Bayesian statistics with R
5. Markov chains Monte Carlo (MCMC)

Olivier Gimenez

March 2021

Get posteriors with Markov chains Monte Carlo (MCMC) methods

Back to the Bayes' theorem

- Bayes inference is easy! Well, not so easy in real-life applications.

Back to the Bayes' theorem

- Bayes inference is easy! Well, not so easy in real-life applications.
- The issue is in $\Pr(\theta \mid \text{data}) = \dfrac{\Pr(\text{data} \mid \theta)\Pr(\theta)}{\Pr(\text{data})}$

Back to the Bayes' theorem

- Bayes inference is easy! Well, not so easy in real-life applications.
- The issue is in $\Pr(\theta \mid \text{data}) = \dfrac{\Pr(\text{data} \mid \theta)\Pr(\theta)}{\Pr(\text{data})}$
- $\Pr(\text{data}) = \int L(\text{data} \mid \theta)\Pr(\theta) d\theta$ is a N-dimensional integral if $\theta = \theta_1, \ldots, \theta_N$

Back to the Bayes' theorem

- Bayes inference is easy! Well, not so easy in real-life applications.
- The issue is in $\Pr(\theta \mid \text{data}) = \dfrac{\Pr(\text{data} \mid \theta)\Pr(\theta)}{\Pr(\text{data})}$
- $\Pr(\text{data}) = \int L(\text{data} \mid \theta)\Pr(\theta) d\theta$ is a N-dimensional integral if $\theta = \theta_1, \ldots, \theta_N$
- Difficult, if not impossible to calculate!

Brute force approach via numerical integration

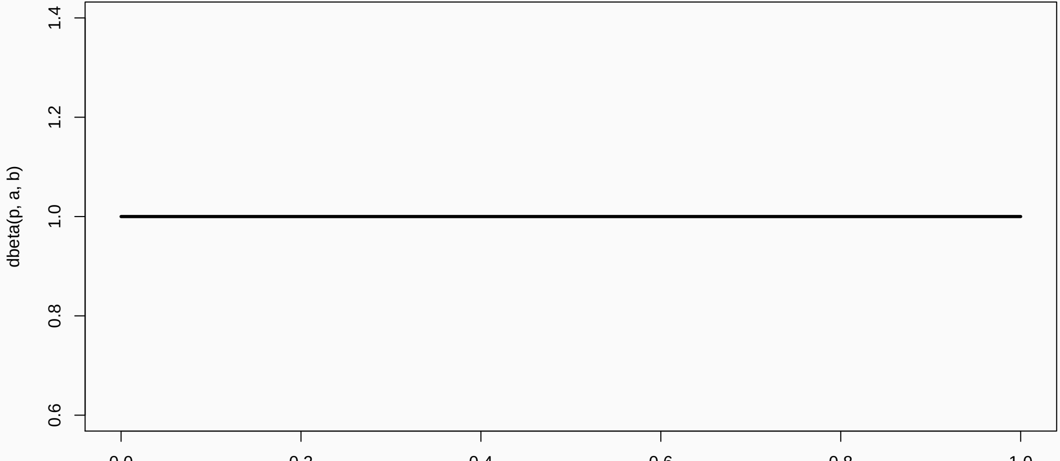
- Deer data

```r
y <- 19 # nb of success
n <- 57 # nb of attempts
```

- Likelihood Binomial$(57, \theta)$
- Prior Beta$(a = 1, b = 1)$

Beta prior

```r
a <- 1; b <- 1; p <- seq(0,1,.002)
plot(p, dbeta(p,a,b), type='l', lwd=3)
```



Apply Bayes theorem

- Likelihood times the prior: $\Pr(\text{data} \mid \theta)\ \Pr(\theta)$

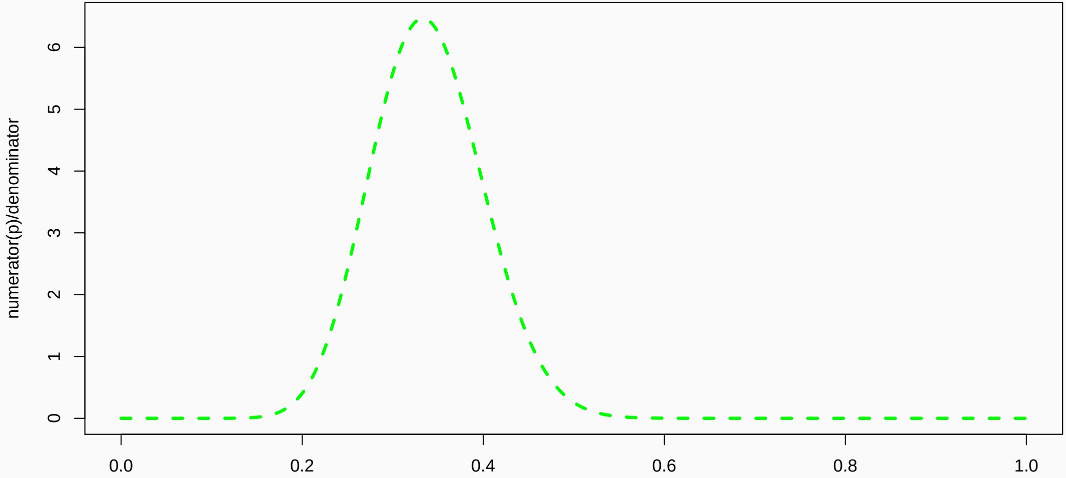
```
numerator <- function(p) dbinom(y,n,p)*dbeta(p,a,b)
```

- Averaged likelihood: $\Pr(\text{data}) = \int L(\theta \mid \text{data})\ \Pr(\theta) d\theta$

```
denominator <- integrate(numerator,0,1)$value
```

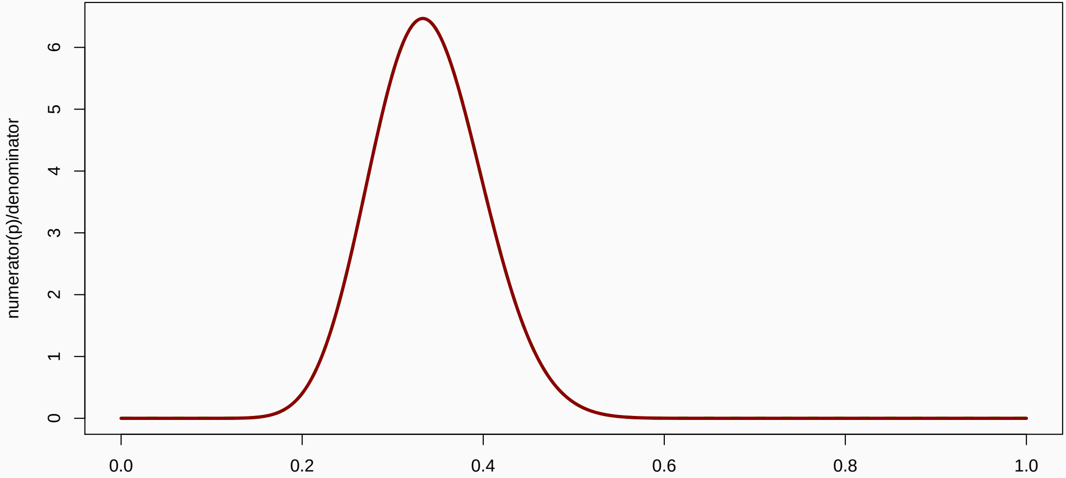
Posterior inference via numerical integration

```
plot(p, numerator(p)/denominator,type="l", lwd=3, col="green", lty=2)
```



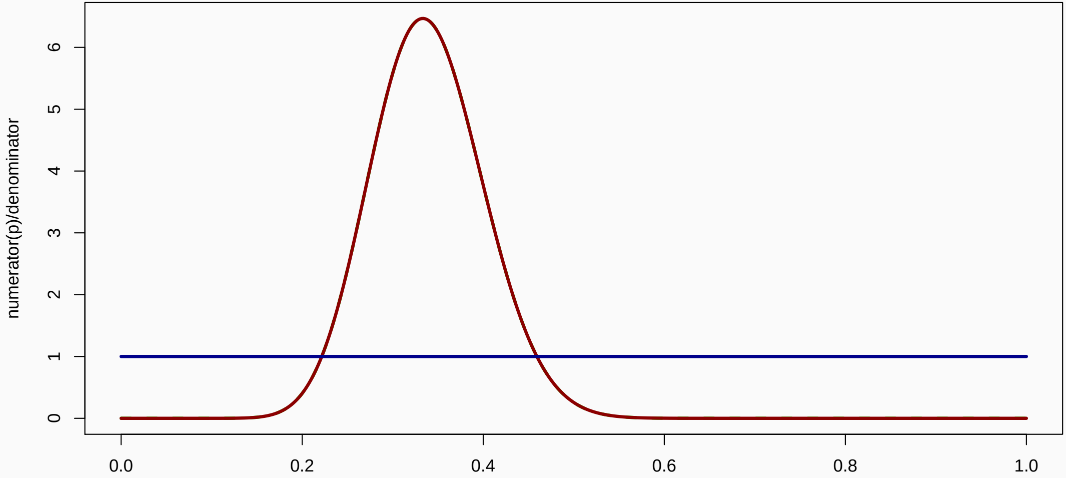
Superimpose explicit posterior distribution (Beta formula)

```
lines(p, dbeta(p,y+a,n-y+b), col='darkred', lwd=3)
```



And the prior

```
lines(p, dbeta(p,a,b), col='darkblue', lwd=3)
```



What if multiple parameters, like in a simple linear regression?

- Example of a linear regression with parameters α, β and σ to be estimated.

What if multiple parameters, like in a simple linear regression?

- Example of a linear regression with parameters α, β and σ to be estimated.
- Bayes' theorem says:

$$P(\alpha, \beta, \sigma \mid \text{data}) = \frac{P(\text{data} \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)}{\iiint P(\text{data} \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)\, d\alpha\, d\beta\, d\sigma}$$

What if multiple parameters, like in a simple linear regression?

- Example of a linear regression with parameters α, β and σ to be estimated.
- Bayes' theorem says:

$$P(\alpha, \beta, \sigma \mid \text{data}) = \frac{P(\text{data} \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)}{\iiint P(\text{data} \mid \alpha, \beta, \sigma)\, P(\alpha, \beta, \sigma)\, d\alpha\, d\beta\, d\sigma}$$

- Do we really wish to calculate a 3D integral?

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*

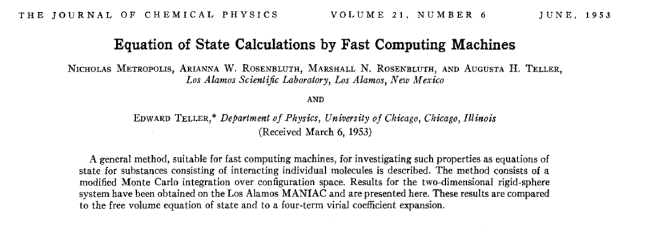
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- Use stochastic simulation to draw samples from posterior distributions.

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- Use stochastic simulation to draw samples from posterior distributions.
- Avoid explicit calculation of integrals in Bayes formula.

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.

THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

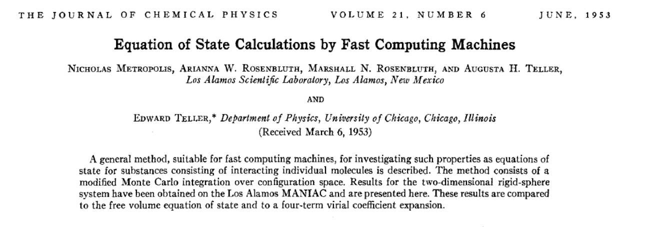
EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- Use stochastic simulation to draw samples from posterior distributions.
- Avoid explicit calculation of integrals in Bayes formula.
- Instead, approximate posterior to arbitrary degree of precision by drawing large sample.

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics.



THE JOURNAL OF CHEMICAL PHYSICS VOLUME 21, NUMBER 6 JUNE, 1953

Equation of State Calculations by Fast Computing Machines

NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,
Los Alamos Scientific Laboratory, Los Alamos, New Mexico

AND

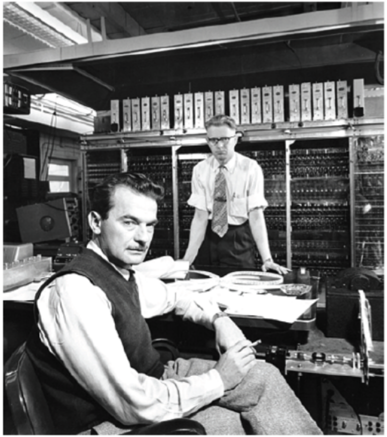
EDWARD TELLER,* *Department of Physics, University of Chicago, Chicago, Illinois*
(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.

- Use stochastic simulation to draw samples from posterior distributions.
- Avoid explicit calculation of integrals in Bayes formula.
- Instead, approximate posterior to arbitrary degree of precision by drawing large sample.
- Markov chain Monte Carlo = MCMC; boost to Bayesian statistics!

MANIAC

MANIAC:
Mathematical Analyzer, Numerical Integrator, and Computer



MANIAC:
1000 pounds
5 kilobytes of memory
70k multiplications/sec

Your laptop:
4–7 pounds
2–8 million kilobytes
Billions of multiplications/sec

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain).

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain).
- Converge to equilibrium (aka stationary) distribution.

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain).
- Converge to equilibrium (aka stationary) distribution.
- Equilibrium distribution is the desired posterior distribution!

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain).
- Converge to equilibrium (aka stationary) distribution.
- Equilibrium distribution is the desired posterior distribution!
- Several ways of constructing these chains: e.g., Metropolis-Hastings, Gibbs sampler, Metropolis-within-Gibbs.

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain).
- Converge to equilibrium (aka stationary) distribution.
- Equilibrium distribution is the desired posterior distribution!
- Several ways of constructing these chains: e.g., Metropolis-Hastings, Gibbs sampler, Metropolis-within-Gibbs.
- How to implement them in practice?!

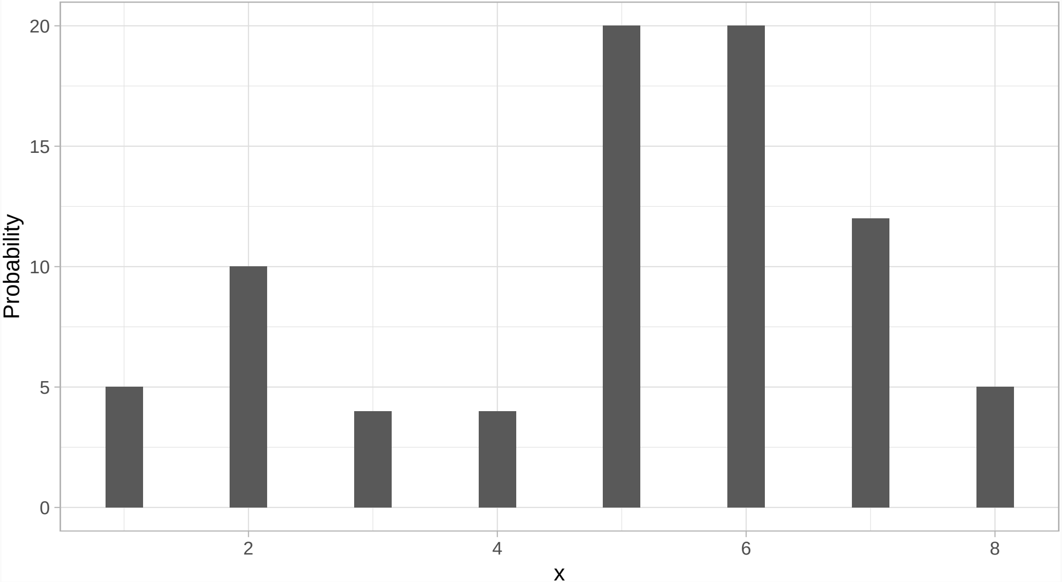
The Metropolis algorithm

- We illustrate sampling from a discrete distribution. Suppose we define a discrete probability distribution on the integers $1, \ldots, K$.

The Metropolis algorithm

- We illustrate sampling from a discrete distribution. Suppose we define a discrete probability distribution on the integers $1, \ldots, K$.
- We write a short function `pd()` in `R` taking on the values $1, \ldots, 8$ with probabilities proportional to the values 5, 10, 4, 4, 20, 20, 12, and 5.

```r
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))
prob_dist
#>   x prob
#> 1 1    5
#> 2 2   10
#> 3 3    4
#> 4 4    4
#> 5 5   20
#> 6 6   20
#> 7 7   12
#> 8 8    5
```


Probability
20
15
10
5
0
2
4
6
8
x

To simulate from this probability distribution, we take a **random walk** described as follows.

To simulate from this probability distribution, we take a **random walk** described as follows.

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

To simulate from this probability distribution, we take a **random walk** described as follows.

1. We start at any possible location of our random variable from 1 to $K = 8$
2. To decide where to visit next, a fair coin is flipped. 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.

To simulate from this probability distribution, we take a **random walk** described as follows.

1. We start at any possible location of our random variable from 1 to $K = 8$
2. To decide where to visit next, a fair coin is flipped. 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 the ratio of the probabilities at the candidate and current locations $R = pd(candidate)/pd(current)$

To simulate from this probability distribution, we take a **random walk** described as follows.

1. We start at any possible location of our random variable from 1 to $K = 8$
2. To decide where to visit next, a fair coin is flipped. 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 the ratio of the probabilities at the candidate and current locations $R = pd(candidate)/pd(current)$
4. We spin a continuous spinner that lands anywhere from 0 to 1 – call the random spin X. If X is smaller than R, we move to the candidate location, and otherwise we remain at the current location.

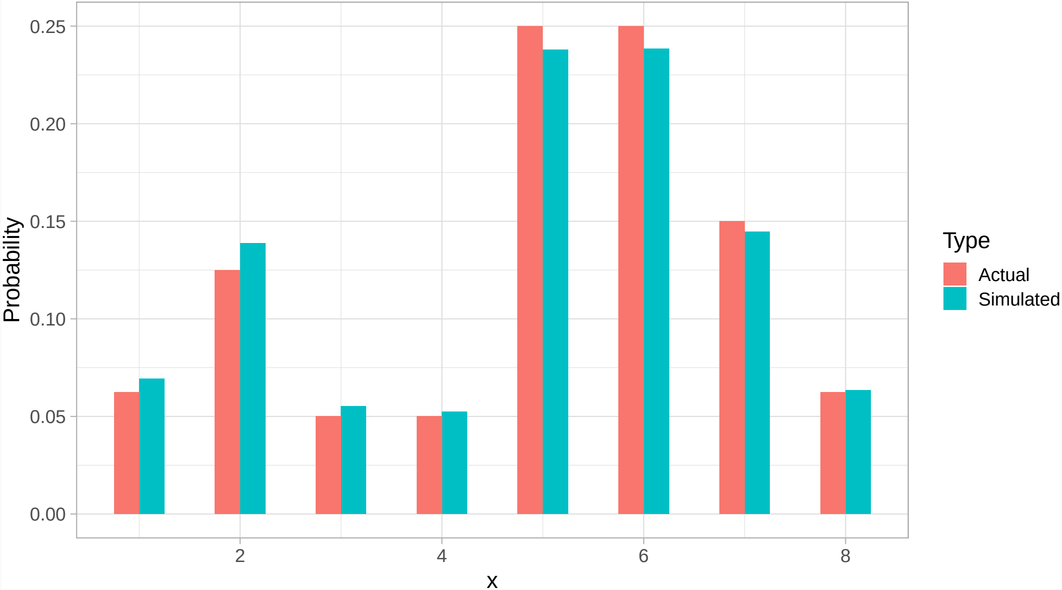
To simulate from this probability distribution, we take a **random walk** described as follows.

1. We start at any possible location of our random variable from 1 to $K = 8$
2. To decide where to visit next, a fair coin is flipped. 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 the ratio of the probabilities at the candidate and current locations $R = pd(candidate)/pd(current)$
4. We spin a continuous spinner that lands anywhere from 0 to 1 – call the random spin X. If X is smaller than R, we move to the candidate location, and otherwise we remain at the current location.
5. We repeat 2-4 a number of times called **steps** (many steps).

```r
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)
}
```

Starting at the value $X = 4$ and running the algorithm for $s = 10,000$ iterations.

```
out <- random_walk(pd, 4, 10000)
head(out)
#> [1] 3 2 2 2 1 1
tail(out)
#> [1] 4 5 5 5 6 7
```


0.25
0.20
0.15
0.10
0.05
0.00
Probability
2
4
6
8
x
Type
Actual
Simulated

Animating the Metropolis algorithm - 2D example

https://mbjoseph.github.io/posts/2018-12-25-animating-the-metropolis-algorithm/

The Markov-chain Monte Carlo Interactive Gallery

https://chi-feng.github.io/mcmc-demo/