Lecture 6

Estimation vs. Approximation

There are a few key elements of a Bayesian data analysis:

1. Model Specification:

2. Prior Specification:

Once these are specified and the data have been gathered, the posterior distribution $p(\theta|y_1, \dots, y_n)$ is fully determined. It is exactly:

$$p(\theta|y_1, \dots, y_n) = \frac{p(\theta)p(y_1, \dots, y_n|\theta)}{p(y_1, \dots, y_n)}$$

3. Posterior Summary: A description of the posterior distribution $p(\theta|y_1, \dots, y_n)$,

For most models we have discussed thus far, $p(\theta|y_1, \dots, y_n)$ is known in closed form or easy to sample from using Monte Carlo procedures. However, in more sophisticated settings, $p(\theta|y_1, \dots, y_n)$ is complicated, and hard to write down or sample from. In these cases, we study $p(\theta|y_1, \dots, y_n)$ by looking at MCMC samples.

Thus, Monte Carlo and MCMC sampling algorithms:

For example, if we have Monte Carlo samples $\theta^{(1)}, \dots, \theta^{(J)}$ that are approximate draws from $p(\theta|y_1, dots, y_n)$, then these sample help describe $p(\theta|y_1, \dots, y_n)$, for example:

- $\int g(\theta)p(\theta|y_1, \dots, y_n)d\theta \approx \frac{1}{J}\sum_j^J g(\theta^{(j)})$

- $\int_\infty^c p(\theta|y_1, \dots, y_n)d\theta = Pr(\theta \leq c|y_1, \dots, y_n) \approx \frac{1}{J}\sum \delta(\theta^{(j)} \leq c)$

To keep the distinction between *estimation* and *approximation* clear, commonly *estimation* is used do describe how we use $p(\phi|y_1, \dots, y_n)$ to make inferences about ϕ and

MCMC Diagnostics

A useful way to think about an MCMC sampler is that there is a *particle* moving through and exploring the parameter space. For each region, or set, A the particle needs to spend time proportional to the target probability, $\int_A p(\phi) d\phi$ Consider an example with three modes and denote these three modes as A_1, A_2, A_3. Assume that A_2 is substantantially less than A_1 and A_3.

Given the weights on the mixture components the particle should spend more time in A_1 and A_3 than A_2. However, if the particle was initialized in A_2 we'd hope that the number of iterations are large enough that:

- The technical term associated with item 1 is *stationarity* which means the chain has converged to the target distribution. For the models we have seen thus far, convergence happens quite rapidly, but we will look at this in more depth later on.

- The second item is focused on the speed the particle moves through the target distribution, this is referred to as *mixing.* An independent sampler like the Monte Carlo procedures we have seen have perfect mixing as each sample is independently drawn from the target distribution. The MCMC samples can be highly correlated and tend to get stuck in certain regions of the space.

People often quantify mixing properties of MCMC samples using the idea of effective sample size. To understand this, first consider the variance of independent Monte Carlo samples:

$$Var_{MC}[\bar{\theta}] = \frac{Var(\theta)}{J}$$

where $\bar{\phi} = \sum_{j=1}^{J} \phi^{(j)}/J$.

The Monte Carlo variance is controlled by the number of samples obtained from the algorithm. In a MCMC setting, consecutive samples $\theta^{(j)}$ and $\theta^{(j+1)}$ are not independent, rather they are usually positively correlated.

Once stationarity has been acheived, the variance of the MCMC algorithm can be expressed as:

$$Var_{MCMC}[\theta] = \dots = Var_{MC}[\bar{\theta}] + \frac{1}{J^2} \sum_{j \neq k} \left[(\theta^{(j)} - \theta_0)(\theta^{(k)} - \theta_0) \right]$$

where θ_0 is the true value of the integral, typically $E[\theta]$.

Now if two consecutive samples are highly correlated the variance of the estimator will be much larger than that of an Monte Carlo procedure with the same number of iterations. This is captured in the idea of the **effective sample**. The effective sample size is computed such that:

$$Var_{MCMC}[\bar{\theta}] = \frac{Var(\theta)}{S_{eff}}$$

where S_{eff} can be interpreted as the number of independent Monte Carlo samples necessary to give the same precision as the MCMC samples. Note that the R function `effectiveSize` in the `coda` package will calculate the effective sample size of MCMC output.

We will talk more about MCMC diagnostics after introducing the Metropolis-Hastings algorithm later in class, but the general procedure is:

An easy solution, especially in the context of Gibbs Sampling is to look at trace plots and histograms of marginal posterior distributions. In conjunction with ESS (Effective Sample Size) calculations this usually gives a good sense of convergence. In other situations, combining visuals displays (trace plots) with other statisics Gelman's R statistic.

The big picture idea with MCMC, is that we want to guarantee that our algorithm has:

Model Checking with the Posterior Predictive Distribution

The posterior predictive distribution,

The posterior predictive distribution should *look like* the data itself.

Consider a dataset that looks at the cell loss from the use of e-cigarettes. Cell loss is quantified as viability, the proportion of functional cells.

```
library(readr)
ecig_viability <- c(85.9, 70.2, 40.5, 47.9, 31.3, 26.5, 82.9, 89.4, 24.6, 58.5, 68.5, 19.0, 42.6,
                    33.0, 16.4, 21.5, 54.0, 29.7, 25.0, 36.8, 18.0, 16.5, 10.9, 13.1, 20.7,  1.3,
                    27.1,  1.4, 0.0, 36.4, 1.2,  0.8,  1.0,  0.0,  0.0,  0.0)
```

We can use a Gibbs sampler to fit this model with a normal sampling model (Note this could be used to build up to an ANOVA model)

```
######### First Gibbs Sampler
y <- ecig_viability
mean.y <- mean(ecig_viability)
var.y <- var(ecig_viability)
num.obs <- length(ecig_viability)
library(LearnBayes) # for rigamma
### initialize vectors and set starting values and priors
num.sims <- 1000
theta_samples <- rep(1, num.sims)
sigmasq_samples <- rep(1, num.sims)
## Hyperparameters
# theta ~n (mu.0, tausq.0)
mu.0 <- 0
tausq.0 <- 10000

#sigmasq ~ IG(nu.0/2, nu.0 * sigmasq.0 / 2)
nu.0 <- .01
sigmasq.0 <- 1
for (i in 2:num.sims){
  # sample theta from full conditional
  mu.n <- (mu.0 / tausq.0 + num.obs * mean.y / sigmasq_samples[i-1]) /
    (1 / tausq.0 + num.obs / sigmasq_samples[i-1] )
  tausq.n <- 1 / (1/tausq.0 + num.obs / sigmasq_samples[i-1])
  theta_samples[i] <- rnorm(1,mu.n,sqrt(tausq.n))

  # sample (1/sigma.sq) from full conditional
  nu.n <- nu.0 + num.obs
  sigmasq.n.theta <- 1/nu.n*(nu.0*sigmasq.0 + sum((y - theta_samples[i])^2))
  sigmasq_samples[i] <- rigamma(1,nu.n/2,nu.n*sigmasq.n.theta/2)
}

#remove burnin
burn_in <- 50
theta_posterior <- theta_samples[-c(1:burn_in)]
```

We can summarize our posterior samples for the mean cell loss and compare the results to a t-test.

```r
library(coda)
summary(as.mcmc(theta_posterior))
```

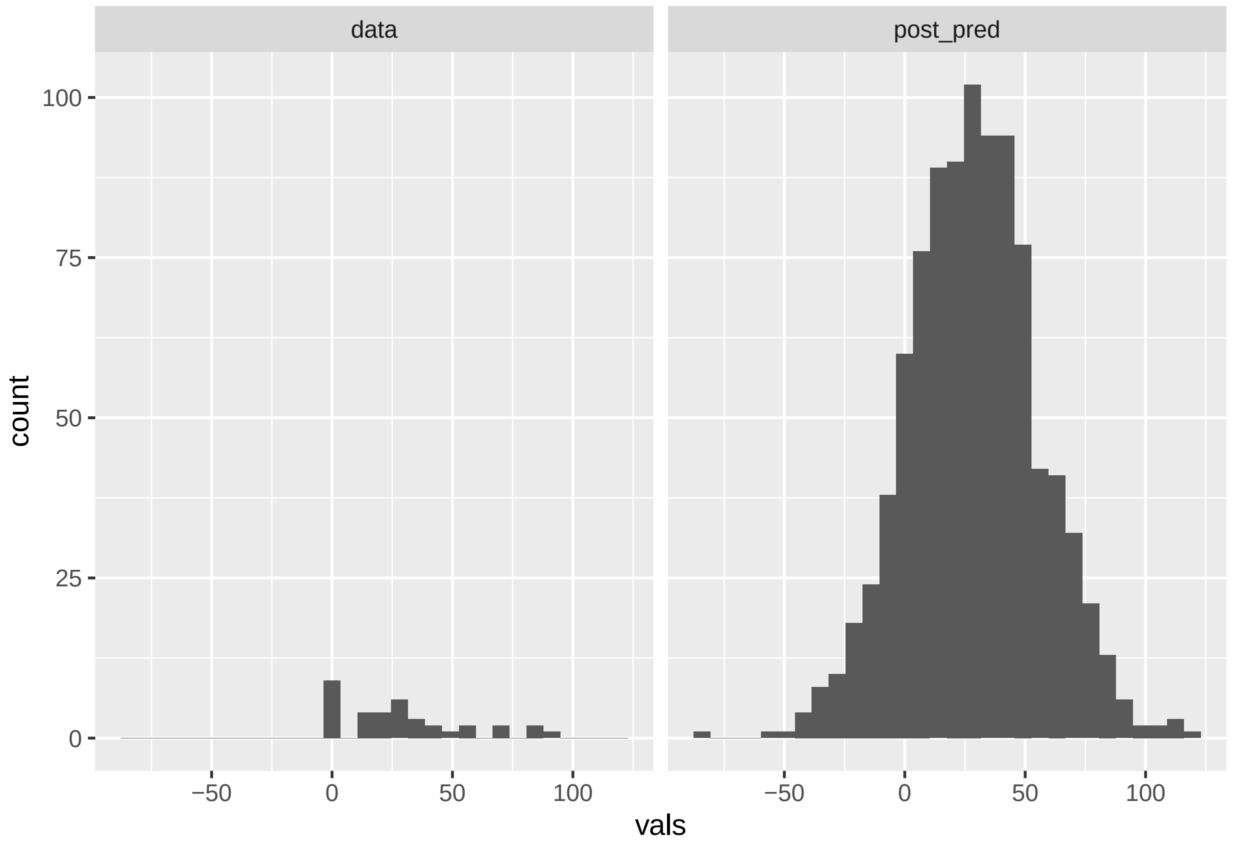
```
##
## Iterations = 1:950
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 950
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##        29.2810         4.5132         0.1464         0.1464
##
## 2. Quantiles for each variable:
##
##  2.5%   25%   50%   75% 97.5%
## 20.33 26.24 29.30 32.28 37.94
```

```r
t.test(ecig_viability)
```

```
##
##  One Sample t-test
##
## data:  ecig_viability
## t = 6.7155, df = 35, p-value = 8.926e-08
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  20.39992 38.07786
## sample estimates:
## mean of x
##  29.23889
```

However, we also want to consider a posterior predictive distribution to evaluate our model fit.

```r
# use samples from posterior to create new data points with the sampling model
library(dplyr)
y_star <- rnorm(num.sims - burn_in, mean = theta_posterior, sd = sqrt(sigmasq_samples[-c(1:burn_in)]))
library(ggplot2)
tibble(vals = c(y_star, y), type = c(rep('post_pred', num.sims - burn_in), rep('data',num.obs))) %>%
  ggplot(aes(x=vals)) + geom_histogram() + facet_grid(.~type)
```


data
post_pred
count
vals