Constructing a Markov chain

A Markov chain is a sequence of random variables

$$X_0, X_1, X_2, \ldots, X_t, \ldots$$

defined so that the distribution of X_t conditional on the preceding variables $X_0, X_1, \ldots, X_{t-1}$ depends *only* on the value of X_{t-1}.

We think about generating the sequence in order

$$X_0 \longrightarrow X_1 \longrightarrow X_2 \longrightarrow \cdots$$

Constructing a Markov chain

The simplest type of chain is defined so that each variable takes values only on a *finite* set

$$\mathcal{S}_X = \{1, 2, \ldots, d\}$$

say: each *marginal* distribution

$$\Pr[X_t = x_t]$$

and each *conditional* distribution

$$\Pr[X_t = x_t | X_{t-1} = x_{t-1}]$$

is simply a discrete distribution $\mathcal{S}_X$, that is, a collection of d probabilities that sum to 1.

Constructing a Markov chain

Suppose that

$$\Pr[X_t = j | X_{t-1} = i] = P_{ij} \qquad i,j = 1,\ldots,d.$$

then we need that

$$\sum_{j=1}^{d} P_{ij} = 1 \qquad \text{for each } i$$

The *transition matrix* P is the matrix of the P_{ij} values.

Note that the probabilities do not depend on t.

Constructing a Markov chain

Suppose also that the *initial* ($t = 0$) probabilities are

$$\Pr[X_0 = i] = p_i^{(0)} \qquad i = 1, \ldots, d$$

that is, the distribution of X_0 is the discrete distribution

$$p^{(0)} = (p_1^{(0)}, \ldots, p_d^{(0)}).$$

Constructing a Markov chain

Note that we can use the Theorem of Total Probability to conclude that

$$\Pr[X_t = j] = \sum_{i=1}^{d} \Pr[X_{t-1} = i] \Pr[X_t = j | X_{t-1} = i]$$

that is

$$p_j^{(t)} = \sum_{i=1}^{d} p_i^{(t-1)} P_{ij}$$

so in vector form

$$p^{(t)} = p^{(t-1)} P$$

Constructing a Markov chain

Suppose $d = 4$, and

$$P = \begin{bmatrix} 0.1 & 0.3 & 0.1 & 0.5 \\ 0.3 & 0.4 & 0.0 & 0.3 \\ 0.3 & 0.1 & 0.2 & 0.4 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{bmatrix} \qquad p^{(0)} = \begin{bmatrix} 0.2 & 0.1 & 0.2 & 0.5 \end{bmatrix}$$

That is

$$\Pr[X_t = 2 | X_{t-1} = 1] = 0.3$$
$$\Pr[X_t = 3 | X_{t-1} = 2] = 0.0$$
$$\Pr[X_t = 4 | X_{t-1} = 4] = 0.1$$

etc.

Constructing a Markov chain

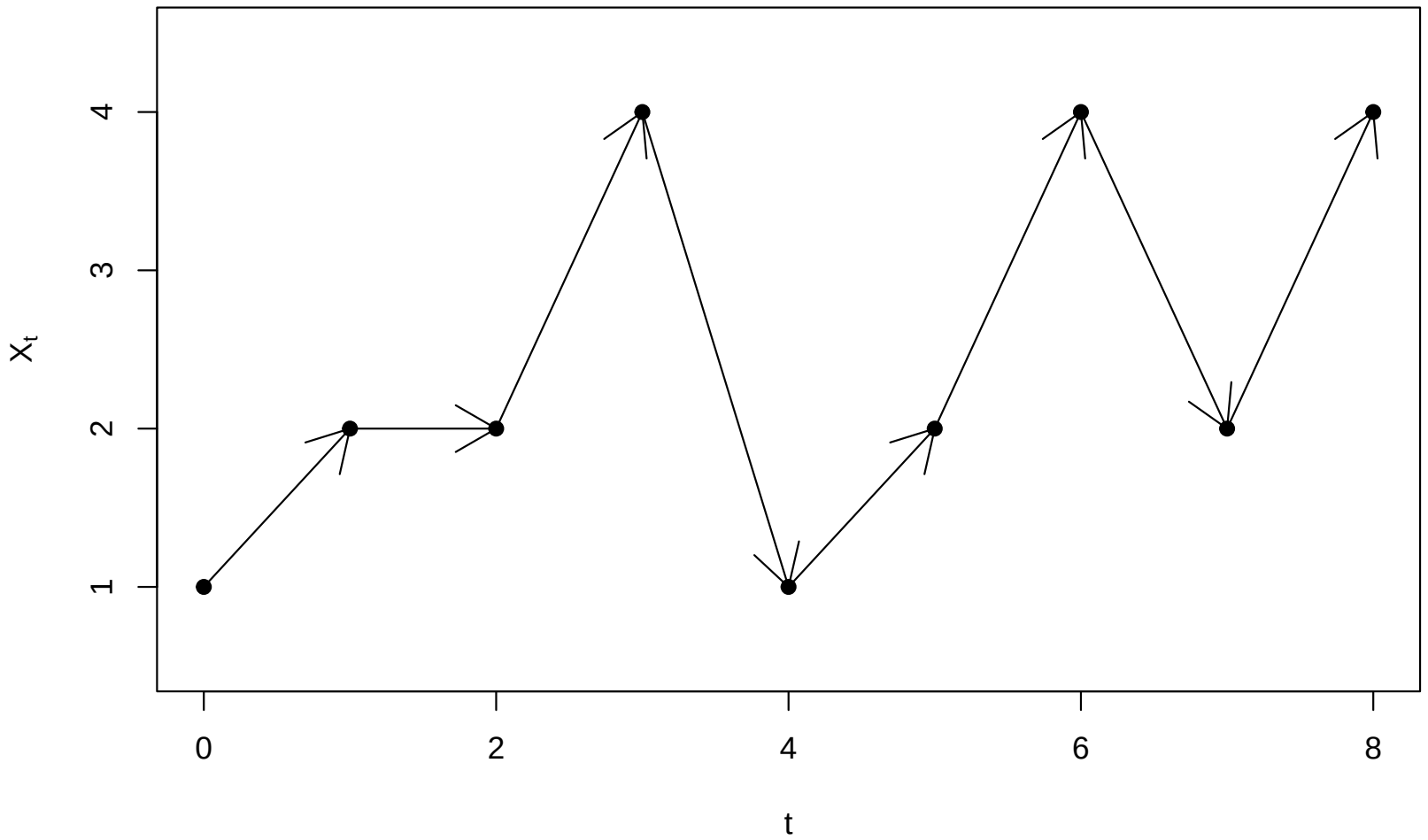
```
par(mar=c(4,4,3,1))
n<-8
d<-4
p.t<-matrix(0,nrow=n+1,ncol=d)
p.t[1,]<-c(0.2,0.1,0.2,0.5)
P<-matrix(0,d,d)
P[1,]<-c(0.1,0.3,0.1,0.5)
P[2,]<-c(0.3,0.4,0.0,0.3)
P[3,]<-c(0.3,0.1,0.2,0.4)
P[4,]<-c(0.3,0.3,0.3,0.1)
P
```

```
+      [,1] [,2] [,3] [,4]
+ [1,]  0.1  0.3  0.1  0.5
+ [2,]  0.3  0.4  0.0  0.3
+ [3,]  0.3  0.1  0.2  0.4
+ [4,]  0.3  0.3  0.3  0.1
```

Constructing a Markov chain

```
#One sample run
set.seed(234)
X.t<-rep(0,n+1)
X.t[1]<-sample(1:d,size=1,prob=p.t[1,])
for(i in 2:(n+1)){
    X.t[i]<-sample(1:d,size=1,prob=P[X.t[i-1],])
    p.t[i,]<-p.t[i-1,] %*% P
}
X.t

+ [1] 1 2 2 4 1 2 4 2 4

plot(0:n,X.t,pch=19,
    xlab='t',ylab=expression(X[t]),ylim=range(0.5,4.5))
arrows(x0=0:(n-1),y0=X.t[1:n],x1=1:n,y1=X.t[2:(n+1)])
```

Constructing a Markov chain



Constructing a Markov chain

For the state distribution $p^{(t)}$ for $t = 0, 1, \ldots, 8$

```
rownames(p.t)<-paste('t=',0:8,sep='')
p.t

+           [,1]      [,2]      [,3]      [,4]
+ t=0 0.2000000 0.1000000 0.2000000 0.5000000
+ t=1 0.2600000 0.2700000 0.2100000 0.2600000
+ t=2 0.2480000 0.2850000 0.1460000 0.3210000
+ t=3 0.2504000 0.2993000 0.1503000 0.3000000
+ t=4 0.2499200 0.2998700 0.1451000 0.3051100
+ t=5 0.2500160 0.3009670 0.1455450 0.3034720
+ t=6 0.2499968 0.3009877 0.1451522 0.3038633
+ t=7 0.2500006 0.3010683 0.1451891 0.3037419
+ t=8 0.2499999 0.3010690 0.1451605 0.3037707
```

We can see that this distribution is becoming a *constant* vector; it takes a few steps to become approximately constant.

Constructing a Markov chain

The *stationary* or *invariant* distribution, π, is a discrete distribution that solves

$$\pi = \pi P.$$

This is an *eigenvalue* problem:

```
eigP<-eigen(P)                                  #Eigen decomposition
eigvalsP<-eigP$values                           #Eigenvalues
eigvP<-eigP$vectors                             #Right eigenvectors
eigvlP<-solve(eigvP)                            #Left eigenvectors
P - eigvP %*% diag(eigvalsP) %*% eigvlP   #Check

+                [,1]           [,2]          [,3]          [,4]
+ [1,]  8.326673e-17 -3.885781e-16  0.000000e+00 3.885781e-16
+ [2,] -5.551115e-17 -4.440892e-16 -6.938894e-17 0.000000e+00
+ [3,]  0.000000e+00 -3.608225e-16 -1.387779e-16 1.110223e-16
+ [4,]  1.110223e-16 -1.665335e-16  5.551115e-17 6.938894e-17
```

Constructing a Markov chain

To find the stationary distribution we select the eigenvector corresponding to the eigenvalue equal to 1 (which here is the *first* eigenvalue in R), suitably standardized

```
eigvalsP

+ [1]  1.0000000  0.2645751 -0.2645751 -0.2000000

pi.vec<-eigvlP[1,]/sum(eigvlP[1,])
pi.vec                                  #Stationary distribution

+ [1] 0.2500000 0.3010753 0.1451613 0.3037634

pi.vec %*% P                            #Verifying the solution

+      [,1]      [,2]      [,3]      [,4]
+ [1,] 0.25 0.3010753 0.1451613 0.3037634
```

Constructing a Markov chain

Note also that by recursion

$$p^{(t)} = p^{(t-1)}P = p^{(t-2)}P^2 = \cdots = p^{(1)}P^{t-1} = p^{(0)}P^t$$

so if we observe

$$p^{(t)} \longrightarrow \pi$$

as $t \longrightarrow \infty$, whatever the value of $p^{(0)}$, then this implies that we can compute π by inspecting P^t as $t \longrightarrow \infty$, that is

$$P^t \longrightarrow \mathbf{1}\pi$$

where $\mathbf{1}$ is the $d \times 1$ vector of 1s.

Constructing a Markov chain

```
P.t<-array(0,c(1000,d,d))
P.t[1,,]<-P
for(k in 2:1000){
    P.t[k,,]<-P %*% P.t[k-1,,]
}
P.t[1000,,]

+      [,1]      [,2]      [,3]      [,4]
+ [1,] 0.25 0.3010753 0.1451613 0.3037634
+ [2,] 0.25 0.3010753 0.1451613 0.3037634
+ [3,] 0.25 0.3010753 0.1451613 0.3037634
+ [4,] 0.25 0.3010753 0.1451613 0.3037634

pi.vec

+ [1] 0.2500000 0.3010753 0.1451613 0.3037634
```

Constructing a Markov chain

We can use the Markov chain to simulate values from π by collecting the X_t values over a long run. Here $n = 10000$.

```
n<-10000
X.t<-rep(0,n+1)
X.t[1]<-sample(1:d,size=1,prob=p.t[1,])
for(i in 2:(n+1)){
    X.t[i]<-sample(1:d,size=1,prob=P[X.t[i-1],])
}
table(X.t)/n

+ X.t
+      1      2      3      4
+ 0.2470 0.3046 0.1429 0.3056

pi.vec

+ [1] 0.2500000 0.3010753 0.1451613 0.3037634
```

Constructing a Markov chain

The X_t values are *correlated*, for example

```
cor(X.t[-1],X.t[-n])

+ [1] -0.2036072
```

so the collected samplea are *not independent*; however, this does not prevent us from using them for Monte Carlo calculations.

Constructing a Markov chain

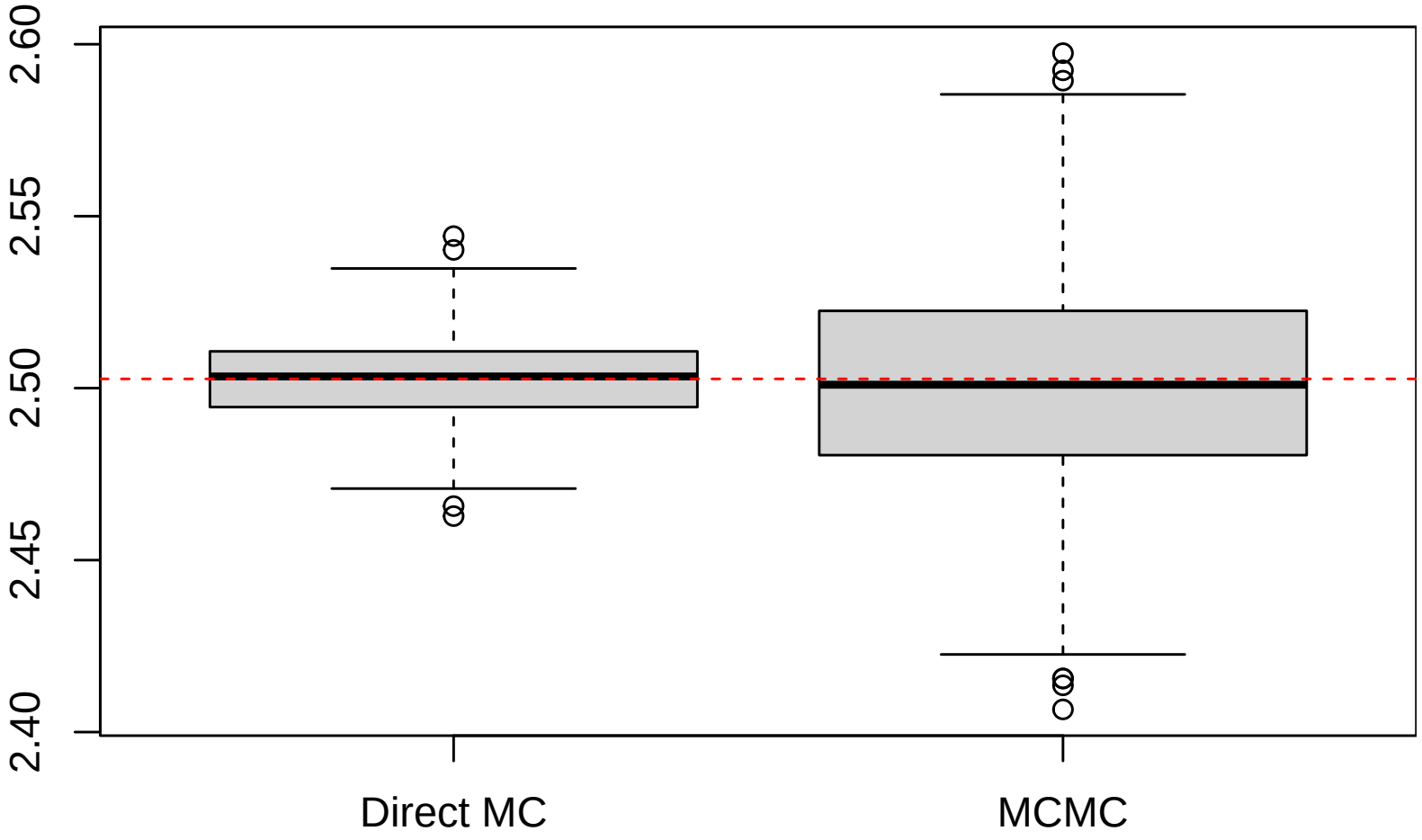
```
#Mean of pi
mu<-sum(c(1:d)*pi.vec)
mu

+ [1] 2.502688

N<-1000
set.seed(34)
#Estimating using direct Monte Carlo
direct.MC<-replicate(N,mean(sample(1:d,size=n,rep=T,prob=pi.vec)))

#Estimating using Markov chain Monte Carlo
mcMC.run<-function(nr,p0v,Pm){
    Xt<-rep(0,nr+1)
    dv<-ncol(Pm)
    Xt[1]<-sample(1:dv,size=1,prob=p0v)
    for(i in 2:(nr+1)){
        Xt[i]<-sample(1:dv,size=1,prob=Pm[Xt[i-1],])
    }
    return(Xt)
}
MC.MC<-replicate(N,mean(mcMC.run(N,pi.vec,P)))
```

Constructing a Markov chain

We can compare the estimated values over 1000 replicated runs using direct Monte Carlo and Markov chain Monte Carlo.

```
par(mar=c(4,4,3,0))
boxplot(cbind(direct.MC,MC.MC),names=c('Direct MC','MCMC'))
abline(h=mu,col='red',lty=2)
```

Constructing a Markov chain



Constructing a Markov chain

```
n*c(var(direct.MC),var(MC.MC))

+ [1] 1.399938 9.896244

var(direct.MC)/var(MC.MC)

+ [1] 0.1414616
```

Here the Markov chain method is more variable than the direct method.