The Metropolis algorithm

Stat 340, Fall 2021

Example: Launch failures

The Federal Aviation Administration (FAA) and the United States Air Force (USAF) were interested in estimating the failure probability for new rockets launched by companies with limited experience. Failures have a have significant on both public safety and aerospace manufacturers' ability to develop and field new rocket systems. Johnson et al. (2005) analyzed data from 1980-2002. In total, there were 11 launches, of which 3 were successful.

You can load the entire data set with the code below.

```r
launches <- read.table("https://alysongwilson.github.io/BR/table21.txt", header = TRUE)
```

Algorithm

1. Select a value $\theta^{(0)}$ where $\pi_n(\theta^{(0)}) > 0$
2. Given the current draw $\theta^{(i)}$, propose a *candidate draw* $\theta^p \sim \text{Unif}(\theta^{(i)} + C, \theta^{(i)} + C)$.
3. Evaluate the (unnormalized) posterior at the current value: $\pi_n(\theta^{(i)})$.
4. Evaluate the (unnormalized) posterior at the candidate: $\pi_n(\theta^c)$.
5. Accept candidate with probability $R = \min\left\{\pi_n(\theta^c)/\pi_n(\theta^{(i)}), 1\right\}$.
   - Draw $U \sim \text{Unif}(0, 1)$, if $U < R$ set $\theta^{(i+1)} = \theta^p$
   - Otherwise, set $\theta^{(i+1)} = \theta^{(i)}$.

Implementation in R

Your textbook authors provide an R function to implement a general Metropolis algorithm in R (see page 326). The below is an adapted version of that function where I added a "safety" check to ensure draws outside the parameter space couldn't be made.

```r
metropolis <- function(logpost, current, C, iter, ...){
  S <- rep(0, iter) # container for draws
  n_accept <- 0     # acceptance counter

  # Iterate through candidate draws
  for(j in 1:iter){
  candidate <- runif(1, min = current - C, max = current + C)
  prob <- exp(logpost(candidate, ...) -
             logpost(current, ...))

  if(is.nan(prob)) prob <- 0 # deal with draws outside parameter space

  accept <- ifelse(runif(1) < prob, "yes", "no")
  current <- ifelse(accept == "yes", candidate, current)
  S[j] <- current
  n_accept <- n_accept + (accept == "yes")
  }

  list(S=S, accept_rate=n_accept / iter) # Return draws and acceptance rate
}
```

To use this function, you need to write a function for the log posterior density (it can be unnormalized) that takes a list argument to pass in the data/sufficient statistics. Whenever possible, use built-in density functions with `log = TRUE`. Run `?Distributions` for a list.

```
log_posterior <- function(.theta, samp) {
  dbinom(samp$y, size = samp$n, prob = .theta, log = TRUE) + dunif(.theta, 0.1, 0.9, log = TRUE)
}
```

Random walk Metropolis algorithm

Example: Fluid breakdown

Engineers needed to understand how long machines can run before replacing oil in a factory. They collected viscosity breakdown times (in thousands of hours) for 50 sample.

```
breakdown <- read.csv("https://alysongwilson.github.io/BR/table23.txt")
```

1. Write a `log_posterior` function. Notice that you can use the `dnorm` function if you log the data.

```
# put your log posterior code here
```

2. Run the `metropolis()` function to obtain draws from the (approximate) posterior distribution.

```
# Run metropolis here
```

3. Check the trace and ACF plots to see if your chain converged and if it's working efficiently. To create a trace plot, you can use the following code chunk. Here `mcmc_draws` store the output from `metropolis()`.

```
ggplot() +
  geom_line(aes(x = seq_along(mcmc_draws$S), y = mcmc_draws$S)) +
  labs(y = bquote(theta), x = "Iteration",
       caption = paste("Acceptance rate:", mcmc_draws$accept_rate))
```

To create an ACF plot, you can use the `acf()` function:

```
acf(mcmc_draws$S, main = "MCMC draws")
```

4. Repeat 2-3 until you're satisfied.
5. Construct and interpret a 95% credible interval for the viscosity breakdown times.