Sta 360: HW #5

Matthew Murray

3/25/2021

Question 1

a

Sequence of samples $\theta_{[1]} \dots \theta_{[s]}$ from $[\theta]$

Estimating integral $\zeta = \mathrm{E}[\mathrm{g}(\theta)]$ with sample estimator $\hat{\zeta} = \sum_{[s=1]}^{S} \frac{g(\theta_{[s]})}{S}$

(From Ch.6 Notes):

$\mathrm{Var}_{MC}[\hat{\zeta_s}] = \frac{Var[g(\theta)]}{S}$

b

$\mathrm{Var}_{Gibbs}[\hat{\zeta}_S] = \mathrm{E}[(\hat{\zeta}_s - \zeta)^2]$

$= \mathrm{E}[(\frac{1}{S}\sum_{s=1}^{S} g(\theta_{[s]}) - \zeta)^2]$

$= \mathrm{E}[(\frac{1}{S}\sum_{s=1}^{S} [g(\theta_{[s]}) - \zeta])^2]$

$= \frac{1}{S^2}\ \mathrm{E}[\sum_{s=1}^{S}\sum_{t=1}^{S}(g\theta_{[s]} - \zeta)(g(t)_{[t]} - \zeta)]$

$= \frac{1}{S^2}\ \mathrm{E}[\sum_{s=1}^{S}(g(\theta_{[s]}) - \zeta)^2 + \sum_{s=1}^{S}\sum_{t=1,t\neq s}^{S}(g\theta_{[s]} - \zeta)(g(t)_{[t]} - \zeta)]$

$= \frac{Var[g(\theta)]}{S} + \frac{1}{S^2}\sum_{s=1}^{S}\sum_{t=1,t\neq s}^{S} E[(g\theta_{[s]} - \zeta)(g(t)_{[t]} - \zeta)]$

$= \mathrm{Var}_{MC}[\hat{\zeta_s}] + \frac{1}{S^2}\sum_{s=1}^{S}\sum_{t=1,t\neq s}^{S} E[(g\theta_{[s]} - \zeta)(g(t)_{[t]} - \zeta)]$

Note: $\mathrm{Cov}(\mathrm{g}(\theta_{[s]}), \mathrm{g}(\theta_{[t]})) = E[(g\theta_{[s]} - \zeta)(g(t)_{[t]} - \zeta)]$

Let $\gamma_{s,t} = \mathrm{Cov}(\mathrm{g}(\theta_{[s]}), \mathrm{g}(\theta_{[t]}))$

$= \mathrm{Var}_{MC}[\hat{\zeta_s}] + \frac{1}{S^2}\sum_{s=1}^{S}\sum_{t=s,s\neq t}^{S}\gamma_{s,t}$

Assuming autocorrelation, this variance is larger than that in (a). This autocorrelation stems from the fact that there is "stickiness", or in other words, correlation between consecutive values in the sampling chain with Gibbs sampling, whereas an independent MC sampler (as seen in part(a)), has perfect mixing, with zero autocorrelation. In summation, we can see that autocorrelation between samples causes the Gibbs sampler to have an inflated variance.

c

```
#### Mixture normal parameters
mu<-c(-3,0,3) # mean vector
s2<-c(.33,.33,.33) # variance vector
w<-c(.45,.1,.45) # weights
```

```
#Monte Carlo Sampling
set.seed(314)
S <- 10000
zeta.mc = sapply(seq(1,S,1),function(x){
  d<-sample(1:3,S, prob=w,replace=TRUE)
  pred = rnorm(S,mu[d],sqrt(s2[d]))
  prob = (pred <= 3)})
sd.zeta.mc <- sd(zeta.mc)

width <- 2 * (1.96 * (sd.zeta.mc/sqrt(10000)))
width
```

```
## [1] 0.01636698
```

The width of the 95% confidence interval for the Gibbs estimate of ζ is 0.0164.

```
#### MCMC: Gibbs sampling
set.seed(314)
th<-0 # initial value for X
S<-10000
THD.MCMC<-matrix(NA,nrow=S,ncol=2)
for(s in 1:S) {
  d<-sample(1:3 ,1,prob = w*dnorm(th,mu,sqrt(s2))) #sampling full conditional d|th
  th<-rnorm(1,mu[d],sqrt(s2[d])) #sampling full conditional th|d
  THD.MCMC[s,]<- c(th,d)
}

sd.zeta.gibbs <- sd(THD.MCMC[,1])
ESS <- effectiveSize(THD.MCMC[,1])

width <- 2 * (1.96 * (sd.zeta.gibbs/sqrt(ESS)))
width
```

```
##     var1
## 2.548164
```

The width of the 95% confidence interval for the Gibbs sampling estimate of ζ is 0.3550.

The confidence interval using Monte Carlo is narrower due to the fact that Monte Carlo estimates often have lower variance, and therefore, more precise estimates.

The confidence interval using Monte Carlo is narrower due to the fact that Monte Carlo estimates often have lower variance, and therefore, more precise estimates.

d

$\hat{\zeta_s^{thin}} = \frac{1}{S/10} \sum_{s=1}^{(S/10)} g(\theta_{[10s]})$

$\text{Var}_{Gibbs}[\hat{\zeta}_s^{thin}] = \text{E}[(\hat{\zeta}_s^{thin} - \zeta)^2]$

$= \text{E}[(\frac{1}{S/10}\sum_{s=1}^{S/10} g(\theta_{[10s]}) - \zeta)^2]$

$= \text{E}[(\frac{1}{S/10}\sum_{s=1}^{S/10} [g(\theta_{[10s]}) - \zeta])^2]$

$= \frac{1}{S^2/100}\ \text{E}[\sum_{s=1}^{S/10}\sum_{t=1}^{S/10}(g(\theta_{[10s]}) - \zeta)(g(\theta_{[10t]}) - \zeta)]$

$= \frac{1}{S^2/100}\ \text{E}[\sum_{s=1}^{S/10}(g(\theta_{[10s]}) - \zeta)^2 + \sum_{s=1}^{S/10}\sum_{t=1,t\neq s}^{S/10}(g(t_{[10s]}) - \zeta)(g(\theta_{[10t]}) - \zeta)]$

$\text{Var}_{Gibbs}[\zeta_s^{\hat{thin}}] = \text{Var}_{MC}[\zeta_s^{\hat{thin}}] + \frac{100}{S^2}\sum_{s=1}^{S/10}\sum_{t=1,t\neq s}^{S/10} \text{Cov}(g(\theta_{[10s]}), g(\theta_{[10t]}))$

Let $\gamma_{10s,10t} = \text{Cov}(g(\theta_{[10s]}), g(\theta_{[10t]}))$

$\text{Var}_{Gibbs}[\zeta_s^{\hat{thin}}] = \text{Var}_{MC}[\zeta_s^{\hat{thin}}] + \frac{100}{S^2}\sum_{s=1}^{S/10}\sum_{t=1,t\neq s}^{S/10} \gamma_{10s,10t}$

By reducing our set of samples, the samples will be less correlated with each other due to the fact that we are taking every 10th sample. This reduction is autocorrelation between samples therefore decreases the variance of our Gibbs sampling estimator. The downside of thinning is that we have less samples. In other words, there is a tradeoff between decreasing sample size and decreasing autocorrelation (and therefore, decreasing variance).

e

One advantage of using thinning for Gibbs sampling is that by throwing away many iterations of the Markov Chain, we are reducing autocorrelation between samples, or as the textbook notes, the "stickiness" in one's MCMC samples. Furthermore, thinning can also make one's code less computationally-intensive and improve run times.

One disadvantage of using thinning for Gibbs sampling is that by throwing away chain values, you lose precision in your estimate of the posterior distribution, which will likely give you wider (less precise) confidence intervals and larger variances.

One practical scenario where thinning may be preferable is if one is looking at data for a sports team's performance over the course of a season. In this scenario, there is a lot of autocorrelation between MCMC samples due to extraneous factors like strength of opponent, injuries, coaching changes, etc. Because of this high autocorrelation, one will need a prodigious amount of samples (for this instance, over 10,000,000 samples, or games) to achieve the same level of accuracy as the "gold standard" of Monte Carlo, which, quite frankly, is unfeasible. By using thinning in this instance, one would in effect decrease the autocorrelation between samples and decrease the run time of his or her code.

Question 2

$Y_i \sim Pareto(m, \alpha), i = 1, \ldots, n$

Priors:

$[\alpha] \sim$ Gamma(a,b)

$[\text{m}] \propto \frac{1}{m}$

a

$\text{Z} = \frac{1}{Y} \sim InvPareto(m, \alpha)$

Transformation Theorem: $f_z(Z) = f_y(1/Z)|\frac{d}{dz}z^{-1}|$

$= \frac{\alpha m^\alpha}{(\frac{1}{z})^{\alpha+1}} z^{-2} \mathbb{1}(\frac{1}{z} \geq m)$

$= \frac{\alpha m^\alpha z^{(\alpha+1)}}{z^2} \mathbb{1}(\frac{1}{z} \geq m) = \alpha m^\alpha z^{\alpha-1} \mathbb{1}(\frac{1}{z} \geq m)$

$[z] \propto z^{\alpha-1} \mathbb{1}(\frac{1}{z} \geq m)$

To sample from this inverse-Pareto distribution, you would first initialize your value for α and then estimate your values of m. Thereafter, given your value of m, you would then find your new value of α and repeat this process for a set number of times.

b

$Y \sim Pareto(m, \alpha)$

$[\alpha] \sim Gamma(a, b)$

$[m] \sim \frac{1}{m}$

$[\alpha \mid m, y_1, \ldots, y_n] \propto [\alpha, m \mid y_1, \ldots, y_n]/[m \mid y_1, \ldots, y_n]$

$\propto [\alpha, m \mid y_1, \ldots, y_n]$

$\propto [y_1, \ldots, y_n \mid \alpha, m][\alpha, m]$

$\propto \prod_{i=1}^n \frac{\alpha m^\alpha}{y_i^{\alpha+1}} \alpha^{a-1} e^{-b\alpha}$

$\propto \alpha^{a+n-1} e^{-b\alpha} \prod_{i=1}^n (\frac{m}{y_i})^\alpha$

$= \alpha^{a+n-1} e^{-b\alpha} e^{-a \sum_{i=1}^n ln(\frac{y_i}{m})}$

$= \alpha^{a+n-1} e^{-(b+\sum_{i=1}^n ln(\frac{y_i}{m}))}$

$\sim Gamma(a+n, b+\sum_{i=1}^n ln(\frac{y_i}{m}))$

c

$[m \mid \alpha, y_1, \ldots, y_n] \propto [m, \alpha \mid y_1, \ldots, y_n]/[\alpha \mid y_1, \ldots, y_n]$

$\propto [m, \alpha \mid y_1, \ldots, y_n]$

$\propto [y_1, \ldots, y_n \mid m, \alpha][m]$

$= \prod_{i=1}^n (\alpha m^\alpha y_i^{\alpha-1}) \frac{1}{m} \mathbb{1}(y_i \geq m)$

$\propto m^{n\alpha-1} \mathbb{1}(min(y_i) \geq m)$

$= m^{n\alpha-1} \mathbb{1}(\frac{1}{m} \geq \frac{1}{min(y_i)})$

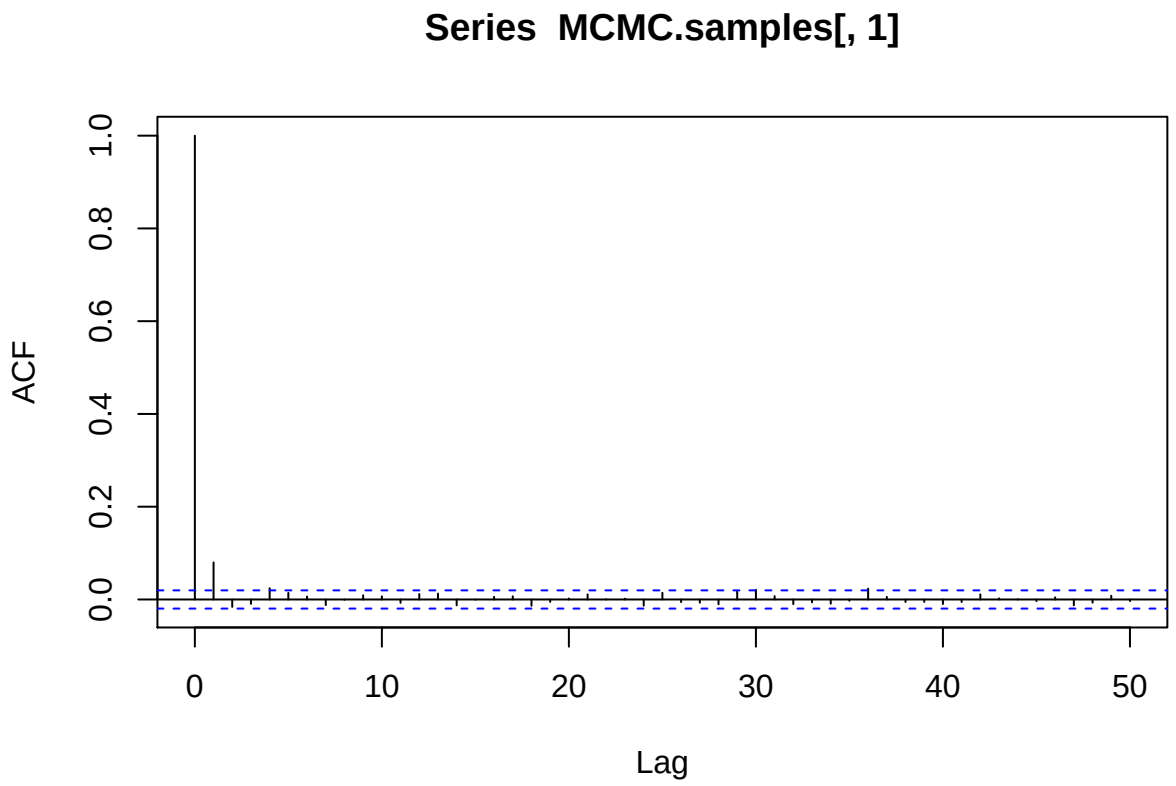
d

```
claims <- read.csv("claims.dat")
```

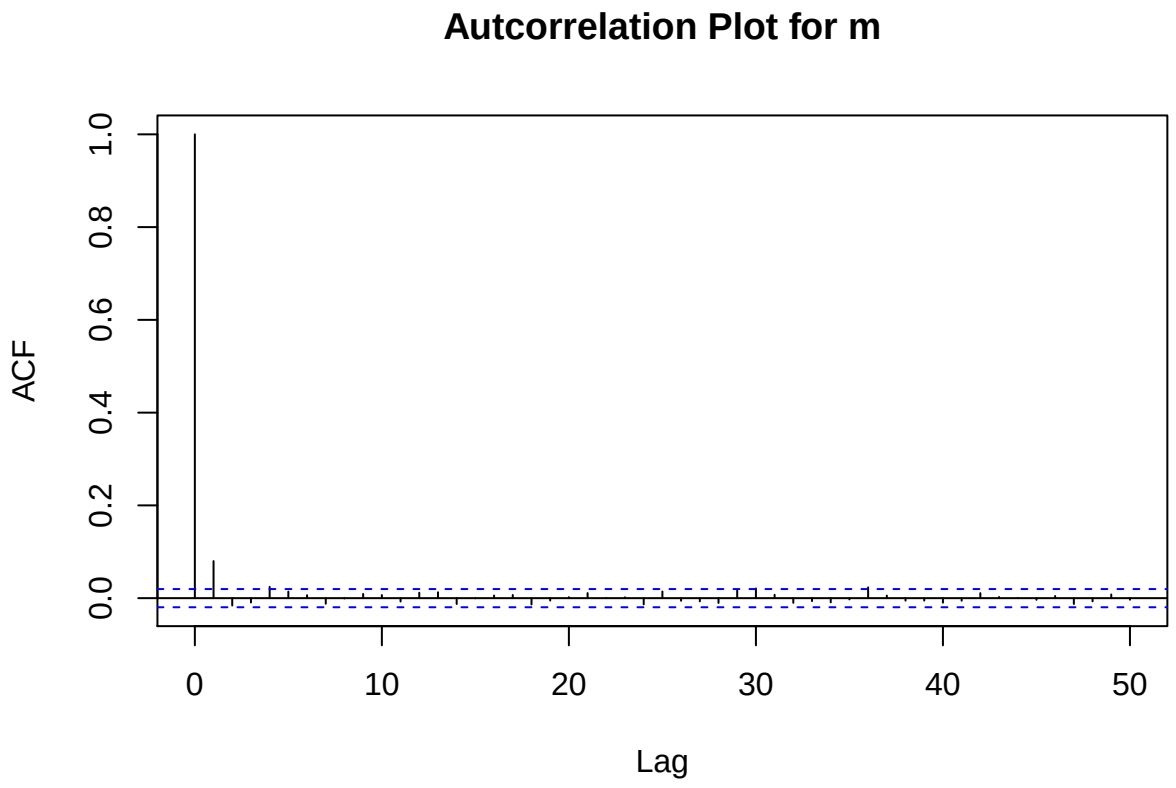
```
set.seed(314)
y <- claims$X3440.08
n <- length(y)

a <- 0.01
b <- 0.01
```

```r
S <- 10000
alpha <- 0.1
MCMC.samples = matrix(NA, nrow = S, ncol = 2)
for (s in 1:S){
  m <- 1/rpareto(1, 1/min(y), n*alpha)
  alpha <- rgamma(1, a + n, b + sum(log(y/m)))
  MCMC.samples[s,] <- c(m, alpha)}
```

```r
#autocorrelation plot for m
autocorrelation.m <- acf(MCMC.samples[,1], lag.max = 50)
```



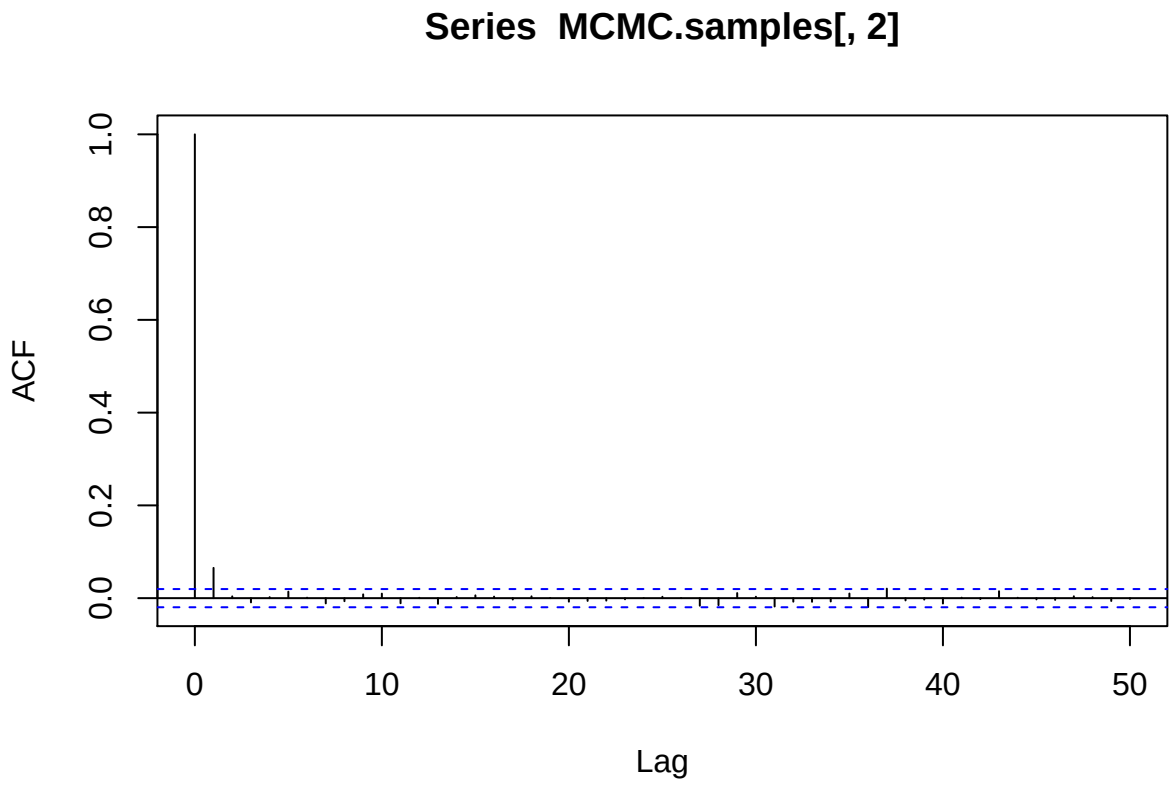

```r
plot(autocorrelation.m, main = "Autcorrelation Plot for m")
```



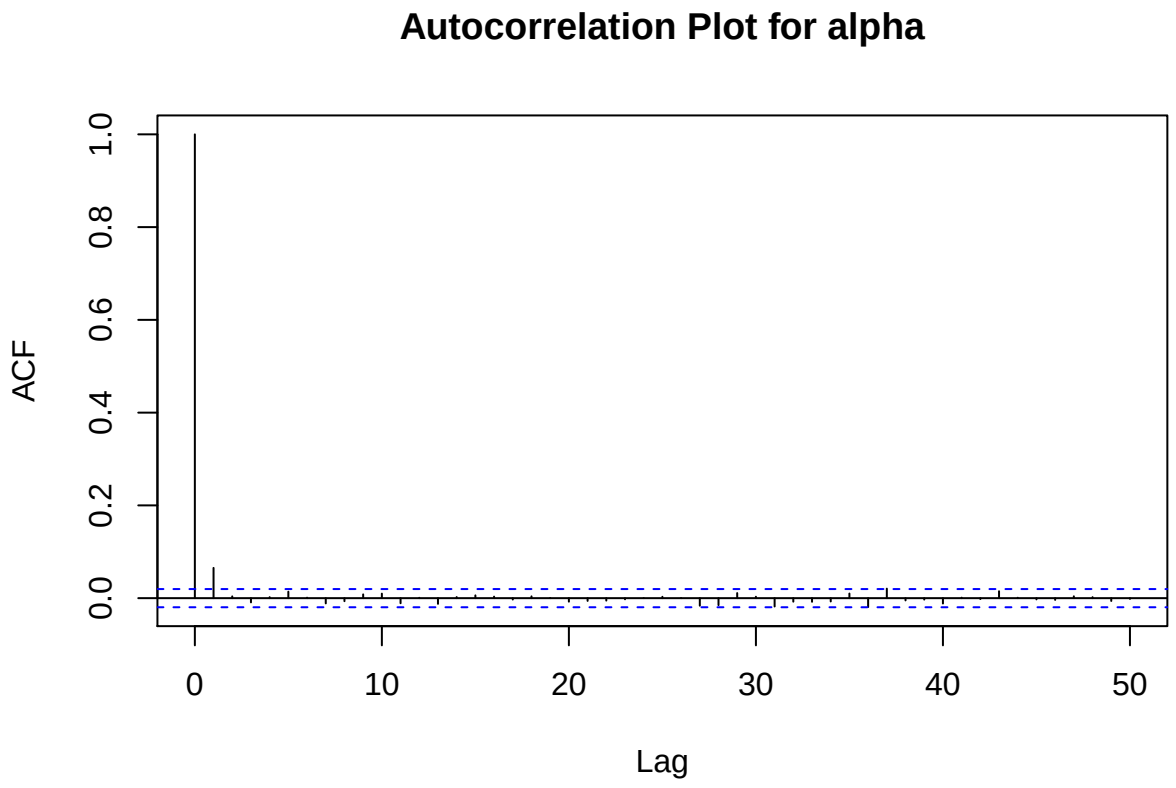

```
#effective sample size for m
effectiveSize(MCMC.samples[,1])
```

```
##    var1
## 8573.79
```

```
#autocorrelation plot for alpha
autocorrelation.a <- acf(MCMC.samples[,2], lag.max = 50)
```




```
plot(autocorrelation.a, main = "Autocorrelation Plot for alpha")
```




```
#effective sample size for alpha
effectiveSize(MCMC.samples[,2])
```

```
##     var1
## 8773.736
```

e

```
S <- 10000
N <- rpois(1,5)

claim.sample.sums = matrix(NA, nrow = S, ncol = 2)

for(s in 1:S){
  y <- c(rpareto(N, (MCMC.samples[s,1]), (MCMC.samples[s,2])))
  x <- sum(y)
  claim.sample.sums[s,] <- c(x, s)
}

median(claim.sample.sums[,1])
```

```
## [1] 14952.63
```

```
quantile(claim.sample.sums[,1])
```

```
##           0%          25%          50%          75%         100%
## 3.223706e+03 9.059707e+03 1.495263e+04 3.440813e+04 1.477261e+10
```

Median = $14,952

You should charge this group $34,408 in premiums to ensure that you make a profit 75% of the time.

Question 3

a

This mixture sampling model will be better than the normal sampling model when there is heterogeneity in the data, or in other words, when the data comes from different samples, and therefore, different distributions. Furthermore, mixture distributions are useful when the data is located in different clusters, creating a multimodal distribution.

b

$[X_i = 1 \mid p, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2, Y_i]$

$\propto [Y_i, X_i = 1, p, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2]$

$= [X_i = 1 \mid Y_i, p, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2][Y_i, p, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2]$

$\propto [X_i = 1 \mid Y_i] = \frac{P(Y_i|X_i=1)P(X_i=1)}{P(X_i=1)P(Y_i|X_i=1)+P(X_i=0)P(Y_i|X_i=0)}$

$= \frac{N(\theta_2,\sigma_2^2)*p}{p*N(\theta_2,\sigma_2^2)+(1-p)N(\theta_1,\sigma_1^2)}$

Full Conditional for $X_i : \sim \mathbf{Bernoulli}(\frac{\mathbf{N(\theta_2,\sigma_2^2)*p}}{\mathbf{p*N(\theta_2,\sigma_2^2)+(1-p)N(\theta_1,\sigma_1^2)}})$

$[p \mid X_i, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2, data]$

$\propto [Y_i, X_i, p, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2]$

$= [p \mid Y_i, X_i, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2][Y_i, X_i, \theta_1, \theta_2, \sigma_1^2, \sigma_2^2]$

$\propto [p \mid X_i] \propto [X_i \mid p][p] = Bern(p) * Beta(a, b)$

$= p(1-p)p^{a-1}(1-p)^{b-1}$

$= p^a(1-p)^b$

$\sim \mathbf{Beta(a+1, b+1)}$

$[\theta_1 \mid X_i, p, \theta_2, \sigma_1^2, \sigma_2^2, Y_i]$

$\propto \prod_{i=1}^n [x_i \mid p][p][\theta_1][\theta_2][\sigma_1^2][\sigma_2^2][Y_i]$

$\prod_{i=1}^n [\theta_1][Y_i \mid X_i = 0]$

$\propto N(\mu, \tau^2) * N(\theta_1, \sigma_1^2) = \mathbf{N}(\mu^*, \tau^{*\mathbf{2}})$ (from Ch.5 Notes)

where $\tau^{*2} = (\frac{n}{\sigma_1^2} + \frac{1}{\tau^2})^{-1}$ and $\mu^* = \tau^{*2}(\frac{\sum_{i=1}^n y_i}{\sigma_1^2} + \frac{\mu}{\tau^2})$

By similar logic, the full conditional for θ_2 is:

$$\propto N(\mu, \tau^2) * N(\theta_1, \sigma_1^2) = \mathbf{N}(\mu^*, \tau^{*\mathbf{2}})$$

where $\tau^{*2} = (\frac{n}{\sigma_2^2} + \frac{1}{\tau^2})^{-1}$ and $\mu^* = \tau^{*2}(\frac{\sum_{i=1}^n y_i}{\sigma_2^2} + \frac{\mu}{\tau^2})$

$[\sigma_1^2 \mid X_i, p, \theta_1, \theta_2, \sigma_2^2, Y_i]$

$$\propto \prod_{i=1}^n [x_i \mid p][\theta_1][\theta_2][\sigma_1^2][\sigma_2^2][Y_i]$$

$$\prod_{i=1}^n [\sigma_1^2][Y_i \mid X_i = 0]$$

$$IG(\nu/2, \nu\gamma^2/2) N(\theta_1, \sigma_1^2)$$

$$\propto \mathbf{IG}(\nu^*/\mathbf{2}, \nu^*\gamma^{*\mathbf{2}}(\theta)/\mathbf{2}) \text{ (from Ch.6 Notes)}$$

where $\nu^* = n + \nu$ and $\gamma^{*2} = \nu^{*-1}(\nu\gamma^2 + \sum_{i=1}^n (y_i - \theta_1)^2)$

By similar logic, the full conditional for σ_2^2 is:

$$\propto \mathbf{IG}(\nu^*/\mathbf{2}, \nu^*\gamma^{*\mathbf{2}}(\theta)/\mathbf{2})$$

where $\nu^* = n + \nu$ and $\gamma^{*2} = \nu^{*-1}(\nu\gamma^2 + \sum_{i=1}^n (y_i - \theta_2)^2)$

c

```
glucose <- read.csv("glucose.dat")
```

```
set.seed(314)

# set prior parameters and hyperparameters
a <- b <- 1
mu <- 120
tau2 <- 200
gamma2 <- 1000
nu <- 10
S <- 10000

#information from dataset
y <- glucose$X86
n <- length(y)
y.bar <- mean(y)
s2 <- var(y)

x <- rbernoulli(n, p = 0.5)
n1 <- sum (x == 0)
n2 <- sum (x == 0)

#initialize values and set up storage matrix
theta.1 <- theta.2 <- 0
sigma2.1 <- sigma2.2 <- 1
THD.MCMC = matrix(NA, nrow = S, ncol = 5)

for (s in 1:S) {

tau2.star = (n1 / sigma2.1 + 1/tau2)^-1
mu.star = tau2.star * (sum(y)/sigma2.1 + mu/tau2)
```

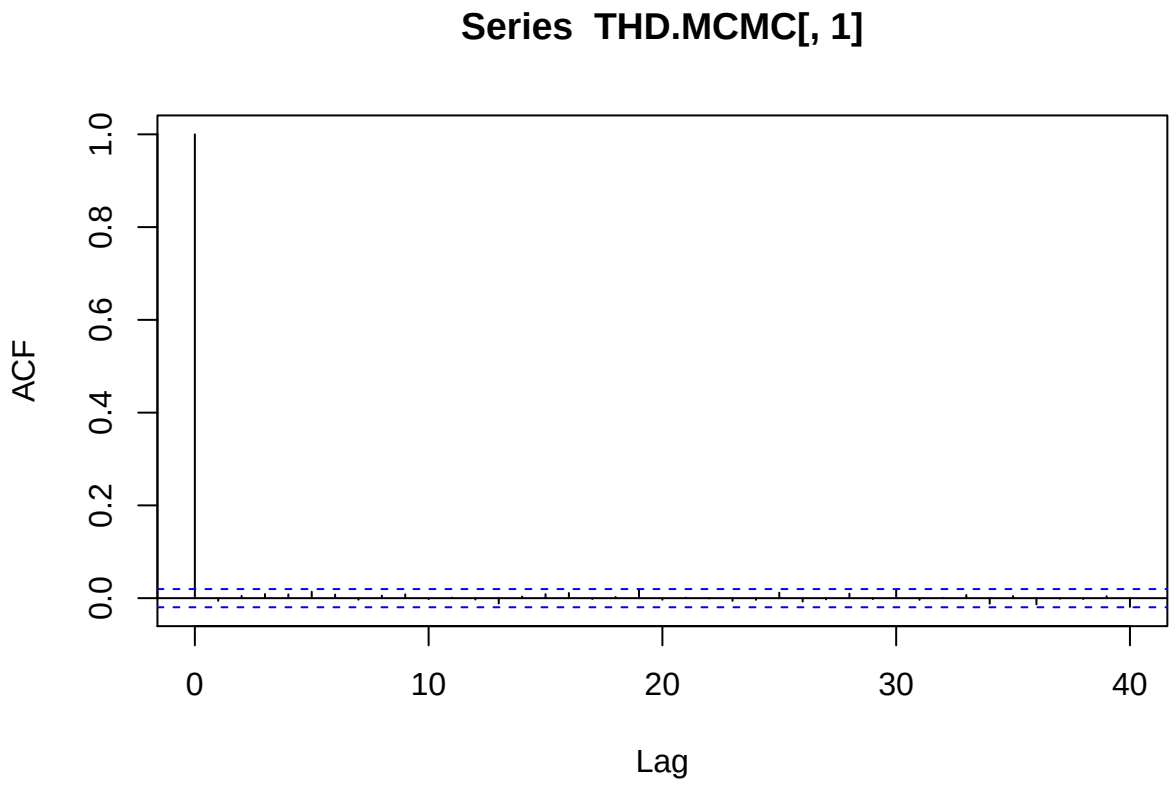
```
theta.1 <- rnorm(1, mu.star, sqrt(tau2.star))

tau2.star = (n2 / sigma2.2 + 1/tau2)^-1
mu.star = tau2.star * (sum(y)/sigma2.2 + mu/tau2)

theta.2 <- rnorm(1, mu.star, sqrt(tau2.star))

nu.star <- nu + n1
sum.of.squares <-  sum(y - theta.1)^2
gamma2.star <- (1/nu.star)*(nu*gamma2 + sum.of.squares)

sigma2.1 = rigamma(1, nu.star/2, (nu.star * gamma2.star)/2)

nu.star <- nu + n2
sum.of.squares <-  sum(y - theta.2)^2
gamma2.star <- (1/nu.star)*(nu*gamma2 + sum.of.squares)

sigma2.2 =  rigamma(1, nu.star/2, (nu.star * gamma2.star)/2)

p <- rbeta(1, a + sum(x), b + n - sum(x))

x.values <- c()
x.lst <- vector(mode = "list", length = S)
for (i in 1:n) {
p.x.i <- ((dnorm(y[i], theta.2, sqrt(sigma2.2)) * p)/((dnorm(y[i], theta.2, sqrt(sigma2.2)) * p) + dnorm
x.values[i] <- rbernoulli(1, p.x.i)
c(list(x.values),x.lst)

}
THD.MCMC[s,] <- c(p, theta.1, theta.2, sigma2.1, sigma2.2)

}
```

c

```
p.effectiveSize <- effectiveSize(THD.MCMC[,1])
p.effectiveSize
```

```
##  var1
## 10000
```

```
p.acf <- acf(THD.MCMC[,1])
```



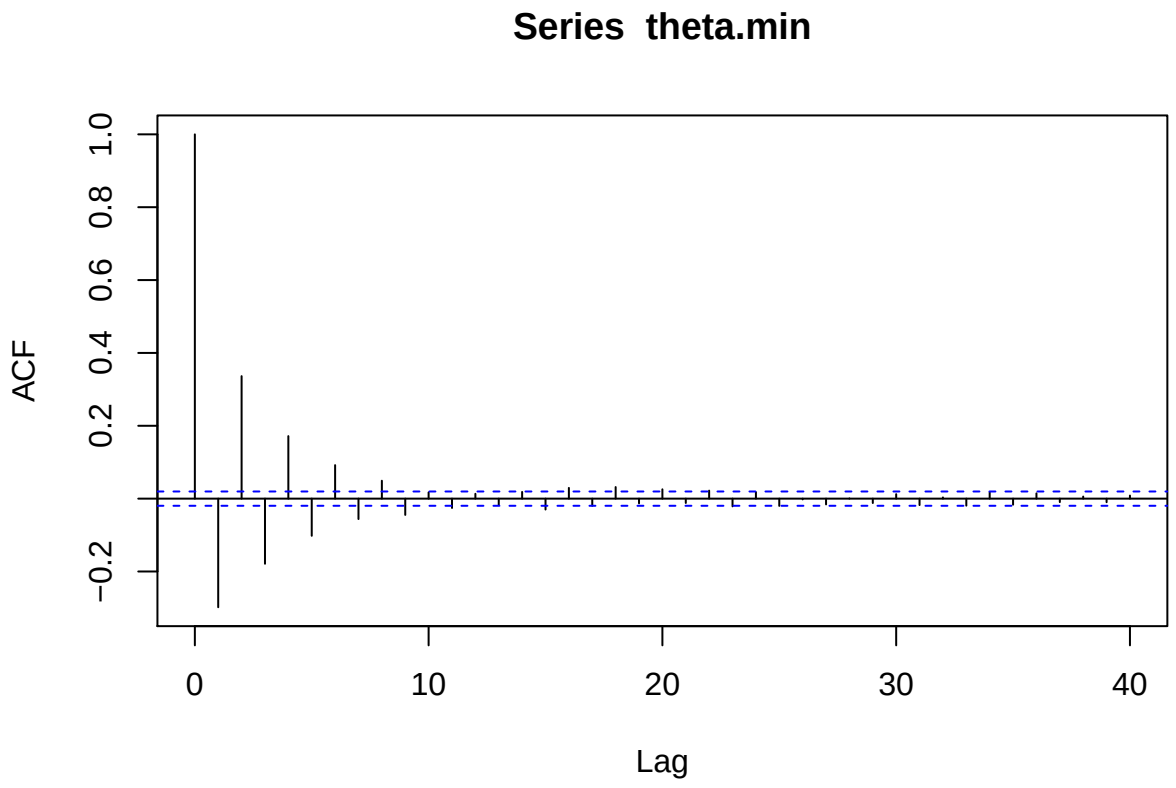

```
p.acf
```

```
##
## Autocorrelations of series 'THD.MCMC[, 1]', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000 -0.006  0.005  0.009  0.008  0.014  0.008 -0.003  0.005  0.008 -0.002
##     11     12     13     14     15     16     17     18     19     20     21
##  0.001 -0.003 -0.011  0.004  0.009  0.011 -0.002  0.002  0.018 -0.003  0.001
##     22     23     24     25     26     27     28     29     30     31     32
## -0.001 -0.006 -0.004  0.012 -0.007 -0.003  0.010 -0.002  0.018 -0.004  0.000
##     33     34     35     36     37     38     39     40
##  0.007 -0.012  0.004 -0.013 -0.001 -0.002  0.004 -0.019
```

```
theta.min = pmin(THD.MCMC[,2], THD.MCMC[,3])
effectiveSize(theta.min)
```

```
##     var1
## 10304.78
```

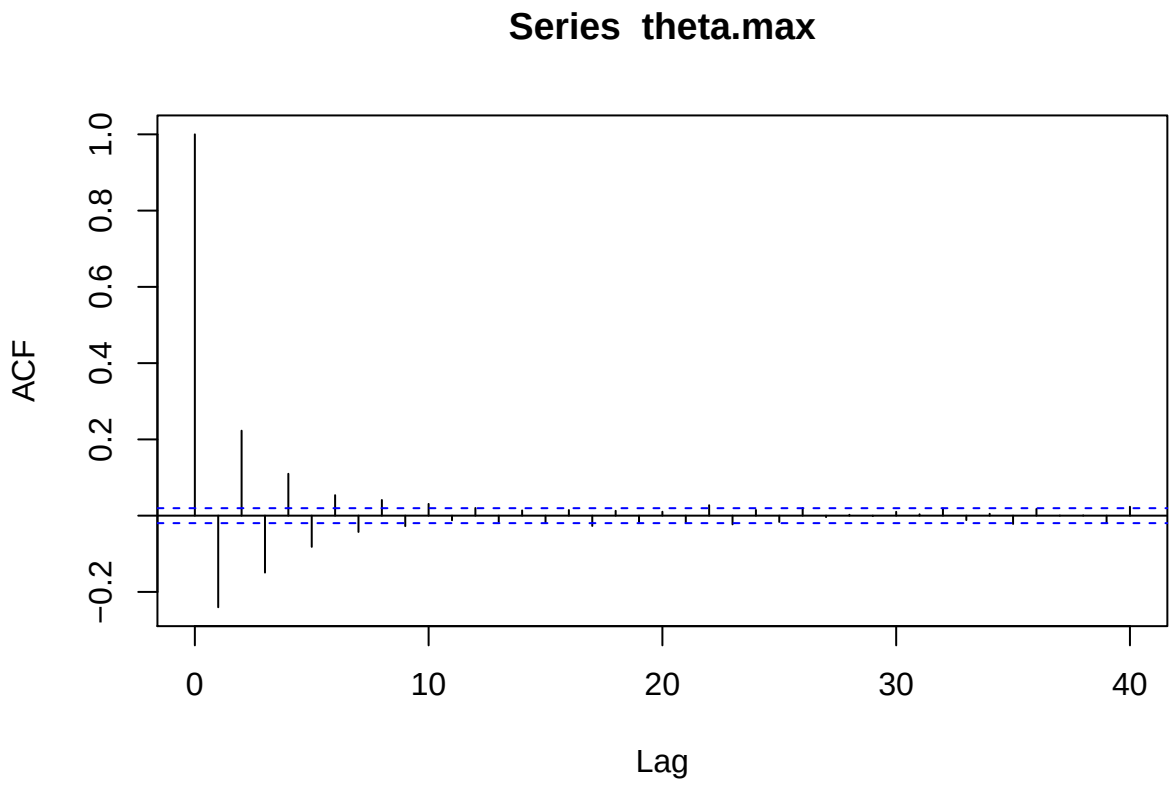
```
acf(theta.min)
```




```
theta.max = pmax(THD.MCMC[,4], THD.MCMC[,5])
effectiveSize(theta.max)
```

```
##     var1
## 12773.99
```

```
acf(theta.max)
```



d

```
set.seed(314)

# set prior parameters and hyperparameters
a <- b <- 1
mu <- 120
tau2 <- 200
gamma2 <- 1000
nu <- 10
S <- 1

#information from dataset
y <- glucose$X86
n <- length(y)
y.bar <- mean(y)
s2 <- var(y)

x <- rbernoulli(n, p = 0.5)
n1 <- sum (x == 0)
n2 <- sum (x == 0)

#initialize values and set up storage matrix
theta.1 <- theta.2 <- 0
```

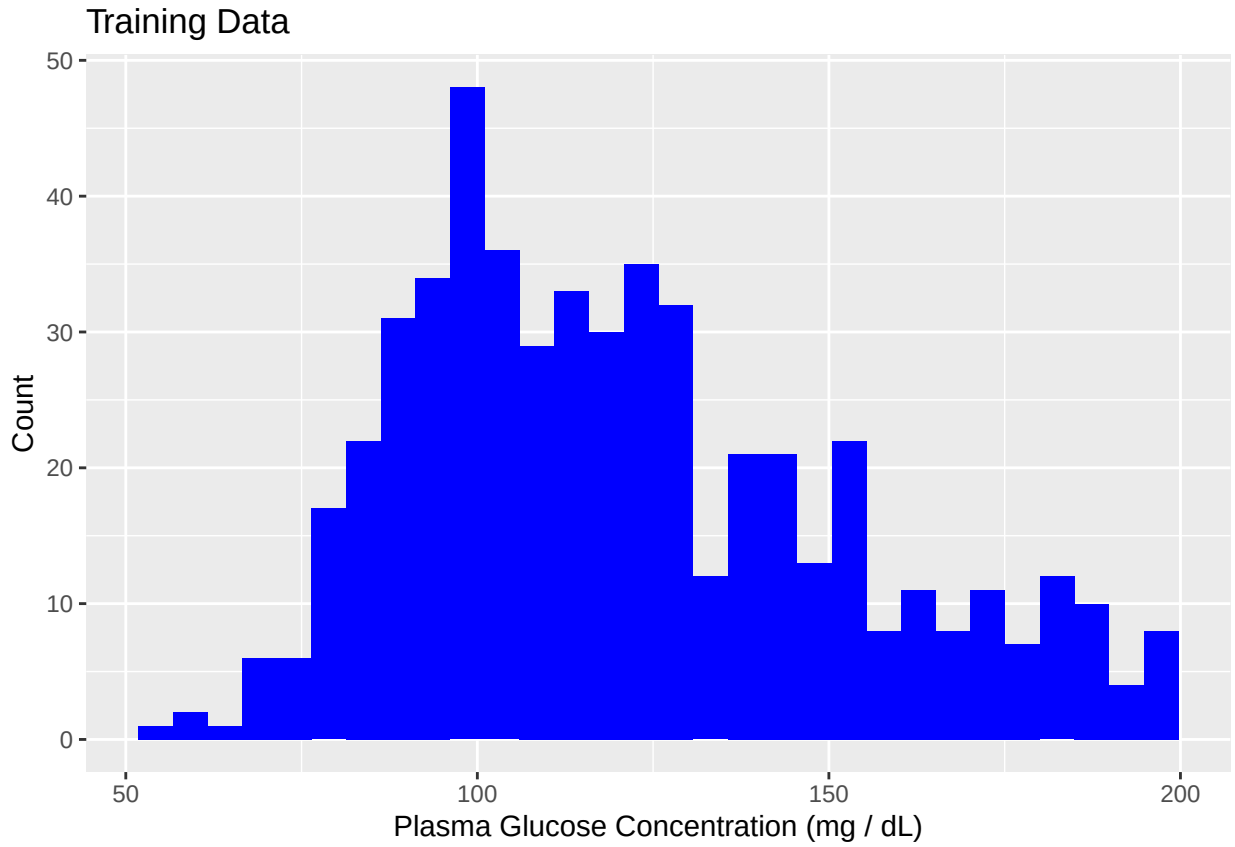
```
sigma2.1 <- sigma2.2 <- 1
THD.MCMC = matrix(NA, nrow = S, ncol = 5)

for (s in 1:S) {

tau2.star = (n1 / sigma2.1 + 1/tau2)^-1
mu.star = tau2.star * (sum(y)/sigma2.1 + mu/tau2)

theta.1 <- rnorm(1, mu.star, sqrt(tau2.star))

tau2.star = (n2 / sigma2.2 + 1/tau2)^-1
mu.star = tau2.star * (sum(y)/sigma2.2 + mu/tau2)

theta.2 <- rnorm(1, mu.star, sqrt(tau2.star))

nu.star <- nu + n1
sum.of.squares <-  sum(y - theta.1)^2
gamma2.star <- (1/nu.star)*(nu*gamma2 + sum.of.squares)

sigma2.1 = rigamma(1, nu.star/2, (nu.star * gamma2.star)/2)

nu.star <- nu + n2
sum.of.squares <-  sum(y - theta.2)^2
gamma2.star <- (1/nu.star)*(nu*gamma2 + sum.of.squares)

sigma2.2 =  rigamma(1, nu.star/2, (nu.star * gamma2.star)/2)

p <- rbeta(1, a + sum(x), b + n - sum(x))

x.values <- c()
for (i in 1:n) {
p.x.i <- ((dnorm(y[i], theta.2, sqrt(sigma2.2)) * p)/((dnorm(y[i], theta.2, sqrt(sigma2.2)) * p) + dnorm
x.values[i] <- rbernoulli(1, p.x.i)

}
THD.MCMC[s,] <- c(p, theta.1, theta.2, sigma2.1, sigma2.2)

}
```

```
set.seed(314)
Z <- 10000

z.values <- c()

for (i in 1:n){
if (x.values[i] == FALSE){
    z <- rnorm(1, theta.1, sqrt(sigma2.1))
   z.values <- append(z.values, z)
}
if (x.values[i] == TRUE){
    z <- rnorm(1, theta.2, sqrt(sigma2.2))
    z.values <- append(z.values, z)
}
```

```
}
```

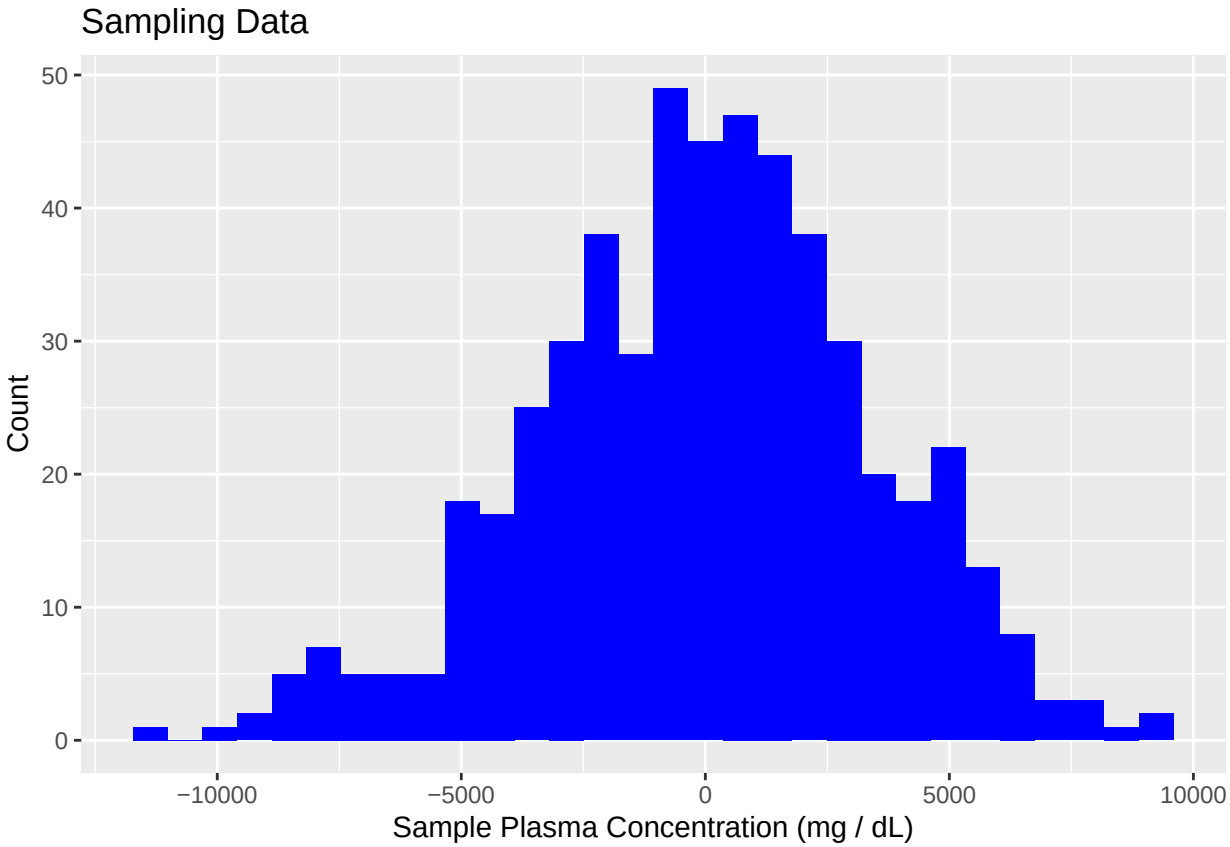
```
#histogram for training data

glucose.df <- data.frame(glucose)

ggplot(glucose.df, aes(x = X86)) + geom_histogram(fill="blue") + labs(title = "Training Data", x = "Pla
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#histogram for predictive samples

z.df <- data.frame(z.values)

ggplot(z.df, aes(x = z.values)) + geom_histogram(fill="blue") + labs(title = "Sampling Data", x = "Samp
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There are definitely discrepancies between these histograms, as the histogram of the predictive samples is multimodal, while that of the training data is just bimodal. Furthermore, the histogram of the predictive samples has a larger range than that of the training data.