STAT 535 – Spring 2024 – Homework 3A

1. The Gibbs sampler can take a long time to converge if the target distribution is multimodal. Suppose we are trying to sample from a density for a parameter θ that is the mixture of three normal densities: θ has density $f(\theta) = 0.45f_1(\theta|\mu = -3, \sigma^2 = 1/3) + 0.10f_2(\theta|\mu = 0, \sigma^2 = 1/3) + 0.45f_3(\theta|\mu = 3, \sigma^2 = 1/3)$, where $f_j(\cdot|\mu, \sigma^2)$ here represents the normal density with mean μ and variance σ^2. The marginal distribution of θ is plotted on the last page.

   Note that we could easily sample from this distribution using ordinary Monte Carlo methods, but consider using Gibbs sampling to sample from it.

   (a) If the indicator $\delta \in \{1, 2, 3\}$, argue that the full conditional distribution for θ is

$$\theta|\delta \sim N(\mu_\delta, \sigma^2_\delta).$$

   What are μ_δ and σ^2_δ here, for $\delta = 1, 2, 3$?

   $\mu_1 = -3$, $\sigma^2_1 = 1/3$, $\mu_2 = 0$, $\sigma^2_2 = 1/3$, $\mu_3 = 3$, $\sigma^2_3 = 1/3$.

   (b) Use Bayes' theorem to show that the full conditional for δ is

$$P[\delta = k|\theta] = \frac{P[\delta = k] \times \text{dnorm}(\theta, \mu_k, \sigma_k)}{\sum_{m=1}^{3} P[\delta = m] \times \text{dnorm}(\theta, \mu_m, \sigma_m)}, \text{ for } k \in \{1, 2, 3\}$$

   where `dnorm` represents the normal density function (R shorthand).

   By Bayes' rule,

$$P[\delta = k|\theta] = \frac{p(\delta = k)p(\theta|k)}{p(\theta)}$$

   and then in the denominator,

$$p(\theta) = \sum_{m=1}^{3} P[\delta = m] \times p(\theta|m) = \sum_{m=1}^{3} P[\delta = m] \times \text{dnorm}(\theta, \mu_m, \sigma_m)$$

   by the law of total probability.

   (c) Write a Gibbs sampling algorithm (you can use the helpful R commands given on the course web page) to sample from the joint density of (θ, δ). Begin the chain with the initial values $\delta^{[0]} = 2$ and $\theta^{[0]} = 0$, and generate 1000 values of θ. Give a plot of a relative frequency histogram of the θ values – using a command like `hist(theta.values, freq=F)` – and comment on how it compares to the true marginal density of θ plotted on the last page. (You could try repeating part (c) a few times to get a sense for the variability in the Gibbs sampler results, as well.)

   The R code is given on the course webpage. The histogram may look something like this (I've added the true density on top of the histogram, just for a reference). Unless you got lucky, this histogram may not do a good job of approximating the distribution of θ.

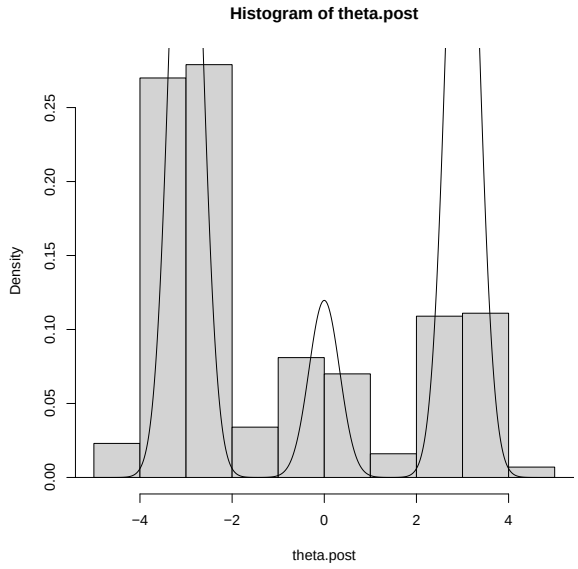

Figure 1: Problem 1: Histogram that approximates the distribution for θ, based on 1000 iterations.

(d) Repeat part (c), but generate 40000 values of θ. Again give a plot of a relative frequency histogram of the θ values and comment on how it compares to the true marginal density of θ plotted on the last page. (You could try repeating part (d) a few times to get a sense for the variability in these Gibbs sampler results, as well.)

The R code is given on the course webpage. The histogram may look something like this (I've added the true density on top of the histogram, just for a reference). This histogram should do a better job of approximating the distribution of θ.

(e) Try trace plots for the θ values and plots of the autocorrelation functions for parts (c) and (d). Comment on what these MCMC diagnostics tell you.

The trace plot shows the MCMC algorithm has not stabilized after 1000 iterations. The ACF plot shows severe autocorrelation between sampled values.

The trace plot shows the MCMC algorithm has not exactly converged to one spot, but that is just because of the nature of the multimodal distribution we are sampling from. All in all it seems to be sampling from the true distribution pretty well. The ACF plot still shows severe autocorrelation — some thinning may be useful.

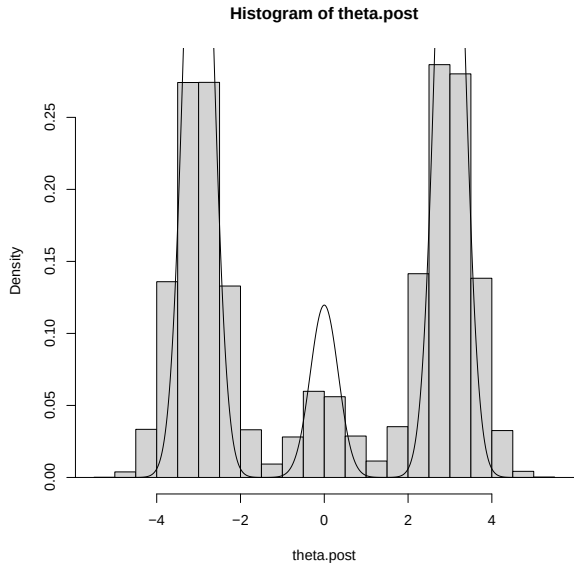

Figure 2: Problem 1: Histogram that approximates the distribution for θ, based on 40000 iterations.

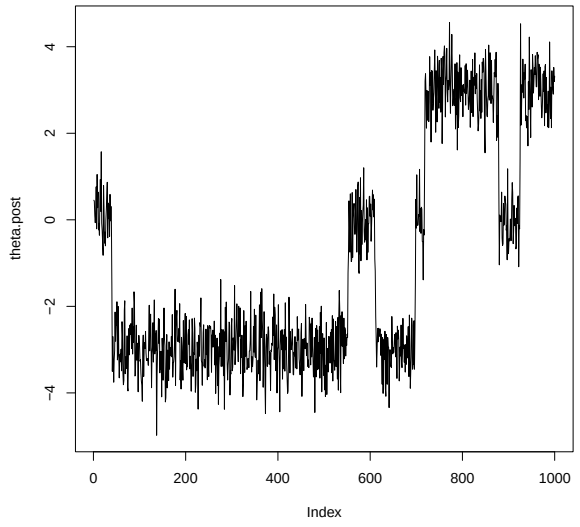

Figure 3: Problem 1: Trace plot based on 1000 iterations.

2. Two separate test developers have created different IQ tests. Each of these tests is designed so that scores follow a normal distribution with population mean 100 and standard deviation 15. Hence any set of test scores can be standardized to so that the

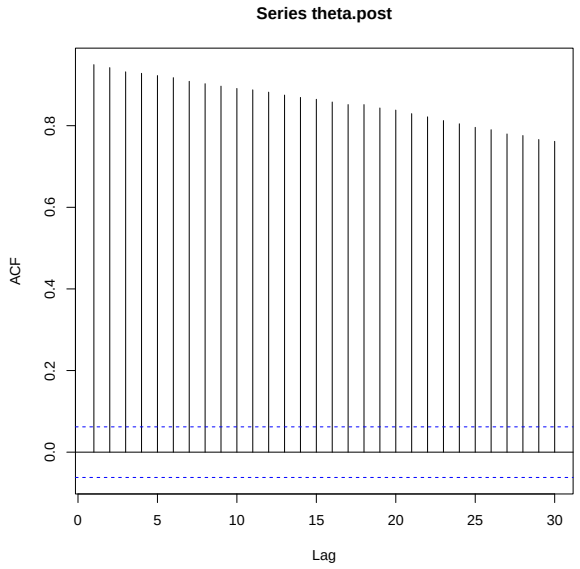

Figure 4: Problem 1: ACF plot based on 1000 iterations.

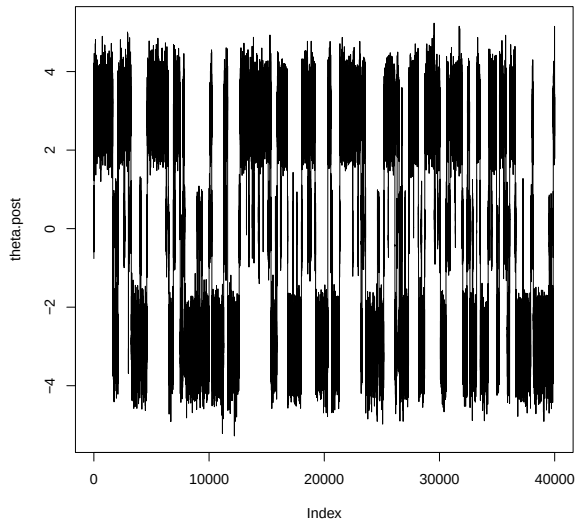

Figure 5: Problem 1: Trace plot based on 40000 iterations.

standardized scores follow a $N(0, 1)$ distribution. We wish to investigate the coefficient of correlation ρ between test-takers' scores on the two tests. Consider a random sample of n test-takers, who each take both IQ tests. Let $X_1, \ldots, X_n$ be the test-takers' standardized scores on the first test, and let $Y_1, \ldots, Y_n$ be the corresponding standardized

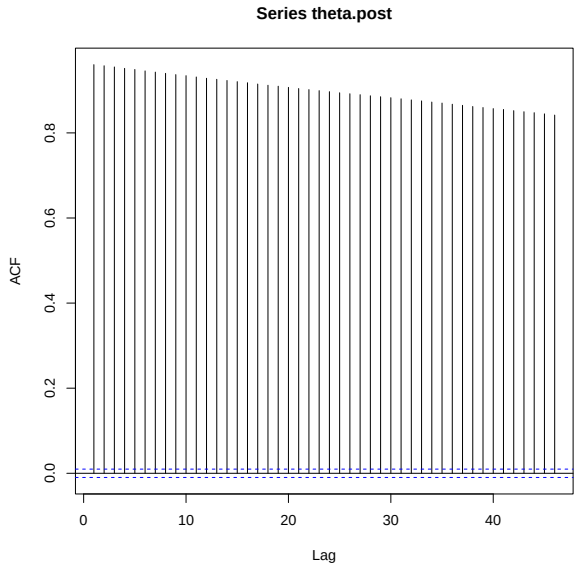

Figure 6: Problem 1: ACF plot based on 40000 iterations.

scores on the second test. The likelihood is

$$L(\rho|\mathbf{x},\mathbf{y}) = [2\pi]^{-n}[1-\rho^2]^{-n/2}\exp\left\{-\frac{1}{2(1-\rho^2)}\left[\sum x_i^2 - 2\rho\sum x_i y_i + \sum y_i^2\right]\right\}.$$

(a) Suppose we use the prior $p(\rho) = 1$, $0 < \rho < 1$, for ρ. Show that the posterior for ρ is

$$p(\rho|\mathbf{x},\mathbf{y}) \propto [(1-\rho^2)]^{-n/2}\exp\left\{-\frac{1}{2(1-\rho^2)}\left[\sum x_i^2 - 2\rho\sum x_i y_i + \sum y_i^2\right]\right\}$$

$$\begin{aligned} p(\rho|\mathbf{x},\mathbf{y}) &\propto L(\rho|\mathbf{x},\mathbf{y})p(\rho) \\ &= [2\pi]^{-n}[1-\rho^2]^{-n/2}\exp\left\{-\frac{1}{2(1-\rho^2)}\left[\sum x_i^2 - 2\rho\sum x_i y_i + \sum y_i^2\right]\right\} \quad (1) \\ &\propto [(1-\rho^2)]^{-n/2}\exp\left\{-\frac{1}{2(1-\rho^2)}\left[\sum x_i^2 - 2\rho\sum x_i y_i + \sum y_i^2\right]\right\} \end{aligned}$$

(b) We gather data on 13 test-takers. Their standardized scores on the two tests are:

```
X_i:  0.92  0.42  3.62  0.89 -0.69  0.45 -0.11 -0.14 -0.47  1.09 -0.34  0.62  0.27
Y_i:  0.26  1.65  2.10  0.62 -1.16  1.29 -0.82 -0.36 -0.29  0.86  0.19  1.25  0.33
```

Plugging in the necessary summary statistics, write and simplify the posterior (up to a constant of proportionality).

The summary statistics are $n = 13$, $\sum x_i^2 = 17.61$, $\sum x_i y_i = 12.48$, $\sum y_i^2 = 13.93$.

So

$$p(\rho|\mathbf{x}, \mathbf{y}) \propto [(1-\rho^2)]^{-13/2} \exp\left\{ -\frac{1}{2(1-\rho^2)} \left[17.61 - 2\rho(12.48) + 13.93 \right] \right\}$$

or equivalently

$$p(\rho|\mathbf{x}, \mathbf{y}) \propto [(1-\rho^2)]^{-6.5} \exp\left\{ -\frac{1}{2(1-\rho^2)} \left[31.54 - 24.96\rho \right] \right\}$$

(c) We will use the following proposal density to generate values of ρ:

(i) Given current value $\rho^{[t]}$, sample $\rho^* \sim \text{Uniform}(\rho^{[t]} - 0.2, \rho^{[t]} + 0.2)$.

(ii) If the sampled $\rho^* < 0$, then set $\rho^* = |\rho^*|$.

(iii) If the sampled $\rho^* > 1$, then set $\rho^* = 2 - \rho^*$.

Argue that this is a symmetric proposal density.

This is quite of tricky and technical, but here is an example argument:

If the current value $\rho^{[t]} \in [0.2, 0.8]$, then the uniform density centered at $\rho^{[t]}$ is clearly symmetric.

Suppose $\rho^{[t]} < 0.2$. For example, say $\rho^{[t]} = 0.1$, so that a sampled ρ^* of 0.03 or -0.03 each yield a candidate $\rho^* = 0.03$. The density value associated with a Unif$(0.1-0.2, 0.1+0.2)$ distribution at locations -0.03 and 0.03 is the same as the density value associated with a Unif$(0.03 - 0.2, 0.03 + 0.2)$ distribution at locations -0.1 and 0.1.

Now suppose $\rho^{[t]} > 0.8$. For example, say $\rho^{[t]} = 0.9$, so that a sampled ρ^* of 0.97 or 1.03 each yield a candidate $\rho^* = 0.97$. The density value associated with a Unif$(0.9 - 0.2, 0.9 + 0.2)$ distribution at locations 0.97 and 1.03 is the same as the density value associated with a Unif$(0.97-0.2, 0.97+0.2)$ distribution at locations 0.9 and 1.1 (each of which yield a next candidate ρ^* of 0.9.

(d) Explain carefully and completely how the provided R code on the course web page performs the Metropolis-Hastings algorithm.

(e) Using the code on the course web page, sample from the posterior distribution of ρ. Perform diagnostics to check convergence and check autocorrelation, and perform remedial action if needed. Summarize the posterior distribution of ρ, including giving an estimated density plot, point estimate, and 95% interval estimate for ρ.

My acceptance rate was around 0.64, which is a little high, but not terrible. This could be changed by adjusting the width of the proposal density.

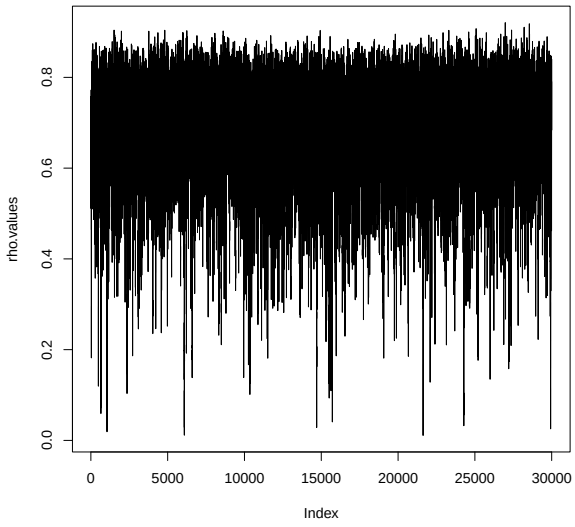

Figure 7: Problem 2: Trace plot based on 30000 iterations.

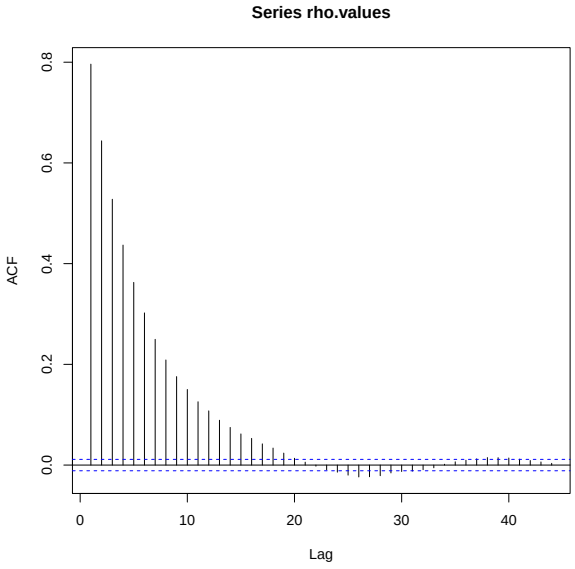

Figure 8: Problem 2: ACF plot based on 30000 iterations.

The trace plot shows the MCMC has stabilized well. The ACF plot shows high autocorrelation. We can remedy this by thinning; I will take every 10th value of the chain.

The trace plot still looks good. The thinned chain shows no autocorrelation, so the

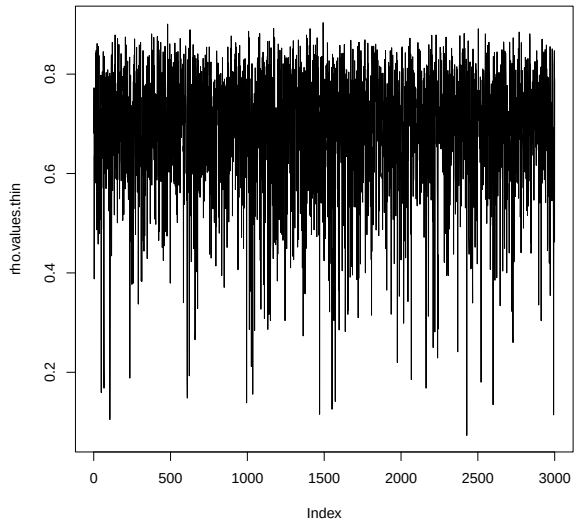

Figure 9: Problem 2: Trace plot based on thinned chain.

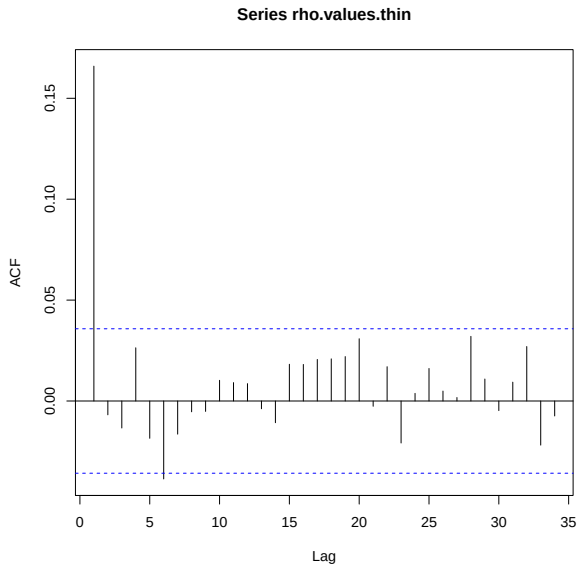

Figure 10: Problem 2: ACF plot based on thinned chain.

problem has been fixed.

Based on the thinned chain: The posterior median is around 0.71. A quantile based 95% credible interval is $(0.38, 0.86)$. An HPD 95% credible interval is $(0.43, 0.88)$.

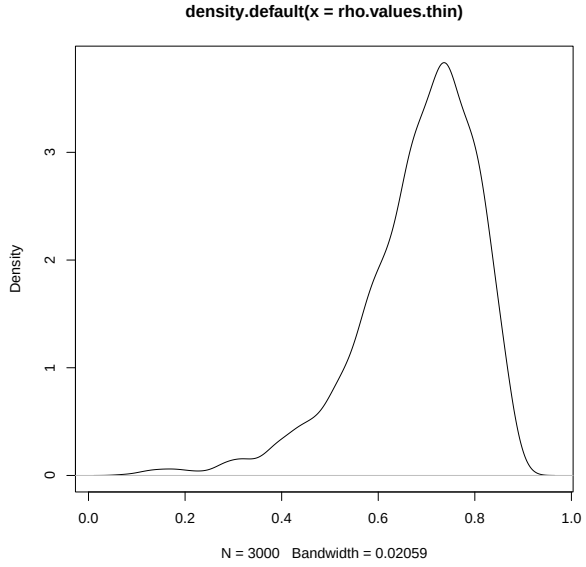

Figure 11: Problem 2: Density plot based on thinned chain.

3. Do Problem 6.3 from the *Bayes Rules!* textbook. [Read Section 6.3 for more details about "mixing slowly."]

   Mixing too slowly means that the chain is taking a long time to become a true sample from the posterior distribution. You would have to run the chain for many, many iterations in order to get valid results.

   High autocorrelation means that the values drawn from the posterior are not independent, so it is not a random sample from the posterior. This could affect the posterior inference.

   If the chain gets stuck, it is not exploring the entire posterior distribution; it may be getting stuck in areas of high probability. Again, the sampled values will not be representative of the whole posterior.

4. Do Problem 7.6 from the *Bayes Rules!* textbook. [Hint: Use R functions like `rnorm` and `runif` to draw the requested value, and **(just for the purposes of this exercise!)**, remember to enter `set.seed(84735)` before each random draw, so that the random number that you draw will match the random number that I draw.]

```
> #(a)
> set.seed(84735)
> rnorm(n=1,mean=4.6,sd=2)
[1] 5.934465
>
```

```
> #(b)
> set.seed(84735)
> rnorm(n=1,mean=2.1,sd=7)
[1] 6.770629
>
> #(c)
> set.seed(84735)
> runif(n=1,min=8.9-2,max=8.9+2)
[1] 9.890753
>
> #(d)
> set.seed(84735)
> runif(n=1,min=1.2-0.5,max=1.2+0.5)
[1] 1.447688
>
> #(e)
> set.seed(84735)
> runif(n=1,min=7.7-3,max=7.7+3)
[1] 9.18613
>
```

5. Do Problem 7.7 from the *Bayes Rules!* textbook. [Hint: Use R functions like `dnorm`, `dunif`, and `dexp` to calculate the value of the proposal density for the specified current value and proposed value of the parameter. These proposal density values are the q parts that go in the numerator and denominator of the Metropolis-Hastings acceptance ratio probability.]

```
> proposed<-2.1
> current<-2
>
> #(a)
>
> top<-((proposed)^(-2))*dnorm(current,mean=proposed, sd=1)
> bottom<-((current)^(-2))*dnorm(proposed,mean=current, sd=1)
> accept.ratio <- top/bottom
> print(accept.ratio)
[1] 0.9070295
>
> #(b)
>
```

```
> top<-(exp(proposed))*dnorm(current,mean=proposed, sd=0.2)
> bottom<-(exp(current))*dnorm(proposed,mean=current, sd=0.2)
> accept.ratio <- top/bottom
> print(accept.ratio)
[1] 1.105171
>
> #(c)
>
> top<-(exp(-10*proposed))*dunif(current,min=proposed-0.5, max=proposed+0.5)
> bottom<-(exp(-10*current))*dunif(proposed,min=current-0.5, max=current+0.5)
> accept.ratio <- top/bottom
> print(accept.ratio)
[1] 0.3678794
>
> #(d)
>
> top<-(exp(-(proposed^4)))*dexp(current,rate=proposed)
> bottom<-(exp(-(current^4)))*dexp(proposed,rate=current)
> accept.ratio <- top/bottom
> print(accept.ratio)
[1] 0.03339631
>
```

We can see that in part (b), we are DEFINITELY going to accept the proposed value, since the acceptance ratio is greater than 1.

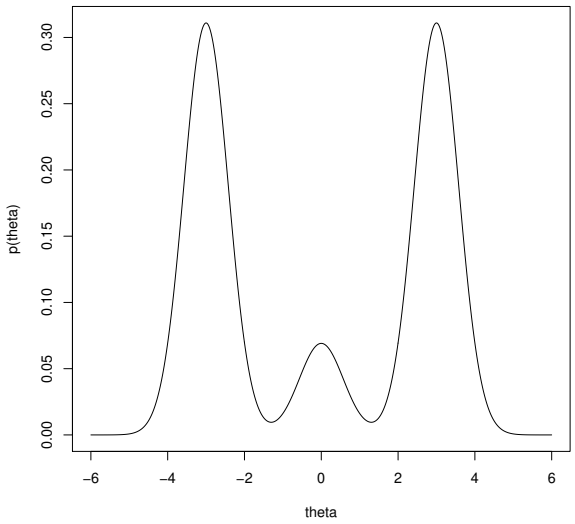

Figure 12: True distribution for θ.