COSC/DATA 405/505

Modelling and Simulation

Outline

- Reversible Markov Chains
- The Metropolis-Hastings Algorithm
- A First Look at Bayesian Statistics
- A Worked Example
- Built-in Software

In what follows, the state space can be infinite or finite.

```
set.seed(222696)  # use this to reproduce output
```

Reversible Markov Chains

A Markov chain is said to be *time-reversible* if the Markov property holds for the chain when it is time reversed:

$$P(X_n = x_n | X_{n+1} = x_{n+1}, \ldots) = P(X_n = x_n | X_{n+1} = x_{n+1})$$

Theorem

A Markov Chain with transition matrix P is reversible if there exists a vector q such that

$$q_j P_{ji} = q_i P_{ij}.$$

Reversible Markov Chains

For such a q, observe what happens when we multiply q by P.

The ith component of the vector qP is

$$\sum_{j=1}^{\infty} q_j P_{ji}$$

and this is

$$q_i \sum_{j=1}^{\infty} P_{ij} = q_i$$

if the Markov chain is reversible.

Therefore

$$\mathbf{q}P = \mathbf{q}.$$

$\rightsquigarrow$ **q is the steady state vector for P.**

Example: Symmetric Random Walk

Suppose $S = \{0, \pm 1, \pm 2, \ldots, \pm k\}$ **for some** $k > 2$**.**

$$X_n = X_{n-1} + 2B_n - 1$$

where B_n **is Bernoulli with parameter** $p = .5$**, independent of** X_{n-1}**.**
When $X_{n-1} = \pm k$**,** $X_n = k - 1$ **(or** $1 - k$**).**

For $|j| < k$**,**

$$P_{j,j+1} = P_{j,j-1} = 0.5.$$

and

$$P_{k,k-1} = 1 = P_{-k,-k+1}.$$

Example: Symmetric Random Walk

$$q_k P_{k,k-1} = q_{k-1} P_{k-1,k} \quad \textbf{or}$$

$$q_k = q_{k-1} \times 0.5$$

and similarly,

$$q_{-k} = q_{1-k} \times 0.5.$$

For $|j| < k - 1$,

$$q_j P_{j,j+1} = q_{j+1} P_{j+1,j} \quad \textbf{or}$$

$$0.5 q_j = 0.5 q_{j+1}.$$

Therefore, all q's other than the q_k and q_{-k} are equal, and have twice the value of q_k and q_{-k}.

This Markov chain is time-reversible.

Example: Symmetric Random Walk

Steady-state distribution:

$$q_j = \frac{1}{2(k-1)+1+1}$$

and

$$q_k = q_{-k} = \frac{1}{4k}$$

e.g. $k = 3$**:**

$$q_3 = q_{-3} = \frac{1}{12}$$

$$q_2 = q_1 = q_0 = q_{-1} = q_{-2} = \frac{1}{6}.$$

A Simulation Check on the Calculation

Entering the transition matrix:

```
P <- matrix(c(0,1,0,0,0,0,0,
.5,0,.5,0,0,0,0,0,.5,0,.5,0,0,0,
0,0,.5,0,.5,0,0,0,0,0,0.5,0,.5,0,
0,0,0,0,.5,0,.5,0, 0, 0, 0, 0, 1, 0), nrow=7, byrow = TRUE)
```

A Simulation Check on the Calculation

Simulating the random walk:

```r
Ntransitions <- 100000 # number of moves
location <- numeric(Ntransitions)#initializing
current.state <- 1 # initial stock
for (t in 1:Ntransitions) {
   current.state <- sample(1:7,
        size = 1, prob = P[current.state, ])
   location[t] <- current.state
}
pi <- table(location)/Ntransitions
pi

## location
##       1       2       3       4       5       6       7
## 0.08196 0.16490 0.16702 0.16906 0.16881 0.16604 0.08221
```

Conventional Calculation of the Steady-State Vector

```r
A <- t(P) - diag(rep(1,7))
A <- rbind(A, rep(1,7))
RHS <- c(rep(0,7), 1)
options(digits=3)
qr.solve(A, RHS)

## [1] 0.0833 0.1667 0.1667 0.1667 0.1667 0.1667 0.0833
```

Other Time-Reversible Markov Chains

Suppose $\{\pi_i, i = 0, \pm 1, \pm 2, \ldots\}$ **is a set of positive real numbers with** $\sum_{i=-\infty}^{\infty} \pi_i = 1$**. (This is a probability distribution on the integers.)**

Set

$$P_{i,j} = \frac{1}{6} \min\left(\frac{\pi_j}{\pi_i}, 1\right), \text{ for } j = i-2, i-1, i+1, i+2$$

and 0 for $|j - i| > 2$**.** $P_{i,i}$ **is set to ensure that the row sums of** P **are 1.**

To verify that the Markov chain is reversible, show that

$$\pi_i P_{i,i+2} = \pi_{i+2} P_{i+2,i}$$

and so on.

This is an example of an infinite-state time-reversible Markov chain. Note that the steady-state vector has infinite length and has i**th entry** π_i**.**

Simulating from the Infinite State Markov Chain

Example:

Suppose $\pi_i = k/(i+1)^4$ **for** $i > 0$ **and** $\pi_i = 0$ **for all** $i < 1$. k **is a constant that ensures that** $\sum_{i=1}^{\infty} \pi_i = 1$. **Note that we can simulate from this Markov chain even without knowing** k.

```
pi.fun <- function(i) {
    out <- 0
    if (i > 0) out <- 1/(i+1)^4
    out
}
```

Simulating from the Infinite State Markov Chain

```r
Ntransitions <- 20000
X <- numeric(Ntransitions)
current.state <- 50  # initialize the Markov chain
for (n in 1:Ntransitions) {
     i <- current.state
     P <- c(min(pi.fun(i-2)/pi.fun(i), 1),
     min(pi.fun(i-1)/pi.fun(i), 1),
     min(pi.fun(i+1)/pi.fun(i), 1),
     min(pi.fun(i+2)/pi.fun(i), 1))/6
     P0 <- 1 - sum(P)
     P <- c(P[1:2], P0, P[3:4])
     transition <- sample(seq(-2,2,1), size = 1, prob = P)
     current.state <- current.state + transition
     X[n] <- current.state
}
observedDist <- table(X[-c(1:1000)])
```

Simulating from the Infinite State Markov Chain

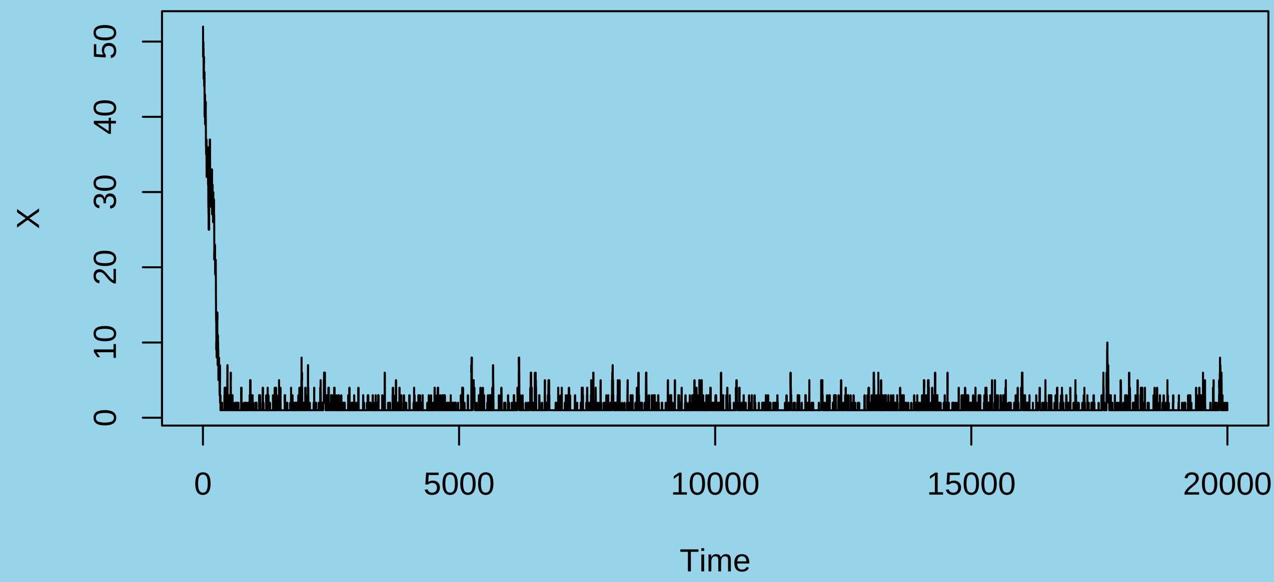
```
observedDist

##
##     1     2     3     4     5     6     7     8     9     1
## 14662  2865   846   334   160    94    18    14     3
```

Burn-In

Why omit the first 1000 observations?

```
ts.plot(X)
```



Estimating k

$$\pi_2 = k/(2+1)^4 = k/81$$

so an estimate of k can be obtained by multiplying the observed probability of a 2 by 81:

```
k <- observedDist[2]/19000*81
k

##    2
## 12.2
```

Markov Chain Monte Carlo Simulation

This procedure is one version of MCMC – developed by Metropolis and Hastings.

Procedure:

1. **Given a distribution π, known up to a proportionality constant (k), find a time-reversible Markov chain with π as the steady state vector.**

2. **Simulate from that Markov chain.**

3. **After simulating for a long enough period (burn-in), the observed states follow the steady state distribution, i.e. π.**

Markov Chain Monte Carlo Simulation

The Law of Large Numbers for regular Markov chains allows us to estimate quantities such as $E[X]$ and $E[g(X)]$ for given functions $g(x)$ by calculating

$$\frac{1}{N}\sum_{n=1}^{N} X_n \text{ and } \frac{1}{N}\sum_{n=1}^{N} g(X_n).$$

MCMC Application - Bayesian Statistics

Example:

Suppose N is Poisson distributed with mean 20, and given N, X is binomially distributed with parameters N and $p = 0.5$.

N is not observed, but suppose $X = 5$. Use MCMC to simulate the distribution of N, given X.

Terminology: the Poisson distribution for N is the *prior* distribution.

the distribution of N, given $X = 5$, is called the *posterior* distribution.

MCMC Application - Bayesian Statistics

```
posterior.fun <- function(i, x) {
    out <- 0
    if (i >= x) out <- dpois(i, lambda = 20)*
                dbinom(x, size = i, prob = .5)
    out
}
```

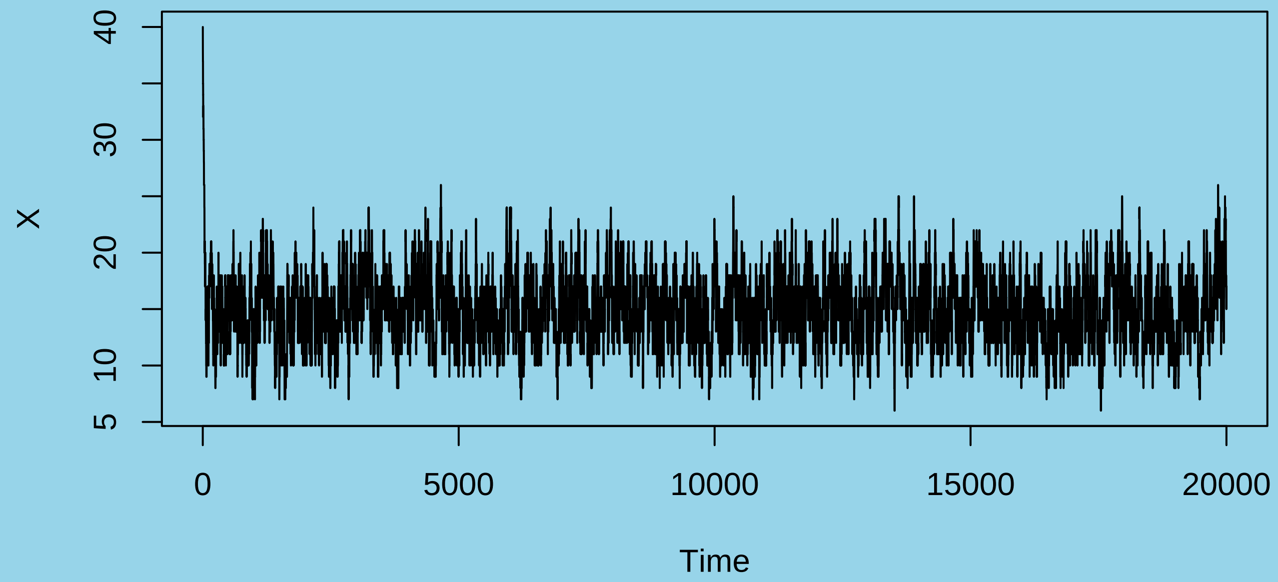
```
pi.fun <- function(i) {
    posterior.fun(i, x=5)
}
```

Simulating the Markov Chain

```r
Ntransitions <- 20000
X <- numeric(Ntransitions)
current.state <- 40  # initialize the Markov chain
for (n in 1:Ntransitions) {
     i <- current.state
     P <- c(min(pi.fun(i-2)/pi.fun(i), 1),
                min(pi.fun(i-1)/pi.fun(i), 1),
                min(pi.fun(i+1)/pi.fun(i), 1),
                min(pi.fun(i+2)/pi.fun(i), 1))/6
     P0 <- 1 - sum(P)
     P <- c(P[1:2], P0, P[3:4])
     transition <- sample(seq(-2,2,1), size = 1, prob = P)
     current.state <- current.state + transition
     X[n] <- current.state
}
observedDist <- table(X[-c(1:1000)])
```

Plotting the Trace

```
ts.plot(X)
```



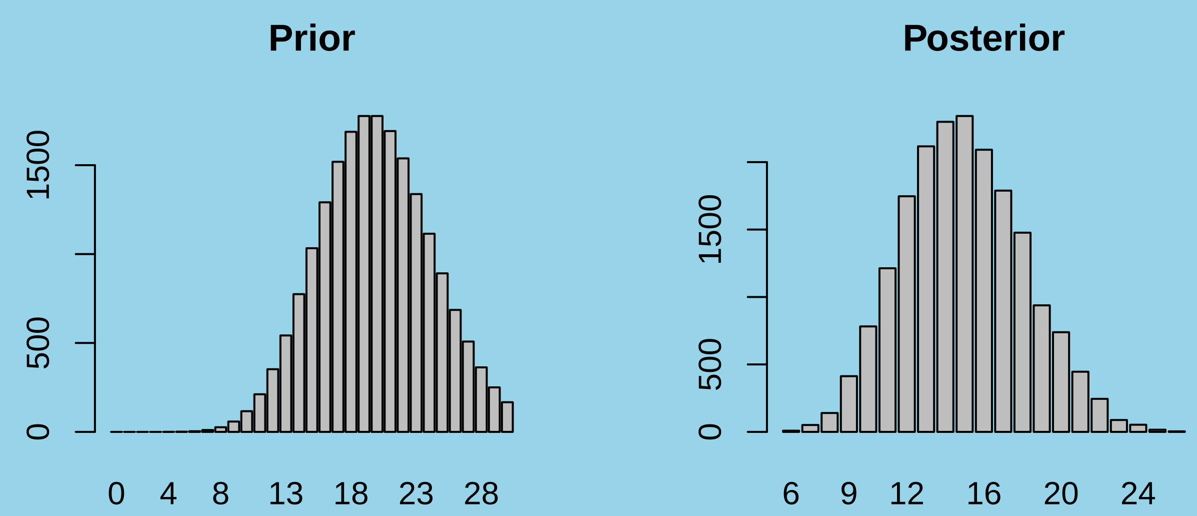
Posterior Distribution of N

```
options(width=50)
observedDist

##
##    6    7    8    9   10   11   12   13   14   15
##    9   51  140  413  782 1213 1747 2117 2299 2342
##   16   17   18   19   20   21   22   23   24   25
## 2092 1789 1477  938  739  446  245   88   53   16
##   26
##    4
```

Posterior Distribution of N

```
par(mfrow=c(1,2))
theoryDist <- 20000*dpois(0:30, lambda = 20)
names(theoryDist) <- 0:30
barplot(theoryDist, main = "Prior")
barplot(observedDist, main = "Posterior")
```



This is how the *data* $X = 5$ influences our *belief* (initially, Poisson(20)) about the distribution of the unknown value N.

What if our Prior Belief was Different?

e.g. $\lambda = 4$:

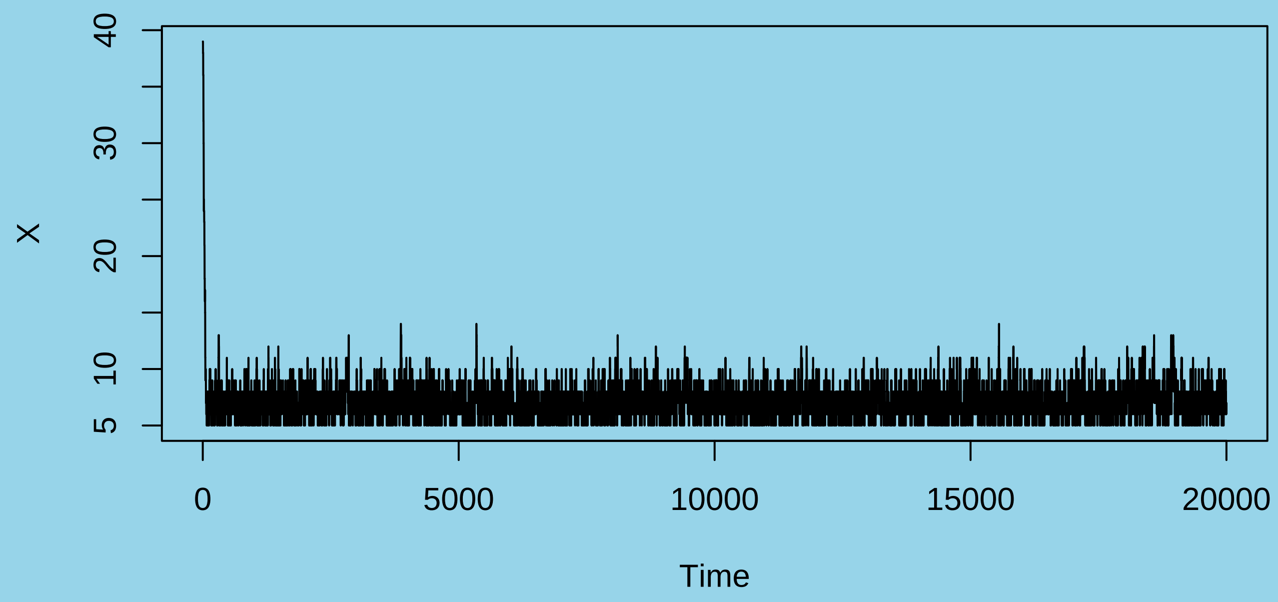
```
posterior.fun <- function(i, x) {
    out <- 0
    if (i >= x) out <- dpois(i, lambda = 4)*
            dbinom(x, size = i, prob = .5)
    out
}
```

Simulating the Markov Chain

```
Ntransitions <- 20000
X <- numeric(Ntransitions)
current.state <- 40  # initialize the Markov chain
for (n in 1:Ntransitions) {
     i <- current.state
     P <- c(min(pi.fun(i-2)/pi.fun(i), 1),
                min(pi.fun(i-1)/pi.fun(i), 1),
                min(pi.fun(i+1)/pi.fun(i), 1),
                min(pi.fun(i+2)/pi.fun(i), 1))/6
     P0 <- 1 - sum(P)
     P <- c(P[1:2], P0, P[3:4])
     transition <- sample(seq(-2,2,1), size = 1, prob = P)
     current.state <- current.state + transition
     X[n] <- current.state
}
observedDist <- table(X[-c(1:1000)])
```

Plotting the Trace

```
ts.plot(X)
```



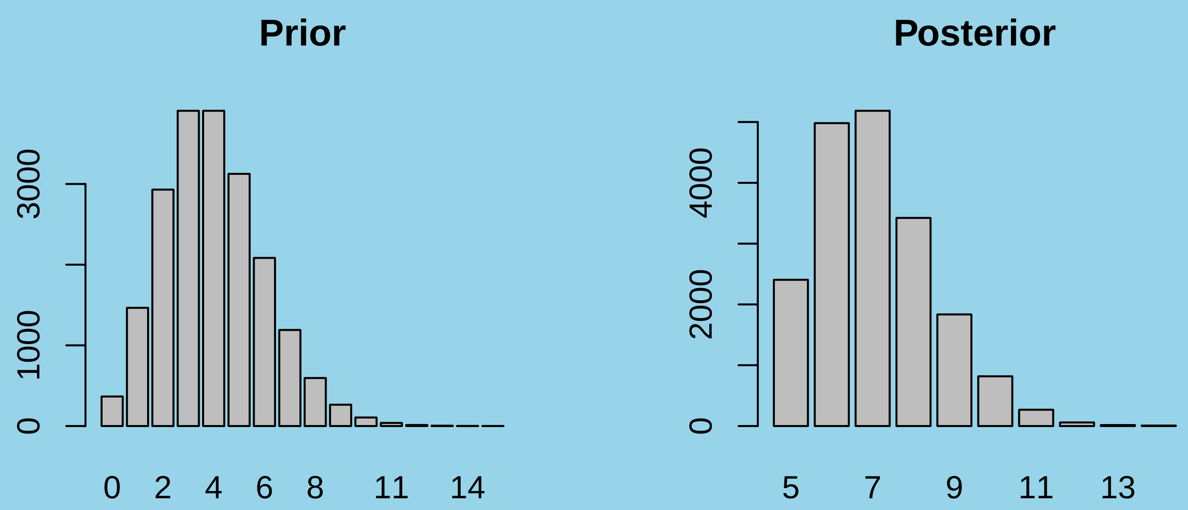
Posterior Distribution of N

```
options(width=50)
observedDist

##
##    5    6    7    8    9   10   11   12   13   14
## 2405 4982 5185 3423 1836  818  268   59   16    8
```

Posterior Distribution of N

```
par(mfrow=c(1,2))
theoryDist <- 20000*dpois(0:15, lambda = 4)
names(theoryDist) <- 0:15
barplot(theoryDist, main = "Prior")
barplot(observedDist, main = "Posterior")
```



This is how the *data* $X = 5$ influences our *belief* (initially, Poisson(4)) about the distribution of the unknown value N.

Using Built-In Software

Perhaps the best way to do MCMC in R is with the `metrop()` function in C. Geyer's *mcmc* package:

```
metrop(obj, initial, nbatch, blen = 1, nspac = 1,
    scale = 1, outfun, debug = FALSE, ...)
```

Main Arguments:

- `obj`: **an R function which evaluates the unnormalized posterior distribution or the result of a previous call to this function.**
- `initial`: **the initial state of the Markov chain.**
- `scale`: **controls the proposal step size in the random walk used for the Markov chain.**