Mixture of Normals

2025-10-06

Mixture of Normals density

The density we're interest in sampling from is

$$f(\theta) = 0.45N\left(\theta; -3, \frac{1}{4}\right) + 0.10N\left(\theta; 0, \frac{1}{4}\right) + 0.45N\left(\theta; 3, \frac{1}{4}\right)$$

This is a *mixutre* of three Normals. You've seen your first mixture distribution in this class on the previous homework!

We can easily evaluate this distribution at a grid of θ values.

```
w_prob <- c(0.45, 0.1, 0.45)
mu_vec <- c(-3, 0, 3)
sigma_vec <- c(0.5, 0.5,0.5)

# true density
theta_vec <- seq(-10, 10, 0.01)
mixture_dens <- rep(NA, length(theta_vec))
for(i in 1:length(theta_vec)){
  mixture_dens[i] <- sum(dnorm(theta_vec[i], mu_vec, sigma_vec)*w_prob)
}
```

Monte Carlo sampling

If we want to sample from this mixture distribution, we can use Monte Calo sampling! For $s = 1, \ldots, S$:

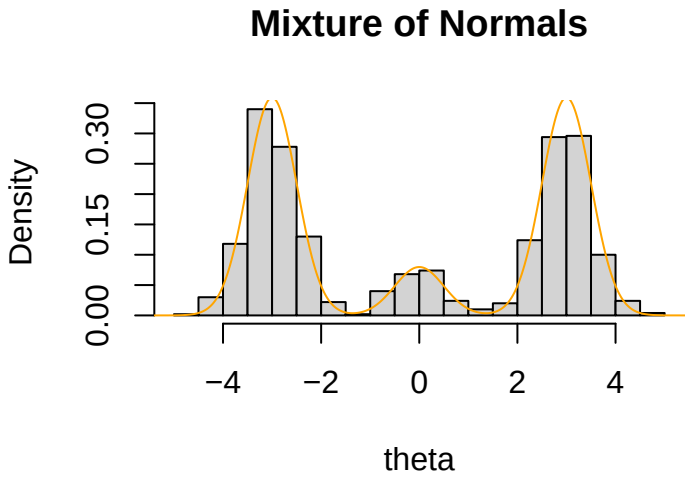
1. Sample a value $w^{(s)} = \{1, 2, 3\}$ according to mixture weights $\{0.45, 0.10, 0.45\}$
2. Given the value of w, sample $\theta^{(s)}$ from the corresponding Normal. For example:

$$\theta^{(s)}|w^{(s)} = 1 \sim N\left(-3, \frac{1}{4}\right)$$

Note this is indeed Monte Carlo sampling because we are drawing *independent* random sample!

Let's draw 1000 MC samples. A normalized histogram of the samples is shown below, along with the true density overlaid in orange:

```
set.seed(1)
# monte carlo
S <- 1000
d_samp <- sample(1:3, S, replace = T, prob = w_prob)
hist(rnorm(S, mu_vec[d_samp], sigma_vec[d_samp]), freq = F, breaks =
       20, main = "Mixture of Normals", xlab = "theta")
lines(theta_vec, mixture_dens, type = "l", col = "orange")
```



MCMC sampling

We can also obtain samples from this distribution via Gibbs sampling (i.e. Markov Chain Monte Carlo)! Note that even though we're not working with a posterior, we can still perform Gibbs Sampling if we have more than one parameter of interest. In this case, why don't I iterate between 1) sampling a θ value from its full conditional and 2) sampling a mixture component w value from its full conditional? That is, for $s = 1, \ldots, S$:

1. Sample $w^{(s)} \sim f(w|\theta^{s-1})$
2. Sample $\theta^{(s)} \sim f(\theta|w^{(s)})$

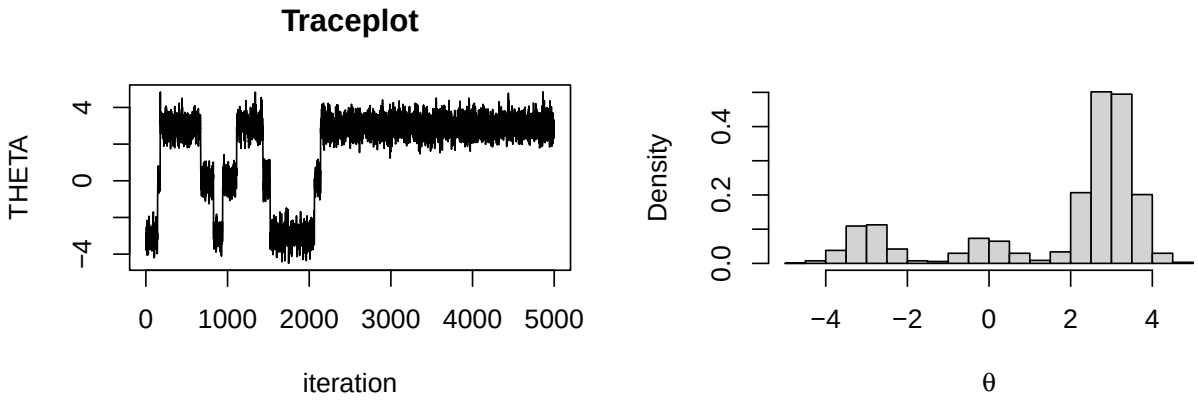
Convince yourself that this sampling scheme is different from the previous!

We already have the distrution for sampling in step 2. So we just need to derive the full conditional for w! Let's do that together.

Gibbs sampler

Let's run our sampler for $S = 5000$ iterations, then take a look at the traceplot and marginal histogram for θ. *Note: even though we're sampling w's along the way, I don't actually care about them (they are latent variables). So I don't bother storing them!*

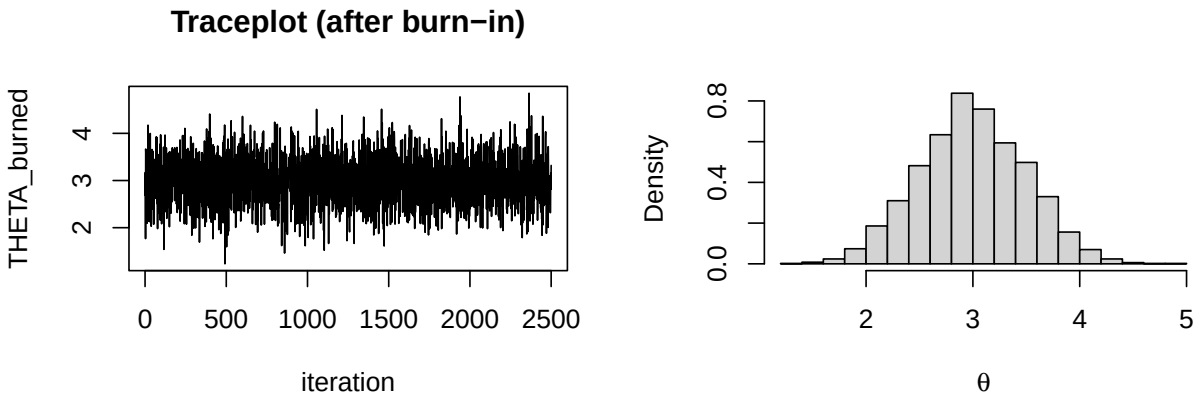
```
seed <- 10
S <- 5000
set.seed(seed)
theta <- -2
THETA <- rep(NA, S)
for(s in 1:S){
  # sample w
  prob_vec_unnorm <- dnorm(theta, mu_vec, sigma_vec) * w_prob
  prob_vec_norm <- prob_vec_unnorm/sum(prob_vec_unnorm)
  w <- sample(1:3, size = 1, prob = prob_vec_norm)

  # sample theta
  theta <- rnorm(1, mu_vec[w], sigma_vec[w])
  THETA[s] <- theta
}
par(mfrow = c(1, 2))
plot(THETA, type = "l", main = "Traceplot", xlab = "iteration")
hist(THETA, freq = F, main = "", xlab = expression(theta))
```



What are we noticing?

Well, we did say that we should throw out the first chunk of iterations to burn-in. So let me throw out the first half and re-visualize:

```
THETA_burned <- THETA[-c(1:(S/2))]
```

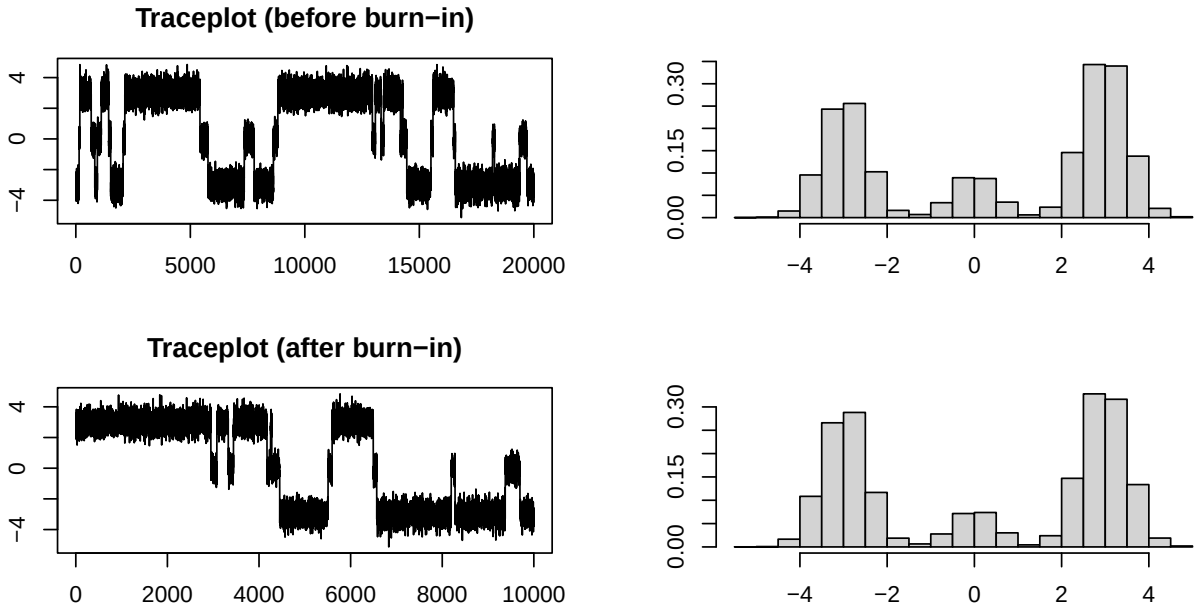


Yikes!! That sure is misleading!!!! After burn-in, it seems like my density has converged to a unimodal distribution. What's happening here?

Let's talk about convergence, mixing, and autocorrelation!

Gibbs, take 2

Let's now run the chain a lot longer, this time for $S = 20000$ iterations.



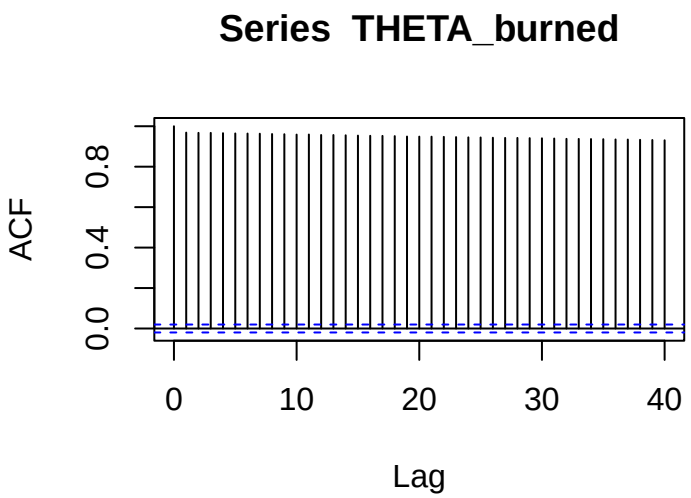
That's much better! Of course, in real life, we don't know what the desire (posterior) density should look like. But this goes to show that you should, if computationally feasible, run your chain for a long time and perhaps for many different S values.

Functions for MCMC diagnostics

Autocorrelation

The `R` function for examining autocorrelation is `acf()`, and you pass in a vector of samples:

```
acf(THETA_burned)
```



The horizontal blue linee give the values beyond which the autocorrelations are (statistically) significantly different from 0. So you'd like ACF values within the blue bands.

If you just want the sample lag t ACF value, you can access it as follows:

```
acf_out <- acf(THETA_burned, plot = F)
# lag-0 is at index 1, so lag-1 is at index 2
acf_out$acf[2]
```

```
[1] 0.9685008
```

Effective sample size

We can obtain n_{eff} using the `effectiveSize()` function from the `R` package `coda` (install it first).

```
coda::effectiveSize(THETA_burned) # yikes
```

```
    var1
6.235923
```