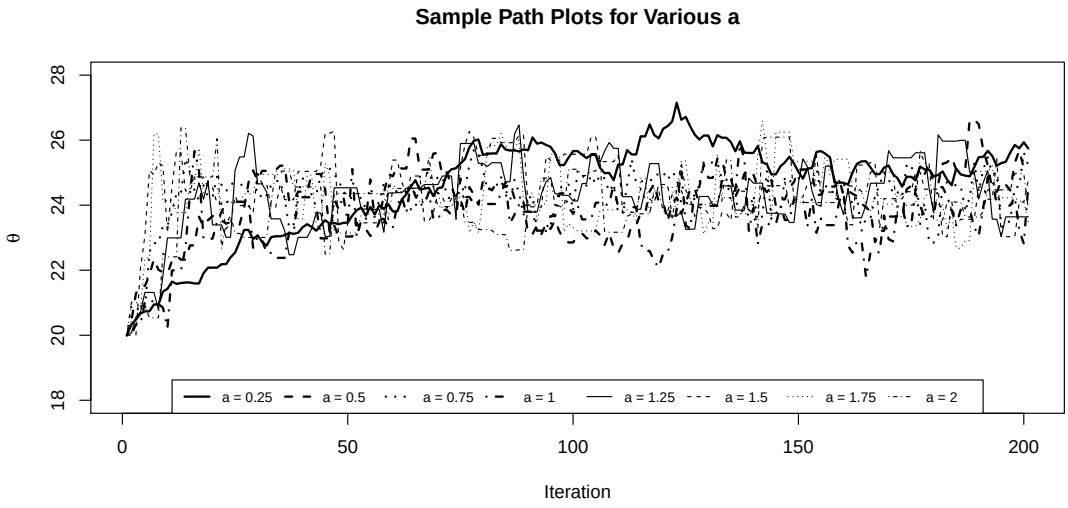
(b) The only modification I made to the code from part (a) was to define a new function to draw from the $N(\theta^{curr}|a^2)$ jumping distribution. Since most draws from the jumping distribution will be within $2a$ of the current value, I expected the ideal a to be smaller than it was in part (a). Just to check, I again started by using the same a values and an initial value of $\theta = 20$.

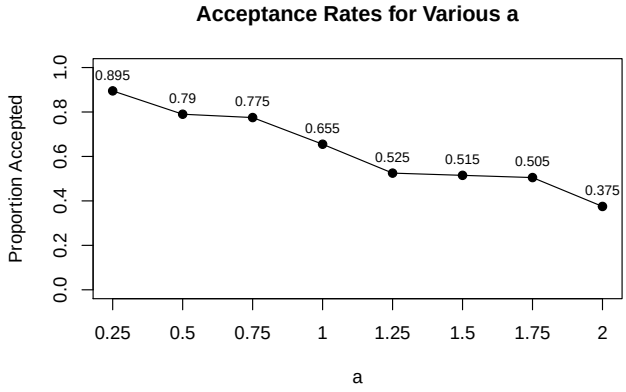


Values of $a = 1$ and larger resulted in very quick convergence. As before, increasing a decreased the acceptance rate. It looks like an acceptable trade-off is near $a = 1$.

I ran the sampler again for several values of a from 0.25 to 2.

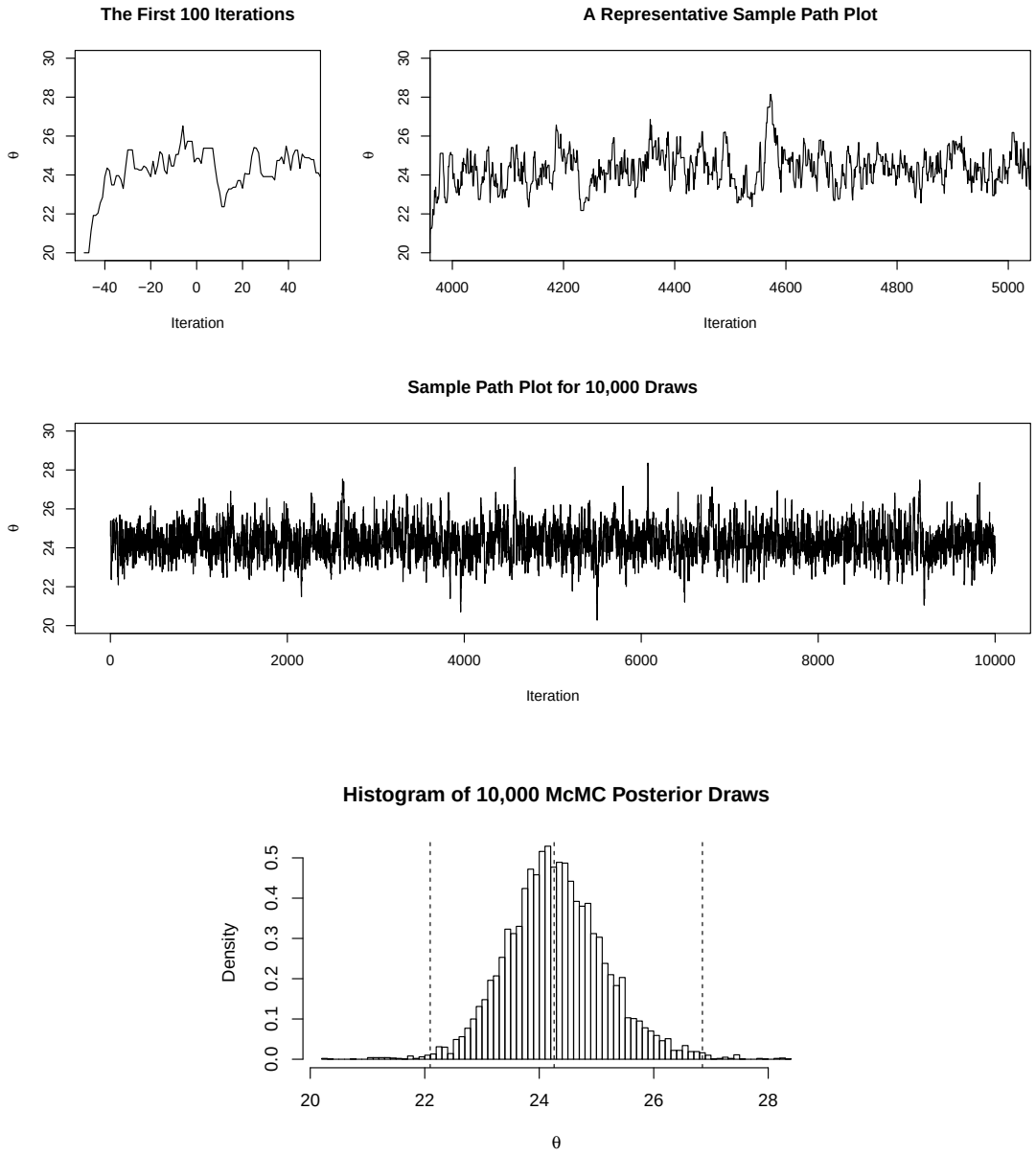






I decided $a = 0.75$ was a reasonable choice. I repeated the analysis from part (a), again using an initial value of $\theta = 20$, running the sampler for 10,050 iterations, and discarding the initial value and the first 50 draws as burn-in.

Sample path plots appear on the next page, followed by a histogram and a summary table. The algorithm converged almost immediately. At a glance, it looks like there were more large jumps than when the Uniform jumping distribution was used.



Posterior Mean	99% Posterior Interval	Posterior Median	$Pr(\theta > 25\|y)$	$Pr(\theta > 30\|y)$
24.3	(22.1, 26.85)	24.26	0.1913	0

(c) Both algorithms converged in fewer than 50 iterations and resulted in nearly identical posterior distributions. In both cases, I considered the tuning parameter as a sort of radius of the possible jumps; neither was difficult to tune. The only difference seems to be that the Normal allows occasional very large jumps, which might help it explore the parameter space more thoroughly. I prefer the Normal distribution, but only for this reason.

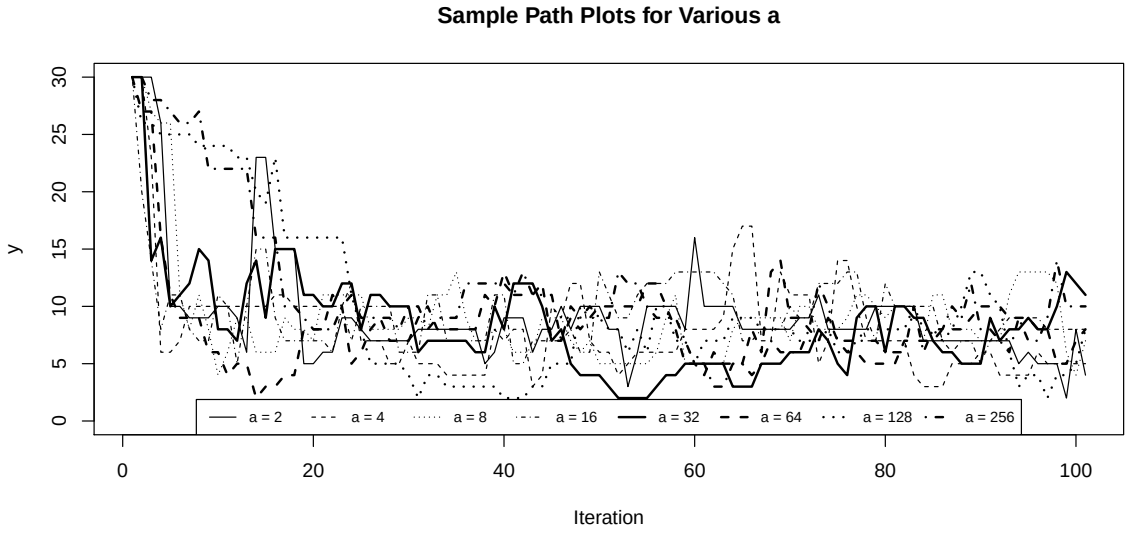
2. I modified my Metropolis sampler into a Metropolis-Hastings sampler. The only major change was to include a function to evaluate the density of the jumping distribution. The code for the sampler takes up most of the next two pages.

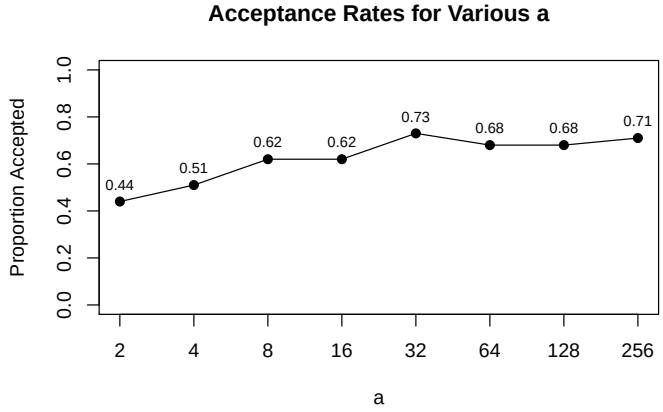
```
# Draw from J
#  current  Current value of y
#  a        Size (tuning parameter)
draw.nbin.J <- function(current, a){return(rnbinom(1, size = a, mu = current))}

# Log mass of J
#  candidate  Candidate value of y
#  current    Current value of y
#  a          Size (tuning parameter)
log.nbin.J <- function(candidate, current, a){
  return(dnbinom(candidate, size = a, mu = current, log = TRUE))
}

# Target mass funciton
log.target.y <- function(y, lambda = 8){
  return(dpois(y, lambda, log = TRUE))
}

# Single variable Metropolis-Hastings sampler
#  y.init  Starting value of y
#  n.iter  Number of iterations
#  t       Log of target distribution
#          First argument must be y (nothing else is passed)
#  Jdist   Log-density/mass function of the jumping distribution
#          First argument must be the value of y whose density is evaluated
#          Second arguments must be the value of y being conditioned upon
#  J       Function to generate one draw from the jumping distrbution
#          First argument must be the current value of y
#  ...     Additional arguments (tuning parameters) to pass to J and Jdist
MH.sampler <- function(y.init, n.iter, t, Jdist, J, ...){
  # Create a vector to store draws
  y <- c(y.init, numeric(n.iter))

  # I'm going to save everything for debugging/transarency/OCD purposes.
  # Create a vector to store candidates
  candidate <- numeric(n.iter)

  # Create a vector to store whether we accepted each candidate
  accept <- numeric(n.iter)

  # Create a vector to store Metropolis ratios
  r <- numeric(n.iter)

  # Create a vector of Unif(0, 1) draws
  u <- runif(n.iter)

  # Note: the theta vector index is ahead by 1.
  # The ith candidate is drawn from J(candidate[i]|y[i]) because y
  # has an extra entry, the initial value, at the begining.
  for(i in 1:n.iter){
    # Draw a candidate
```

```
    candidate[i] <- J(y[i], ...)

    # Get the Metropolis-Hastings ratio
    r[i] <- exp(t(candidate[i]) - t(y[i])
                + Jdist(y[i], candidate[i], ...)
                - Jdist(candidate[i], y[i], ...))

    # Accept the candidate if u < r
    # We'll always accept if the candidate has higher density (r > 1)
    if(u[i] < r[i]){
      y[i+1] <- candidate[i]
      accept[i] <- TRUE
    }else{
          y[i+1] <- y[i]
          accept[i] <- FALSE
    }
  }

  # Calculate acceptance rate
  accept.rate <- mean(accept)

  # Make a list of all this stuff
  # Later on I might remove the initial value from theta, but for now
  # I want it on my path plots to see the whole convergence process.
  return(list('y' = y,
              'candidate' = candidate,
              'accept' = accept,
              'r' = r,
              'u' = u,
              'accept.rate' = accept.rate))
}
```

The proposal distribution is Negative Binomial with mean y^{curr} and shape parameter a. The variance is $y^{curr} + \frac{(y^{curr})^2}{a}$. As a goes to infinity, the variance decreases to y^{curr}; decreasing a will increase the variance and encourage larger jumps. Since the variance also depends on y^{curr}, my instinct is to choose a to "balance" y^{curr} in the sense that a is small enough to prevent enormous jumps when y^{curr} is large, but still allow the sampler to easily move around the parameter space when y^{curr} is small. Since the target distribution has a known mean of 8, considering a on the order of 8 or 64 seems like a reasonable starting point.

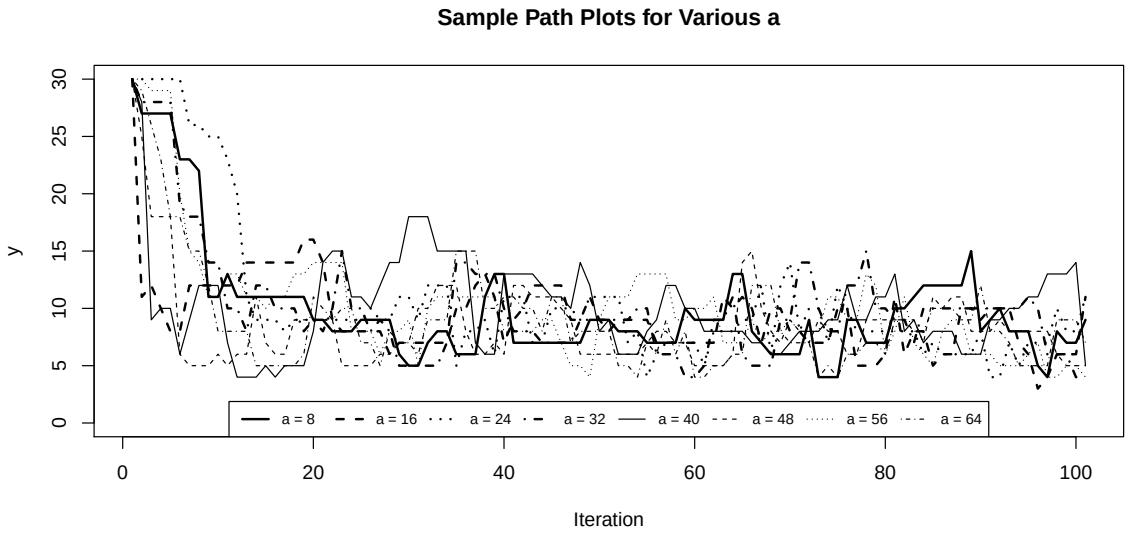
I started by setting an initial value of $y = 30$ and ran the sampler for 100 iterations for several different a values. In keeping with my theme for this assignments, I used powers of 2 to see the effect of a across different magnitudes. The sample path plots and acceptance rates appear on the next page.

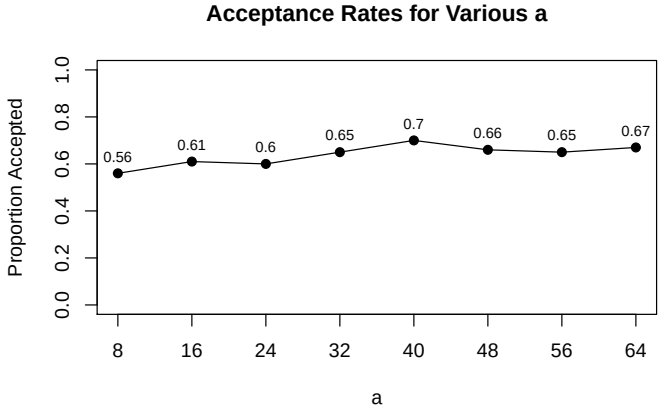




The largest a values, 128 and 256, led to convergence by the 30th iteration. The other runs reached the area of high probability in less than 10 iterations. I do not see any other notable differences in the sample path plot.

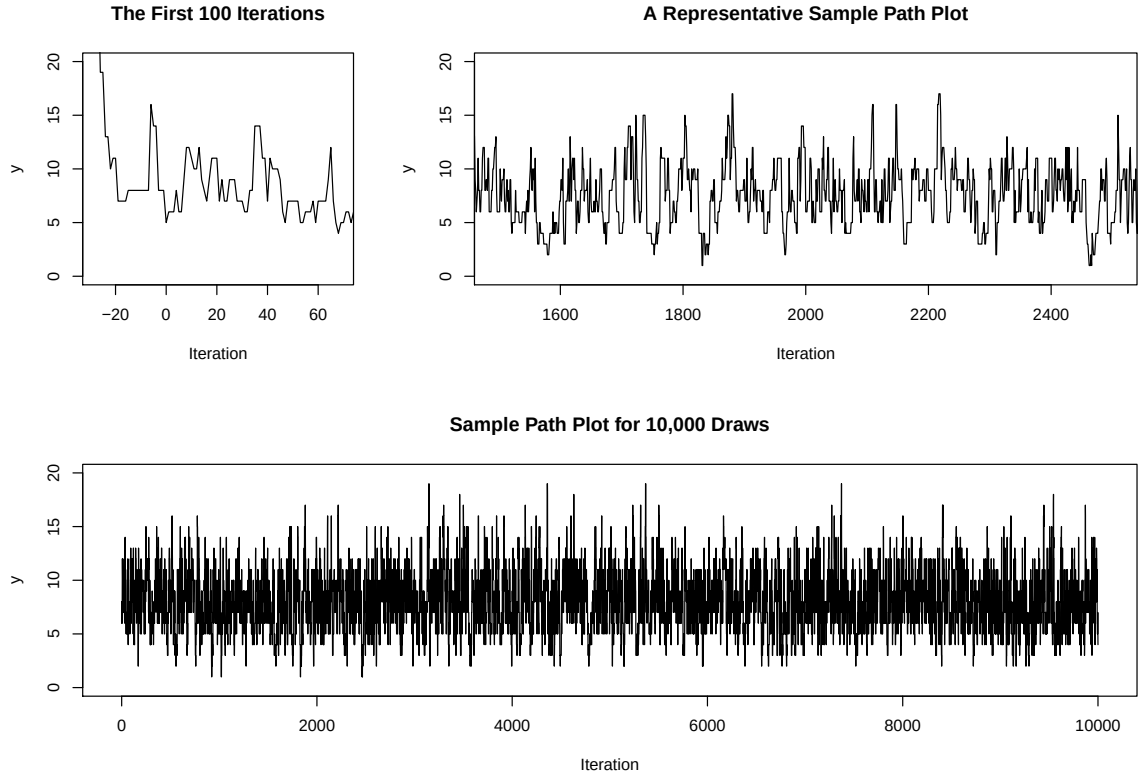
The acceptance rate plot shows a vague increasing trend that levels off around $a = 32$. It seemed worthwhile to consider some additional a values between 8 and 64, so I again ran the sampler for 100 iterations with an initial value of $y = 30$.



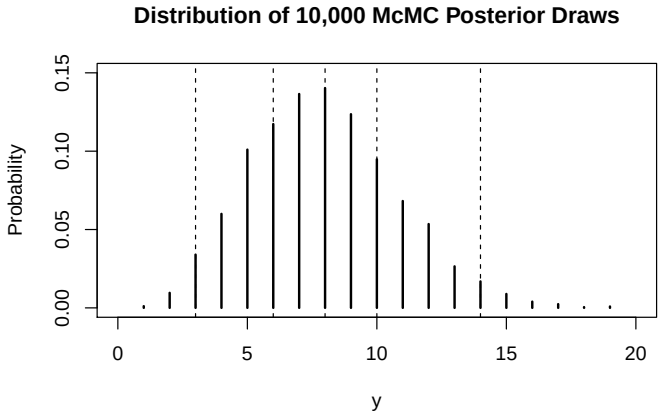


Again, the sample path plots look essentially the same. The acceptance rates have a bit of an increasing trend, but it seems like any value in the interval $[8, 64]$ would be reasonable. I chose $a = 32$ simply because it is near the middle of this interval.

I left the initial value at 30 and ran the Metropolis-Hastings sampler for 10,030 iterations. I discarded the initial value and first 30 draws as burn-in. Sample path plots and summaries appear on the following page.



The distribution of the draws appears below. The vertical lines mark some selected quantiles, also tabulated below the plot.



Posterior Mean	2.5%	25%	Posterior Median	75%	97.5%
7.95	3	6	8	10	14

3. Bayesian analyses often use simulation methods, like Markov chain Monte Carlo, because the task of finding a posterior distribution is often very difficult to do mathematically. It can involve tricky conditional probability calculations, and sometimes it is impossible to find an exact expression for the distribution function.

   There are several methods to indirectly generate random draws from the posterior distribution by using the probability model and the prior distribution. For making inference, an extremely large sample from the posterior is just as useful as a formula. A computer can draw tens of thousands of random numbers much more easily than a human can do complicated calculus problems to describe the distribution analytically, so computational methods are a natural alternative.

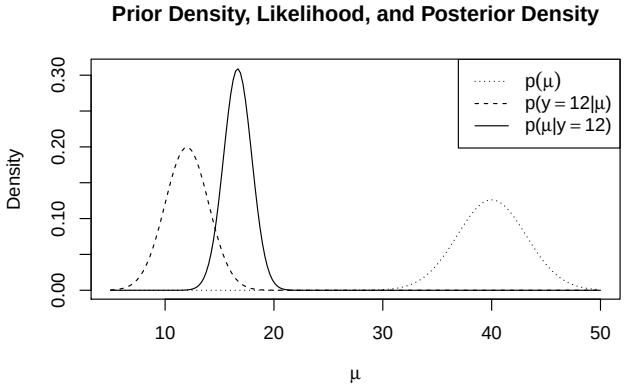
   However, it should be noted that a Bayesian analysis does not have to use simulations, and Markov chain Monte Carlo methods are not used exclusively by Bayesian statisticians. For simple Bayesian analyses of Normal means or Binomial probabilities, there exist simple formulas to find posterior distributions; computer-intensive techniques are not needed. Also, McMC is frequently used to fit multilevel models outside of a Bayesian context. Such analyses typically use non-informative priors and are equivalent to estimating the parameters by maximizing the likelihood. The posterior distribution is used to find the standard errors of the estimators, and the results are interpreted in a frequentist way. Bayesian data analysis is defined by the philosophy of finding a posterior distribution, not by the use of Monte Carlo methods.

4. (a) The model is $y|\mu \sim \mathrm{N}(\mu, \sigma^2)$, $\mu \sim \mathrm{N}(\mu_0, \tau_0^2)$, where $\sigma^2 = 2$ is known. For now, I will set $\mu_0 = 40$ and $\tau_0^2 = 10$ to create a weak prior that disagrees with the data.
   Then the posterior distribution of μ is $\mu|y \sim \mathrm{N}(\mu_1, \tau_1^2)$, where

$$\mu_1 = \frac{\frac{\mu_0}{\tau_0^2} + \frac{y}{\sigma^2}}{\frac{1}{\tau_0^2} + \frac{1}{\sigma^2}} = \frac{\frac{40}{10} + \frac{12}{2}}{\frac{1}{10} + \frac{1}{2}} = \frac{50}{3} \approx 16.67,$$

$$\tau_1^2 = \frac{1}{\frac{1}{\tau_0^2} + \frac{1}{\sigma^2}} = \frac{1}{\frac{1}{10} + \frac{1}{2}} = \frac{5}{3} \approx 1.667.$$

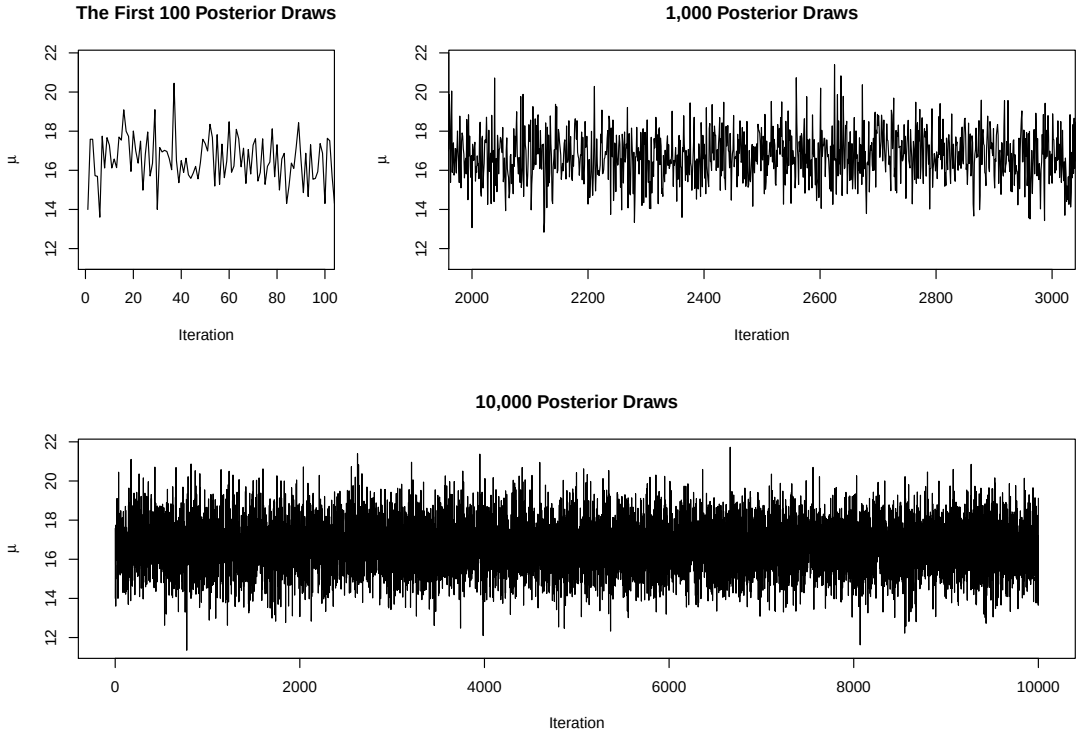
   The distribution curves appear on the plot below.



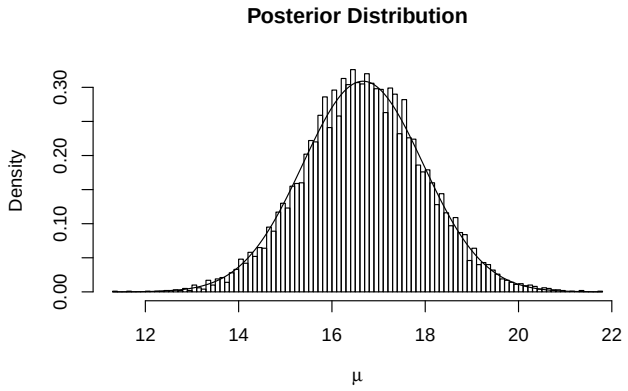
(b) JAGS requires precisions instead of variances or standard deviations, so I have defined the precisions of y and μ as $p_y = \dfrac{1}{\sigma^2}$ and $p_\mu = \dfrac{1}{\tau_0}$.

```
# Variables:
#  mu    Mean (parameter of interest)
#  y     Data (a single observation) - specified in data argument
#  p.y   Data precision - specified in data argument
#  mu.0  Prior mean - specified in data argument
#  p.mu  Prior precision specified in data argument
jags.model <- function(){
  # Likelihood
  y ~ dnorm(mu, p.y)

  # Prior
  mu ~ dnorm(mu.0, p.mu)
}
jags.data <- list('y' = 12,
                  'p.y' = 1/2, # = 1/sigma^2
                  'mu.0' = 40,
                  'p.mu' = 1/10) # = 1/tau_0^2
```

(c) I set an initial value of $\mu = 30$ and then ran one chain for 10,000 iterations, discarding none as burn-in. I chose the initial value to be far from the center of the posterior, but the sampler still converged immediately.

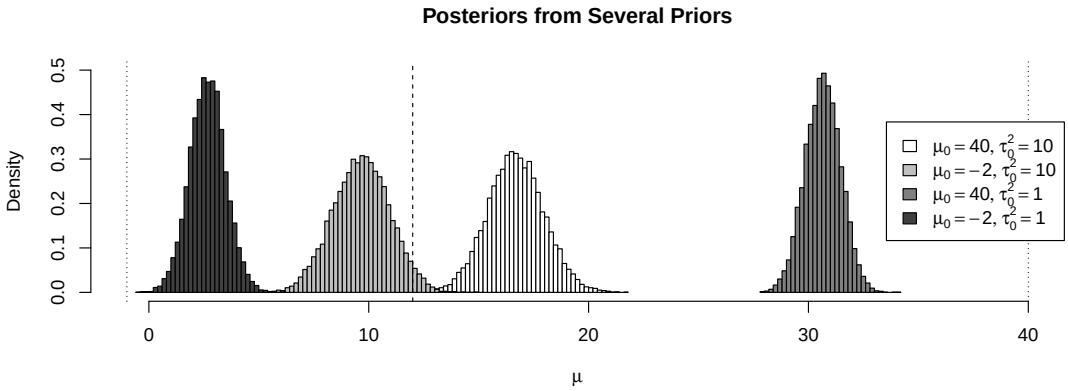


(d) The following plot shows a histogram of the 10,000 McMC draws with the $\mathrm{N}\left(\frac{50}{3}, \frac{5}{3}\right)$ density superimposed. The resemblance is uncanny.



(e) I previously used $\mu_0 = 40$, $\tau_0^2 = 10$. I will now compare each combination of $\mu_0 \in \{-2, 40\}$ and $\tau_0^2 \in \{1, 10\}$. I chose the $\mu_0 = -2$ to center the prior on the other side of y, but have it closer to y than the prior centered at $\mu_0 = 40$ was. I chose $\tau_0^2 = 1$ to strengthen the prior.

I generated 10,000 draws using each prior and plotted the results on the histogram below. The dashed vertical line shows the observed value, $y = 12$, and the dotted vertical lines show the values of μ_0. Just as one would expect after studying the analytical formulas, the posterior distributions are centered between their respective prior means and 12. The stronger priors resulted in posteriors that remained nearer to their prior means.



(f) According to the JAGS webpage, "JAGS is Just Another Gibbs Sampler" (`http://mcmc-jags.sourceforge.net/`).

Gelman & Hill say that BUGS stands for "Bayesian Inference using Gibbs Sampling" on page 11, although their capitalization implies that it should be abbreviated as "BIGS."

Page xvii of the Stan user's guide (found at `http://mc-stan.org/`) states that Stan is named for Stanislaw Ulam, a "coinventor of Monte Carlo methods" who collaborated with Nicholas Metropolis.

5. (a) I will be as brief as I can, but the model is complicated and worth reiterating.
Sinding-Larsen and Xu used straightforward Monte Carlo simulation to approximate the posterior distribution of their parameters. The model and simulation process was described in Xu and Sinding-Larsen as follows.
The joint probability function is

$$p(Y,S,N,\beta,\mu,\sigma^2) = p(N,\beta,\mu,\sigma^2)p(Y,S|N,\beta,\mu,\sigma^2)$$
$$= p(N)p(\beta)p(\mu)p(\sigma^2)\binom{N}{n}\prod_{j=1}^{n}\frac{p(Y_j|\mu,\sigma^2)Y_j^{\beta}}{\sum_{k=j}^{n}Y_k^{\beta}+\sum_{k=n+1}^{N}S_k^{\beta}}\prod_{j=n+1}^{N}p(S_j|N,\beta,\mu,\sigma^2)$$

where

- $Y_1,\ldots,Y_n,S_{n+1},\ldots,S_N$ are the sizes of all the oil pools in the study area. $Y = (Y_1,\ldots,Y_n)$ are the observed data (for pools that have been discovered; $n = 22$), and $S = (S_{n+1},\ldots,S_N)$ are the sizes of the undiscovered pools. It is assumed that $Y_1,\ldots,Y_n,S_{n+1},\ldots,S_N$ are independent (conditional on the order in which they were discovered), $\log(Y_1),\ldots,\log(Y_n),\log(S_{n+1}),\ldots,\log(S_N)$ come from a $\mathrm{N}(\mu,\sigma^2)$ population, and

  $$Pr(Y_1,\ldots,Y_n \text{ were the first } n \text{ discoveries}|N,\beta) = \binom{N}{n}\prod_{j=1}^{n}\frac{Y_j^{\beta}}{\sum_{k=j}^{n}Y_k^{\beta}+\sum_{k=n+1}^{N}S_k^{\beta}}.$$

- $\mu \sim \mathrm{N}(2.38, 0.54)$ is the mean size of all the pools,
- σ^2 follows a Gamma distribution with $E(\sigma^2) = 2.89$ and $Var(\sigma^2) = 0.54$,
- $\beta \sim \mathrm{Unif}(-0.213, -0.088)$,
- $N|M,\pi \sim \mathrm{Binomial}(M,\pi)$ is the unknown total number of pools in the area,
- $\pi \sim \mathrm{Beta}(2.30, 4.28)$,
- and $M-n$ is the unknown number of prospective pools yet to be tested. In Sinding-Larsen and Xu, it is said that the distribution of $M-n$ is based on expert knowledge and satisfies

m	13	16	20	26	30	35	38
$Pr(M-n \geq m)$	1.00	0.95	0.75	0.50	0.25	0.05	0.00

  but the exact form is not given.

Before simulating the posterior, they used the following Monte Carlo process to approximate the unconditional prior distribution of N.

i. Draw M and π.
ii. Draw $N|M,\pi$.

They squashed (our term, not theirs) over M and π to get $p(N)$. Then they simulated the joint posterior distribution of S,N,β,μ,σ^2 as follows.

i. Draw N, β, μ, and σ^2.
ii. Draw $\log(S_{n+1}),\ldots,\log(S_N)$ and compute $(S_{n+1}),\ldots,(S_N)$.
iii. Compute the weight,

$$W = \binom{N}{n}\prod_{j=1}^{n}\frac{p(Y_j|\mu,\sigma^2)Y_j^{\beta}}{\sum_{k=j}^{n}Y_k^{\beta}+\sum_{k=n+1}^{N}S_k^{\beta}}.$$

The marginal distribution of S_j was found by squashing over the other parameters. The marginal posterior distributions of N, β, μ, and σ^2 were found by summing the W values and normalizing. The results were presented in histograms and cumulative distribution plots.

(b) They did not use an iterative algorithm, so convergence was not an issue. They said they took 80,000 posterior draws, which sounds like a lot. It is more than I ever used on a Stat 506 assignment, and certainly more than Andrew Gelman recommends for the situations we have studied. They do not explain why they used so many draws; however each draw includes a sequence of undiscovered pool sizes, the number of which is itself a random draw. I find it believable that tens or hundreds of thousands of draws are necessary to accurately characterize the distribution, but they did not justify whether 80,000 is enough. Their histograms look a bit jagged, so I am not fully convinced that they took enough draws.

References

- Sinding-Larsen, R., and Xu, J., 2005, Bayesian Discovery Process Modeling of the Lower and Middle Jurassic Play of the Halten Terrace, Offshore Norway, as Compared with the Previous Modeling: Natural Resources Research, v. 14, no. 3, p. 235-248.
- Xu, J., and Sinding-Larsen, R., 2005, How to choose priors for Bayesian estimation of the discovery process model: Natural Resources Research, v. 14, no. 3, p.211 -233.