Stat 532 Assignment 8 (Part 1)

Kenny Flagg

October 30, 2015

1. (a) We were not given any specific prior information, so I decided to use a weakly informative prior with small values for the hyperparameters. I chose $\tau \sim \text{Gamma}\left(\frac{1}{2}, 1\right)$, $\mu|\tau \sim \text{N}\left(0, \frac{1}{\tau}\right)$. Figure 1 shows the prior density plotted on both the μ, τ-space and the μ, σ-space. Since μ is on the logit scale, the distribution is disperse enough to cover most of the reasonable values. It will not be a problem that the prior mode is at $\mu = 0$ even though this corresponds to a cancer rate of 0.5.

   For the Gibbs sampler, I used a Metropolis-Hastings step to draw the ϕ_i. I used a $\text{N}\left(\mu, \frac{1}{\tau}\right)$ jumping distribution so that the Metropolis ratio simplified to

$$r = \frac{\text{expit}(\phi_i^{cand})^{y_i}(1 - \text{expit}(\phi_i^{cand}))^{m_i - y_1}}{\text{expit}(\phi_i^{curr})^{y_i}(1 - \text{expit}(\phi_i^{curr}))^{m_i - y_1}}.$$

   I ran four chains with initial values of

   i. $\tau = 1$, $\mu = 0$, $\phi_i = 0; i = 1, \ldots, 20$,
   ii. $\tau = 1$, $\mu = -2$, $\phi_i = -2; i = 1, \ldots, 20$,
   iii. $\tau = 0.25$, $\mu = 0$, $\phi_i = 0; i = 1, \ldots, 20$,
   iv. $\tau = 0.25$, $\mu = -2$, $\phi_i = -2; i = 1, \ldots, 20$.

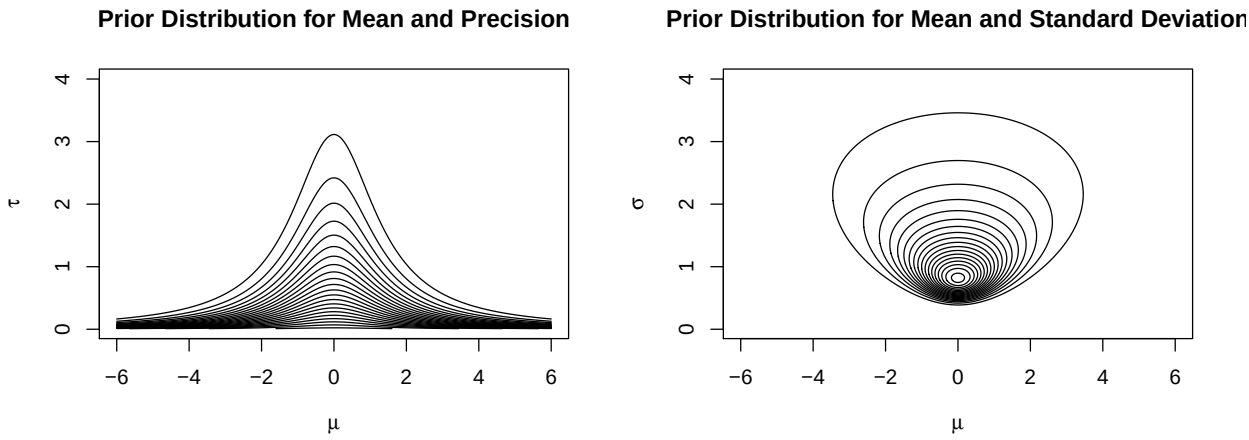

Figure 1: Left: Prior distribution of μ, τ. Right: Prior distribution of μ, σ

The chains converged very quickly so I used 100 iterations as warmup and then ran each chain for another 5,000 simulations. The code for my sampler appears below.

```
n.chains <- 4
n.iter <- 5100 # 100 warmup plus 5000 draws from each chain
set.seed(4863)

# Complete conditionals
draw.tau <- function(mu, a, b, k0, t0, phi, n){
  return(rgamma(1, (n+1)/2 + a, sum((phi-mu)^2)/2 + k0*(mu-t0)^2/2 + b))
}
draw.mu <- function(tau, a, b, k0, t0, phi, n){
  return(rnorm(1, (n*mean(phi)+k0*t0)/(n+k0), 1/sqrt(tau*(n+k0))))
}

# Log Metropolis ratio component (I canceled stuff on paper)
l.r.phi <- function(phi, y, m){
  return(y*log(expit(phi))+(m-y)*log(1-expit(phi)))
}
draw.phi.cands <- function(mu, tau, n){
  return(rnorm(n, mu, 1/sqrt(tau)))
}

# Initialize a list of lists to hold the draws
cancer.gibbs <- replicate(n.chains, simplify = FALSE,
                          list('p' = matrix(nrow = n.iter, ncol = n),
                               'phi' = matrix(nrow = n.iter, ncol = n),
                               'mu' = numeric(n.iter),
                               'tau' = numeric(n.iter),
                               'sigma' = numeric(n.iter)))

# Initial values for chain 1
cancer.gibbs[[1]]$phi[1,] <- 0
cancer.gibbs[[1]]$mu[1] <- 0
cancer.gibbs[[1]]$tau[1] <- 1

# Initial values for chain 2
cancer.gibbs[[2]]$phi[1,] <- -2
cancer.gibbs[[2]]$mu[1] <- -2
cancer.gibbs[[2]]$tau[1] <- 1

# Initial values for chain 3
cancer.gibbs[[3]]$phi[1,] <- 0
cancer.gibbs[[3]]$mu[1] <- 0
cancer.gibbs[[3]]$tau[1] <- 0.25

# Initial values for chain 4
cancer.gibbs[[4]]$phi[1,] <- -2
cancer.gibbs[[4]]$mu[1] <- -2
cancer.gibbs[[4]]$tau[1] <- 0.25

# Big outer loop for the iterations
for(i in 2:n.iter){
  # I like to do things in parallel, so the inner loop is for the chains
  for(j in 1:n.chains){
    # Tau
```

```
    cancer.gibbs[[j]]$tau[i] <- draw.tau(cancer.gibbs[[j]]$mu[i-1],
                                         a, b, k0, t0,
                                         cancer.gibbs[[j]]$phi[i-1,], n)
    # Sigma
    cancer.gibbs[[j]]$sigma[i] <- 1 / sqrt(cancer.gibbs[[j]]$tau[i])
    # Mu
    cancer.gibbs[[j]]$mu[i] <- draw.mu(cancer.gibbs[[j]]$tau[i],
                                       a, b, k0, t0,
                                       cancer.gibbs[[j]]$phi[i-1,], n)
    # Phi
    # These are conditionally independent, so I'll do them all at once.
    phi.cands <- with(cancer.gibbs[[j]], draw.phi.cands(mu[i], tau[i], n))
    cancer.gibbs[[j]]$phi[i,] <- with(cancer.gibbs[[j]],
      ifelse(runif(n) < exp(
        l.r.phi(phi.cands, y, m) - l.r.phi(phi[i-1,], y, m)
        ), phi.cands, phi[i-1,]))
    # P
    cancer.gibbs[[j]]$p[i,] <- expit(cancer.gibbs[[j]]$phi[i,])
  }
}
```

I re-ran the Beta-Binomial rejection sampling example code for 80,000 draws. 20,249 were accepted. The parameter of interest is the overall stomach cancer rate, which is expit(μ) in the Logit-Normal model and η in the Beta-Binomial model. Figure 2 compares the posterior distributions. The Logit-Normal model included an additional hierarchical level where the cancer rate for each city was modeled. The Beta-Binomial model used complete pooling, so the inferences should be somewhat different.

The Logit-Normal model estimates the stomach cancer rate for the population to be lower than the estimate from the Beta-Binomial model. The Logit-Normal partial-

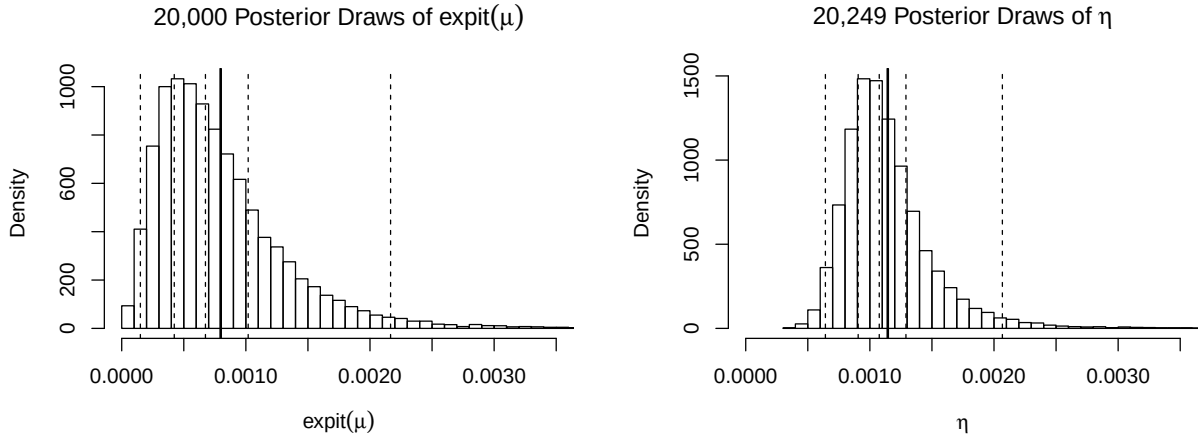

Figure 2: Posterior distributions on the probability scale. The solid vertical lines are the posterior means. The dashed vertical lines are the 2.5, 25, 50, 75, and 97.5 percentiles. Left: Draws of expit(μ) from the Logit-Normal model. Right: Draws of η from the Beta-Binomial model.

pooling model found the posterior mean of expit(μ) to be 0.00080 and the posterior median to be 0.00067. The Beta-Binomial model estimated the posterior mean of η as 0.00114 and the median was 0.00108.

(b) The Logit-Normal results included 20,000 posterior draws of each ϕ_i, so I implemented a posterior predictive check by drawing one $\tilde{y}_i \sim \text{Binomial}(m_i, \text{expit}(\phi_i))$ for each draw of each ϕ_i.

The Beta-Binomial model used a single η for all cities, so I drew $\tilde{y}_i \sim \text{Binomial}(m_i, \eta); i = 1, \ldots, 20$ for each of the 20,249 draws of η.

To put everything on the same continuous scale, I divided each draw by the city's population. Figure 3 displays the resulting posterior predictive distributions of $\frac{\tilde{y}_i}{m_i}$.

The complete pooling is apparent in the right panel. The distributions are centered near 0.001, regardless of the original data. In contrast, the left panel shows that the predictions form the hierarchical model were centered near the observed values.

(c) Of these two particular models, I would use the hierarchical Logit-Normal model for inference. Since it models the individual ϕ_i it allows both for precise inference within the same 20 cities and for inference that includes additional uncertainty for new cities.

In general, however, I think I would prefer a multilevel version of the Beta-Binomial. The η, K parameterization is easier to understand than the logit-scale μ, τ.

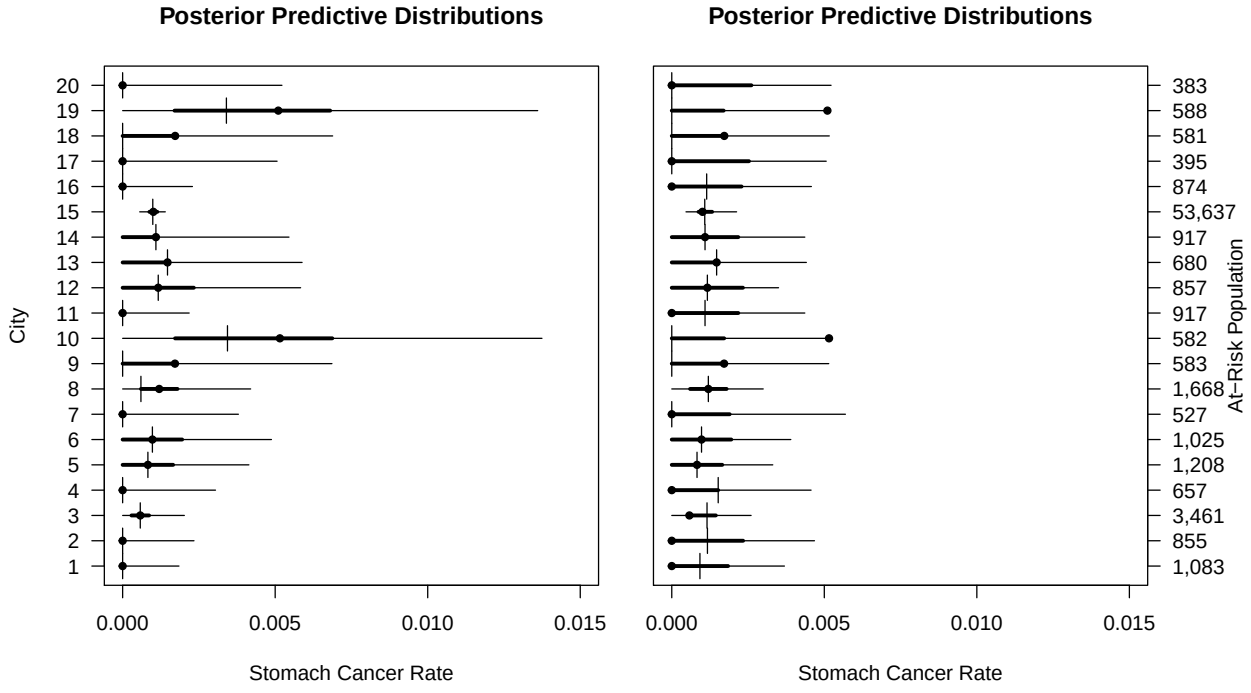

Figure 3: Posterior predictive distributions for each city. Dots represent the original observed cancer rates and vertical bars are the posterior medians. Horizontal bars show 50% and 95% posterior intervals. Left: Logit-Normal model. Right: Beta-Binomial model.

(d) My Stan model code appears below. I ran four chains with the same initial values that I used for the Gibbs sampler. Since the simulation ran blazingly fast, I ran each chain for 10,000 iterations and discarded the first half as warmup.

```
cancer.code <- '
data{
  int<lower=0> n;
  int<lower=0> y[n];
  int<lower=0> m[n];
  real<lower=0> a;
  real<lower=0> b;
  real t0;
  real<lower=0> k0;
}

parameters{
  vector[n] phi;
  real mu;
  real<lower=0> tau;
}

transformed parameters{
  real sigma;
  vector<lower=0, upper=1>[n] p;
  sigma <- 1 / sqrt(tau);
  for(i in 1:n){
    p[i] <- inv_logit(phi[i]);
  }
}

model{
  for(i in 1:n){
    phi[i] ~ normal(mu, sigma);
    y[i] ~ binomial(m[i], p[i]);
  }
  tau ~ gamma(a, b);
  mu ~ normal(t0, sigma / sqrt(k0));
}
'


cancer.data <- list('n' = n,
                    'y' = y,
                    'm' = m,
                    'a' = a,
                    'b' = b,
                    't0' = t0,
                    'k0' = k0)

cancer.stan <- stan_model(model_code = cancer.code, model_name = 'cancer')
```

(e) Figures 4 and 5 compare the posterior distributions of μ and π_i from the Gibbs sampler and from Stan. They are practically identical.

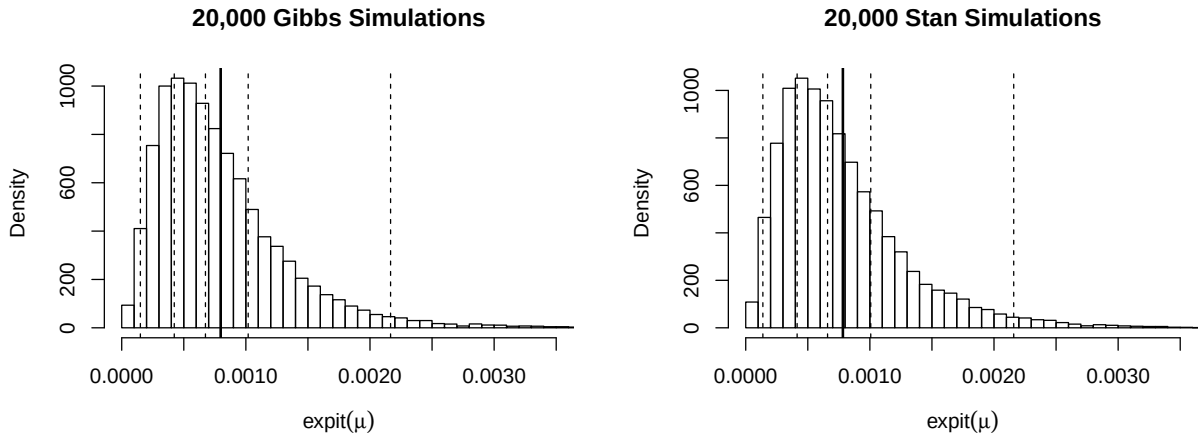

Figure 4: Posterior distributions of μ on the probability scale. The solid vertical lines are the posterior means. The dashed vertical lines are the 2.5, 25, 50, 75, and 97.5 percentiles.

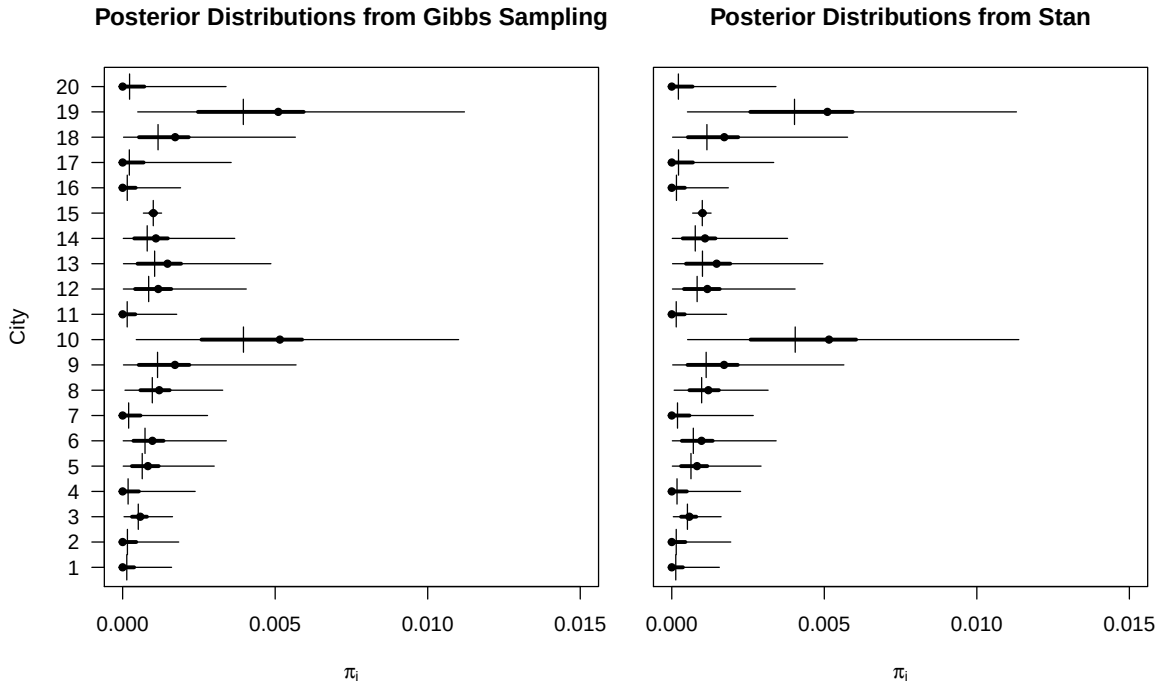

Figure 5: Posterior distributions of the π_i. Dots represent the observed cancer rates and vertical bars are the posterior medians. Horizontal bars show 50% and 95% posterior intervals.

4. (a) I worked through the HMC exercise before I started reading the sections. While reading, I found it helpful to picture a ball rolling around on a surface over a two-dimensional parameter space. It took several readings for me to understand that the intermediate θ values during the leapfrog steps are not saved. My understanding is now that the HMC algorithm is much like a Metropolis sampler where candidates are generated by randomly perturbing a particle's momentum and then following its trajectory through the parameter space.

   I tried not to think too deeply about physics, tuning, or local adaptation. I did think a bit about randomly varying ϵ and L. It seems like the Hamiltonian sampler will share the Gibbs sampler's problems with disjoint parameter spaces, but I wonder if allowing a large ϵ to occur would be an good way to let the chain jump between disjoint regions.

   I did not find Section 12.5 to be very helpful, but if I find myself writing an HMC sampler I will certainly return to it for guidance.

   (b) The visual aid was helpful, and I found it easy to picture higher-dimensional examples in my head.

   I ran through the exercise once, read the sections, and then returned to the exercise. At some point after running the roll function the second time, I came to understood that it only demonstrated one iteration of the sampler. When I find the time, I want to extend the demonstration to include several iterations with additional random draws of ϕ.

   Below are the functions I wrote for the introduction exercise. I plotted the posterior curve in Figure 6.

```
phi.post <- function(phi, y, n, mu, sigma){
  return(exp(phi)^y * (1+exp(phi))^(-n) * exp(-((phi-mu)^2)) / (2*sigma^2))
}

phi.neglog.gradient <- function(phi, y, n, mu, sigma){
  return(-y+n*exp(phi)/(1+exp(phi))+(phi-mu)/(sigma^2))
}
```

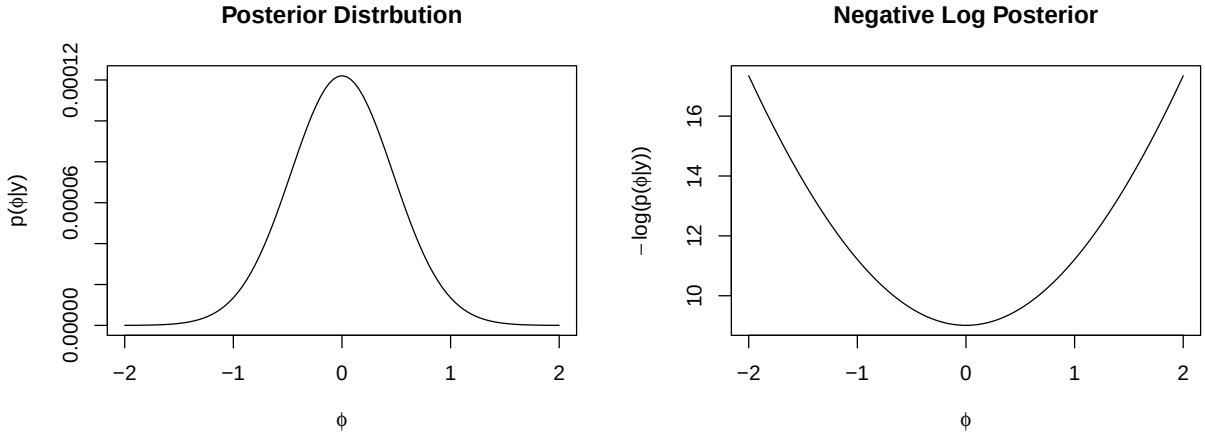

Figure 6: The posterior distribution for the HMC exercise.

(c) I first ran the Normal example. Initially I ran it as is, saw that Stan was working, and moved on. I later came back and compared the three different versions of the model code. The results had no differences that I noticed. The `increment_log_prob(-theta^2 / 2)` version won my favor as the most interesting way to code the model. Figure 7 shows some results of the `increment_log_prob(-theta^2 / 2)` model.

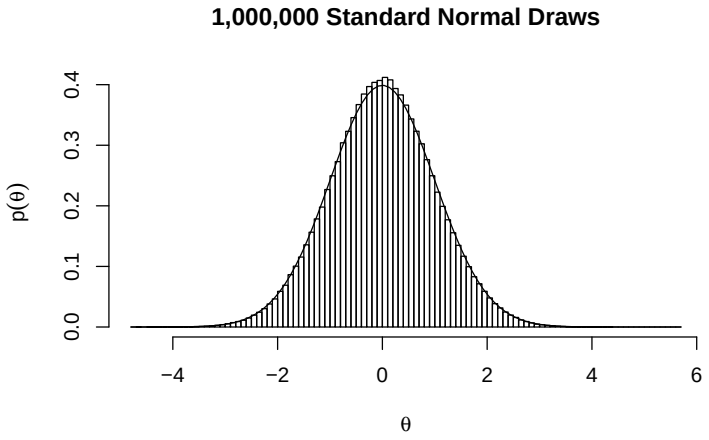

Figure 7: Histogram of Stan draws with standard normal curve.

The first thing I noticed about the linear model example was that it didn't work, as shown by the incorrect estimates in Table 1. As soon as I looked at the model code I noticed that there was an $N(100, 1)$ prior on the intercept, and none of the other parameters had priors specified. I placed vague $N(0, 100)$ priors on all of the coefficients and got results that match the `lm` results.

	`lm` Results		Example Stan Results		My Stan Results	
	Estimate	SE	Posterior Mean	Posterior SD	Posterior Mean	Posterior SD
(Intercept)	38.75	1.79	99.54	0.99	38.77	1.92
hp	-0.02	0.01	0.12	0.07	-0.02	0.01
cyl	-0.94	0.55	-10.35	3.22	-0.96	0.59
wt	-3.17	0.74	-9.15	4.81	-3.14	0.77
Residual SE	2.51		16.83	2.38	2.64	0.37

Table 1: Comparison of an `lm` fit, the example Stan model, and my Stan model

When I ran the censored data example, I got warning messages saying that an inverse scale parameter was zero. The posterior draws ranged from about 0.003 to 0.007, which seemed like suspiciously little variation.

In order to figure out what was going on, I found the posterior distribution analytically. For a single observation y_i, $p(y_i|\lambda) = \lambda e^{-\lambda y_i}$ and $Pr(y_i \geq C|\lambda) = e^{-\lambda C}$. Then if $p(\lambda) \propto 1$

and the last N_{cens} of the y_i are censored, the posterior distribution is

$$p(\lambda|y, N_{cens}) \propto \lambda^{N-N_{cens}} e^{-\lambda\left(\sum_{i=1}^{N-N_{cens}} y_i + CN_{cens}\right)}$$

which is Gamma $\left(N+1, \sum_{i=1}^{N-N_{cens}} y_i + CN_{cens}\right)$. In one run of the example model with $N = 100$ and $C = 250$, I got $N_{cens} = 30$ and $\sum_{i=1}^{N-N_{cens}} y_i = 6{,}455.752$. Figure 8 shows that the Gamma(101, 13,955.752) density curve matches the posterior draws. The model was working fine; the data lead to a very precise posterior distribution.

Debugging these examples was a fantastic way to learn how to use Stan. I had to go through the language reference and get to know the different data types and look up the cryptic function names. (It turns out that `increment_log_prob` does exactly what its name says, but it was good to learn that `exponential_ccdf_log` gives $\log(Pr(Y > y|\beta))$ for $Y \sim \text{Expon}(\beta)$.)

I decided that I like Stan very much. The simulations run quickly and I appreciate the flexibility of directly incrementing the log probability. My big question is, when there are many ways to do one simple thing (as in the Normal example), how do I decide which is best?

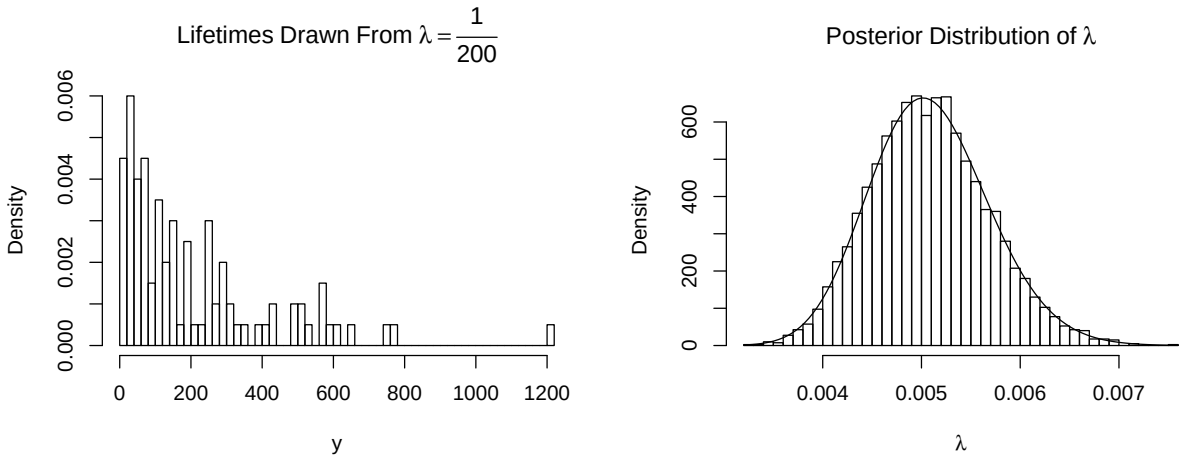

Figure 8: Left: 100 Lightbulb lifetimes drawn from Expon $\left(\frac{1}{200}\right)$. Right: 4,000 posterior draws of λ with the analytical density curve.