Chapter 9.3a The Metropolis Algorithm

Jim Albert and Monika Hu

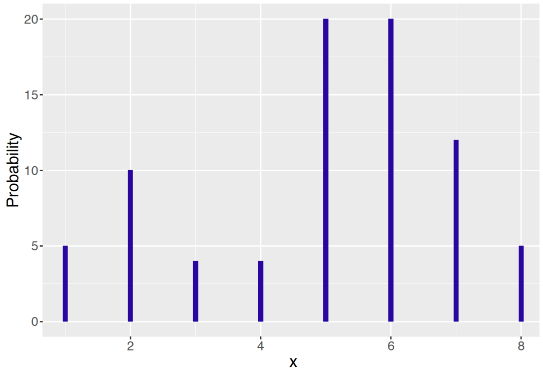
Chapter 9 Simulation by Markov Chain Monte Carlo

Example: Walking on a number line

- Markov chains can be used to sample from an arbitrary probability distribution.
- Illustrate sampling from a discrete distribution. on the integers 1, . . . , K.
- As an example, we write a short function `pd()` in R taking on the values 1, . . . , 8 with probabilities proportional to the values 5, 10, 4, 4, 20, 20, 12, and 5.

```
pd <- function(x){
  values <- c(5, 10, 4, 4, 20, 20, 12, 5)
  ifelse(x %in% 1:length(values), values[x], 0)
}
prob_dist <- data.frame(x = 1:8,
                        prob = pd(1:8))
```

Graph of Distribution



Take a Random Walk

1. We start at any possible location of our random variable from 1 to $K = 8$.

2. A fair coin decides where to visit next. If the coin lands heads, we think about visiting the location one value to the left, and if coin lands tails, we consider visiting the location one value to right. We call this location the "candidate" location.

3. We compute
$$R = \frac{pd(candidate)}{pd(current)}, \tag{1}$$
the ratio of the probabilities at the candidate and current locations.

4. We spin a continuous spinner X that lands anywhere from 0 to 1 . If X is smaller than R, we move to the candidate location, and otherwise we remain at the current location.

Algorithm

- Steps 1 through 4 define an irreducible, aperiodic Markov chain on the state values $\{1, 2, \ldots, 8\}$.
- Step 1 gives the starting location and the transition matrix P is defined by Steps 2 through 4.
- One "discovers" the discrete probability distribution pd is by starting at any location and walking through the distribution many times repeating Steps 2, 3, and 4
- If this process is repeated for a large number of steps, the distribution of our actual visits should approximate the probability distribution pd.

Running this in R

A R function `random_walk()`is written implementing this random walk algorithm.

- There are three inputs to this function, the probability distribution `pd`, the starting location `start` and the number of steps of the algorithm `s`.

```
random_walk <- function(pd, start, num_steps){
  y <- rep(0, num_steps)
  current <- start
  for (j in 1:num_steps){
    candidate <- current + sample(c(-1, 1), 1)
    prob <- pd(candidate) / pd(current)
    if (runif(1) < prob) current <- candidate
    y[j] <- current
  }
  return(y)
}
```

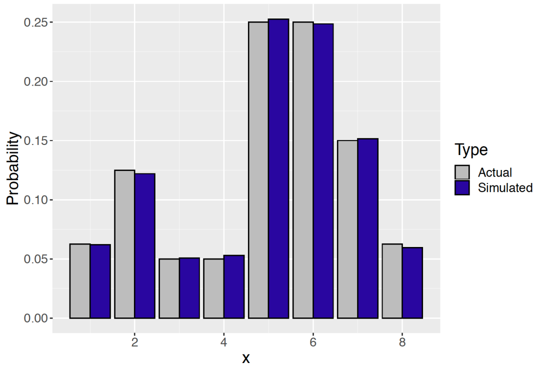
Run the Random Walk Algorithm

- We implement the random walk algorithm by inputting this probability function, starting at the value $X = 4$ and running the algorithm for $s = 10{,}000$ iterations.

```
out <- random_walk(pd, 4, 10000)
data.frame(out) %>% group_by(out) %>%
  summarize(N = n(), Prob = N / 10000) -> S
```

Does it Work?

- Figure shows a histogram of the simulated values from the random walk is compared with the actual probability distribution.
- Note that the collection of simulated draws appears to be a close match to the true probabilities.


Probability
x
Type
Actual
Simulated