Assignment Sheet: Markov Chains — Model Answers

1. $\Rightarrow$ Denote by $\mathcal{A} = \{t_k+1,\ldots,t\}$, by $\mathcal{C} = \{t_1,\ldots t_k\}$, and by $\mathcal{B} = \{1,\ldots,t_k\} \setminus \mathcal{C}$. Then

$$
\begin{aligned}
&\mathbb{P}(X_{t+1}=x_{t+1}|X_{t_k}=x_{t_k},\ldots,X_{t_1}=x_{t_1}) = \mathbb{P}\left(X_{t+1}=x_{t+1}\middle|\bigcap_{i\in\mathcal{C}}\{X_i=x_i\}\right)\\
&= \sum_{\mathbf{x}_\mathcal{B}\in S^{|\mathcal{B}|}} \mathbb{P}\left(X_{t+1}=x_{t+1}\cap\bigcap_{i\in\mathcal{B}}\{X_i=x_i\}\middle|\bigcap_{i\in\mathcal{C}}\{X_i=x_i\}\right)\\
&= \sum_{\mathbf{x}_\mathcal{B}\in S^{|\mathcal{B}|}} \mathbb{P}\left(\bigcap_{i\in\mathcal{B}}\{X_i=x_i\}\middle|\bigcap_{i\in\mathcal{C}}\{X_i=x_i\}\right)\cdot\mathbb{P}\left(X_{t+1}=x_{t+1}\middle|\bigcap_{i\in\mathcal{B}\cup\mathcal{C}}\{X_i=x_i\}\right)\\
&= \sum_{\mathbf{x}_\mathcal{A}\in S^{|\mathcal{A}|}}\sum_{\mathbf{x}_\mathcal{B}\in S^{|\mathcal{B}|}} \mathbb{P}\left(\bigcap_{i\in\mathcal{B}}\{X_i=x_i\}\middle|\bigcap_{i\in\mathcal{C}}\{X_i=x_i\}\right)\cdot \underbrace{\mathbb{P}\left(\bigcap_{i\in\mathcal{A}\cup\{t+1\}}\{X_i=x_i\}\middle|\bigcap_{i\in\mathcal{B}\cup\mathcal{C}}\{X_i=x_i\}\right)}_{\substack{=\mathbb{P}\left(X_{t+1}=x_{t+1},\ldots,X_{t_k+1}=x_{t_k+1}\middle|X_{t_k}=x_{t_k},\ldots,X_0=x_0\right)\\ =\mathbb{P}\left(X_{t+1}=x_{t+1},\ldots,X_{t_k+1}=x_{t_k+1}\middle|X_{t_k}=x_{t_k}\right)}}\\
&= \underbrace{\sum_{\mathbf{x}_\mathcal{A}\in S^{|\mathcal{A}|}} \mathbb{P}\left(X_{t+1}=x_{t+1},\ldots,X_{t_k+1}=x_{t_k+1}\middle|X_{t_k}=x_{t_k}\right)}_{=\mathbb{P}(X_{t+1}=x_{t+1}|X_{t_k}=x_{t_k})}\cdot\underbrace{\sum_{\mathbf{x}_\mathcal{B}\in S^{|\mathcal{B}|}}\mathbb{P}\left(\bigcap_{i\in\mathcal{B}}\{X_i=x_i\}\middle|\bigcap_{i\in\mathcal{C}}\{X_i=x_i\}\right)}_{=1}\\
&= \mathbb{P}(X_{t+1}=x_{t+1}|X_{t_k}=x_{t_k})
\end{aligned}
$$

$\Leftarrow$ Set $k=t+1$, $t_k=t$, $t_{k-1}=t-1$, …, $t_0=0$. Then the statement on the assignment sheet is identical to the Markov property.

2. (a) We use induction. For $m=1$ we have that

$$
\begin{pmatrix} \frac{\beta+\alpha(1-\alpha-\beta)^1}{\alpha+\beta} & \frac{\alpha-\alpha(1-\alpha-\beta)^1}{\alpha+\beta} \\ \frac{\beta-\beta(1-\alpha-\beta)^1}{\alpha+\beta} & \frac{\alpha+\beta(1-\alpha-\beta)^1}{\alpha+\beta} \end{pmatrix} = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} = \mathbf{K}^{(1)} = \mathbf{K}
$$

Suppose for m we have that

$$
\mathbf{K}^{(m)} = \begin{pmatrix} \frac{\beta+\alpha(1-\alpha-\beta)^m}{\alpha+\beta} & \frac{\alpha-\alpha(1-\alpha-\beta)^m}{\alpha+\beta} \\ \frac{\beta-\beta(1-\alpha-\beta)^m}{\alpha+\beta} & \frac{\alpha+\beta(1-\alpha-\beta)^m}{\alpha+\beta} \end{pmatrix},
$$

then

$$
\begin{aligned}
\mathbf{K}^{(m+1)} = \mathbf{K}^{(m)}\mathbf{K} &= \begin{pmatrix} \frac{\beta+\alpha(1-\alpha-\beta)^m}{\alpha+\beta} & \frac{\alpha-\alpha(1-\alpha-\beta)^m}{\alpha+\beta} \\ \frac{\beta-\beta(1-\alpha-\beta)^m}{\alpha+\beta} & \frac{\alpha+\beta(1-\alpha-\beta)^m}{\alpha+\beta} \end{pmatrix} \cdot \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}\\
&= \begin{pmatrix} \frac{\beta+\alpha(1-\alpha-\beta)^m}{\alpha+\beta}(1-\alpha) + \frac{\alpha-\alpha(1-\alpha-\beta)^m}{\alpha+\beta}\beta & \frac{\beta+\alpha(1-\alpha-\beta)^m}{\alpha+\beta}\alpha + \frac{\alpha-\alpha(1-\alpha-\beta)^m}{\alpha+\beta}(1-\beta) \\ \frac{\beta-\beta(1-\alpha-\beta)^m}{\alpha+\beta}(1-\alpha) + \frac{\alpha+\beta(1-\alpha-\beta)^m}{\alpha+\beta}\beta & \frac{\beta-\beta(1-\alpha-\beta)^m}{\alpha+\beta}\alpha + \frac{\alpha+\beta(1-\alpha-\beta)^m}{\alpha+\beta}(1-\beta) \end{pmatrix}\\
&= \begin{pmatrix} \frac{\beta+\alpha(1-\alpha-\beta)^{m+1}}{\alpha+\beta} & \frac{\alpha-\alpha(1-\alpha-\beta)^{m+1}}{\alpha+\beta} \\ \frac{\beta-\beta(1-\alpha-\beta)^{m+1}}{\alpha+\beta} & \frac{\alpha+\beta(1-\alpha-\beta)^{m+1}}{\alpha+\beta} \end{pmatrix}
\end{aligned}
$$

(b) To find the invariant distribution we need to solve $\boldsymbol{\mu}' = \boldsymbol{\mu}'\mathbf{K}$, i.e. $(\mathbf{K}-\mathbf{I})'\boldsymbol{\mu} = \mathbf{0}$:

$$
\begin{pmatrix} -\alpha & \beta \\ \alpha & -\beta \end{pmatrix} \cdot \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}
$$

Thus $(\mu_1,\mu_2)' \propto (\beta,\alpha)'$, thus

$$
\boldsymbol{\mu} = \left(\frac{\beta}{\alpha+\beta}, \frac{\alpha}{\alpha+\beta}\right)'
$$

(c) We have that $\mu_1 k_{12} = \frac{\beta}{\alpha+\beta}\alpha = \frac{\alpha}{\alpha+\beta}\beta = \mu_2 k_{21}$, thus $\mathbf{K}$ is in detailed balance with $\boldsymbol{\mu}$, which makes X time-reversible (if initialised according to $\boldsymbol{\mu}$).

3. The communicating classes are: $\{5\}$ (transient, aperiodic), $\{10\}$ (transient, aperiodic), $\{4, 9\}$ (transient, aperiodic), $\{1, 2, 3, 6, 7, 8\}$ (recurrent, periodic with period 2).

4. (a) All states communicate with 0, as $k_{i0} = \frac{1}{2} > 0$ and $k_{0i}^{(i)} = \frac{1}{2^i} > 0$, thus all states communicate, i.e. the chain is irreducible.

(b) We have for all i that $\mathbb{P}(X_{t+1} = 0 | X_t = i) = \frac{1}{2}$. Thus

$$k_{00}^{(t)} = \mathbb{P}(X_t = 0 | X_0 = 0) = \sum_{i\in\mathbb{N}_0} \mathbb{P}(X_t = 0, X_{t-1} = i | X_0 = 0) = \sum_{i\in\mathbb{N}_0} \underbrace{\mathbb{P}(X_t = 0 | X_{t-1} = i)}_{=\frac{1}{2}} \mathbb{P}(X_{t-1} = i | X_0 = 0)$$

$$= \frac{1}{2} \underbrace{\sum_{i\in\mathbb{N}_0} \mathbb{P}(X_{t-1} = i | X_0 = 0)}_{=1} = \frac{1}{2}$$

Thus $\mathbb{E}(V_0 | X_0 = 0) = \sum_{t=0}^{+\infty} k_{ii}^{(t)} = \sum_{t=0}^{+\infty} \frac{1}{2} = +\infty$, thus the state 0 is recurrent. Due to the irreducibility of the chain, all other states must then be recurrent as well.

(c) We have to find the distribution $\boldsymbol{\mu}$ such that $\boldsymbol{\mu}'\mathbf{K} = \boldsymbol{\mu}'$ where

$$\mathbf{K} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & \dots \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \dots \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

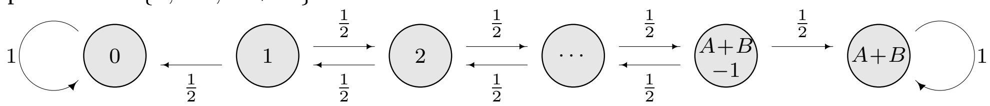
Thus

$$\mu_0 = \sum_{i\in\mathbb{N}_0} \frac{1}{2}\mu_i = \frac{1}{2} \underbrace{\sum_{i\in\mathbb{N}_0} \mu_i}_{=1} = \frac{1}{2}$$

and $\mu_i = \frac{1}{2}\mu_{i-1}$, and

$$\boldsymbol{\mu} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \dots\right).$$

5. (a) The state space is $S = \{0, \dots, A + B\}$.



(b) The communicating classes are $\{0\}$ (recurrent), $\{1, \dots, A+B-1\}$ (transient), and $\{A+B\}$ (recurrent). As the transient class $\{1, \dots, A + B - 1\}$ is finite, the Markov chain will at some point leave this class forever, i.e. hit 0 or $A + B$, in which case either you or your friend are bankrupt, and the game is over.

(c) We have that H_t does not depend on t:

$H_t \subset H_{t-1}$**:** If $X_\tau = 0$ for some $\tau > t$, then we also have that $X_\tau = 0$ for some $\tau > t - 1$.

$H_{t-1} \subset H_t$**:** If $X_\tau = 0$ for some $\tau > t - 1$. Then also $X_{\tau+1} = X_{\tau+2} = \ldots = 0$ (because of $k_{00} = 1$). Thus $X_\tau = 0$ for some $\tau > t$.

Thus $H_0 = H_1 = \ldots = H_{t-1} = H_t = \ldots$.

As the states 0 and $A + B$ are absorbing (i.e. we cannot leave them), we have that

$$h_0 = \mathbb{P}(H_t | X_t = 0) = 1 \qquad\qquad h_{A+B} = \mathbb{P}(H_t | X_t = A + B) = 0$$

Furthermore, for $i \in \{1, \dots, A + B - 1\}$

$$\begin{aligned} h_i &= \mathbb{P}(H_t | X_t = i) = \mathbb{P}(H_{t+1} | X_t = i) = \mathbb{P}(H_{t+1} \cap \{X_{t+1} = i - 1\} | X_t = i) + \mathbb{P}(H_{t+1} \cap \{X_{t+1} = i + 1\} | X_t = i) \\ &= \underbrace{\mathbb{P}(H_{t+1} | X_{t+1} = i - 1, X_t = i)}_{=\mathbb{P}(H_{t+1}|X_{t+1}=i)} \mathbb{P}(X_{t+1} = i - 1 | X_t = i) + \underbrace{\mathbb{P}(H_{t+1} | X_{t+1} = i + 1, X_t = i)}_{=\mathbb{P}(H_{t+1}|X_{t+1}=i+1)} \mathbb{P}(X_{t+1} = i + 1 | X_t = i) \\ &= h_{i-1} \cdot \frac{1}{2} + = h_{i+1} \cdot \frac{1}{2}. \end{aligned}$$

Thus the vector $\mathbf{h} = (h_0, h_1, \dots, h_{A+B})$ is the solution to the linear system of equations

$$h_0 = 1, \qquad h_i = \frac{1}{2}h_{i-1} + \frac{1}{2}h_{i+1} \text{ for } i \in \{1, \dots, A+B-1\}, \qquad h_{A+B} = 0,$$

i.e.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{1}{2} & -1 & \frac{1}{2} & \dots & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & \dots & \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{A+B-1} \\ h_{A+B} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

The solution to this system of linear equations is $\mathbf{h} = \frac{1}{A+B}(A+B, A+B-1, \dots, 1, 0)$.

Thus, at the begin of the game, the probability that you will loose all your money is

$$\mathbb{P}(H_0 | X_0 = A) = h_A = \frac{A+B-A}{A+B} = \frac{B}{A+B}.$$

6. (a) We have that $S_t = s$ if and only if $X_t = X_{t+1} = \dots = X_{t+s-1}$ and $X_s \neq X_t$. Thus for $s \in \mathbb{N}$

$$\begin{aligned} p(s|X_t = x_t) &= \mathbb{P}(S_t = s | X_t = x_t) = \underbrace{\mathbb{P}(X_{t+1} = x_t | X_t = x_t) \cdots \mathbb{P}(X_{t+s-1} = x_t | X_{t+s-2} = x_t)}_{=(k_{x_t x_t})^{s-1}} \\ &\quad \cdot \underbrace{\mathbb{P}(X_{t+s} \neq x_t | X_{t+s-1} = x_t)}_{=1-k_{x_t x_t}} = (k_{x_t x_t})^{s-1}(1 - k_{x_t x_t}). \end{aligned}$$

(b) We have that

$$\begin{aligned} \mathbb{P}(S_t > s | X_t = x_t) &= 1 - \mathbb{P}(S_t \leq s | X_t = x_t) = 1 - \sum_{r=1}^{s} p(s|X_t = x_t) = 1 - (1 - k_{x_t x_t}) \sum_{r=1}^{s} (k_{x_t x_t})^{s-1} \\ &= 1 - \sum_{r=1}^{s} (k_{x_t x_t})^{s-1} + \sum_{r=1}^{s} (k_{x_t x_t})^{s} = 1 - (k_{x_t x_t})^0 + (k_{x_t x_t})^s = (k_{x_t x_t})^s \end{aligned}$$

Thus

$$\mathbb{P}(S_t > s + s_0 | S_t > s_0, X_t = x_t) = \frac{\mathbb{P}(S_t > s + s_0 | X_t = x_t)}{\mathbb{P}(S_t > s_0 | X_t = x_t)} = \frac{(k_{x_t x_t})^{s+s_0}}{(k_{x_t x_t})^{s_0}} = (k_{x_t x_t})^s = \mathbb{P}(S_t > s | X_t = x_t),$$

yielding

$$\begin{aligned} \mathbb{P}(S_t > s + s_0 | S_t > s_0) &= \sum_{x_t} \mathbb{P}(S_t > s + s_0 | S_t > s_0, X_t = x_t) \mathbb{P}(X_t = x_t) = \sum_{x_t} \mathbb{P}(S_t > s | X_t = x_t) \mathbb{P}(X_t = x_t) \\ &= \mathbb{P}(S_t > s). \end{aligned}$$

7. The following Python program can be used for answering this question.

```
####################################
#   Python code for question 7   #
####################################

from scipy import *
from scipy import linalg
import pylab

# We will be using the code from the computer practicals to compute
# the invariant distribution
def invariant_distribution(K):
    eigen = linalg.eig(K.T)          # Find eigenvals and vecs
    idx = eigen[0].argmax()          # Find which eigenval is largest (i.e. 1)
    mu = eigen[1][:,idx]             # Extract corresponding eigenvec
    mu = mu / sum(mu)                # Normalise distribution
    return mu

# Define cross-tabulation matrix
```

```
C = array([ [50., 19., 26., 8., 7., 11., 6., 2],
            [16., 40., 34., 18., 11., 20., 8., 3],
            [12., 35., 65., 66., 35., 88., 23., 21],
            [11., 20., 58., 110., 40., 183., 64., 32],
            [2., 8., 12., 23., 25., 46., 28., 12],
            [12., 28., 102., 162., 90., 554., 230., 177],
            [0., 6., 19., 40., 21., 158., 143., 71],
            [0., 3., 14., 32., 15., 126., 91., 106] ])

# Absolute frequencies of occupations of fathers is row-wise sum
fathers = C.sum(1)
# Absolute frequencies of occupations of sons is column-wise sum
sons = C.sum(0)

# Compute the relative frequencies
freq_fathers = fathers / sum(fathers)
freq_sons = sons / sum(sons)
print(freq_fathers)
print(freq_sons)

# To turn C into a transition kernel we need to divide each row by its sum
K = (C.T / fathers).T

# Compute the invariant distribution
mu = invariant_distribution(K)
print(mu)

# Visualise the result using a barplot
pylab.bar(linspace(1,8,8)-.4,freq_fathers,0.25,color="r",
          label="Actual distribution of fathers")
pylab.bar(linspace(1,8,8)-.1,freq_sons,0.25,color="g",
          label="Actual distribution of sons")
pylab.bar(linspace(1,8,8)+.2,mu,0.25,color="b",
          label="Invariant distribution")
pylab.legend(loc=2)
pylab.show()
```

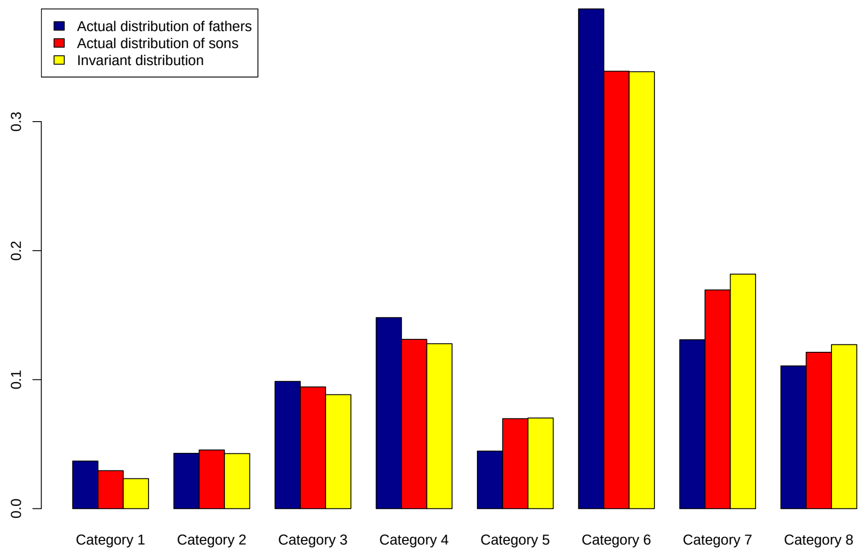
Alternatively, you can use the following R program.

```
###################################
#      R code for question 7      #
###################################

# We will be using the code from the computer practicals to compute
# the invariant distribution
invariant.distribution <- function(K) {
  mu <- eigen(t(K))$vectors[,1]     # Find the eigenvector corresponding
                                    # to the largest eigenvalue (i.e. 1)
  mu <- mu / sum(mu)                # Normalise distribution
  as.numeric(mu)                    # Return mu (with complex part (=0) removed)
}

# Define cross-tabulation matrix
C <- rbind( c(50, 19, 26, 8, 7, 11, 6, 2),
            c(16, 40, 34, 18, 11, 20, 8, 3),
            c(12, 35, 65, 66, 35, 88, 23, 21),
            c(11, 20, 58, 110, 40, 183, 64, 32),
            c(2, 8, 12, 23, 25, 46, 28, 12),
            c(12, 28, 102, 162, 90, 554, 230, 177),
            c(0, 6, 19, 40, 21, 158, 143, 71),
            c(0, 3, 14, 32, 15, 126, 91, 106))

# Absolute frequencies of occupations of fathers is row-wise sum
fathers <- apply(C, 1, sum)
# Absolute frequencies of occupations of sons is column-wise sum
sons <- apply(C, 2, sum)

# Compute the relative frequencies
freq.fathers <- fathers / sum(fathers)
```

```r
freq.sons <- sons / sum(sons)
print(freq.fathers)
print(freq.sons)

# To turn C into a transition kernel we need to divide each row by its sum
K = sweep(C, 1, fathers, "/")

# Compute the invariant distribution
mu = invariant.distribution(K)
print(mu)

# Visualise the result using a barplot
colors <- c("darkblue","red","yellow")
barplot(rbind(freq.fathers,freq.sons,mu), beside=TRUE, col=colors,
        names.arg=paste("Category",1:8))
legend("topleft", fill=colors, legend=c("Actual␣distribution␣of␣fathers",
                                     "Actual␣distribution␣of␣sons",
                                     "Invariant␣distribution"))
```

The results are summarised in the table below:

	Occupational status (category)							
	1	2	3	4	5	6	7	8
Empirical dist'n of fathers	0.036878	0.042882	0.098628	0.148085	0.044597	0.387364	0.130932	0.110635
Empirical dist'n of sons	0.029445	0.045455	0.09434	0.131218	0.069754	0.339051	0.169525	0.121212
Invariant distribution	0.023253	0.042673	0.088362	0.127855	0.070244	0.338673	0.181799	0.127141



Both the distribution of the fathers' occuptational status and the distribution of the sons' occupational status are very close to the invariant distribution, so it appears that the British society was not changuing much in the middle of the 20th century.

8. The following Python program can be used for answering this question.

```python
####################################
#   Python code for question 8   #
####################################

from scipy import *
from scipy import random
import pylab

# Simulates a path of length n+1 from the Wright-Fisher model
# Initial distribution is given by the array initial (length 2)
def simulate_wright_fisher(initial, n):
    result = zeros((n+1,2))                      # Create array to store result
    result[0,:] = initial                        # Set first line to initial dist'n
    two_N = sum(initial)                         # Compute 2*N
    for t in xrange(n):
```

```
        result[t+1,0] = random.binomial(two_N, result[t,0]/two_N)
                                                   # Draw X[t+1]|X[t]=x[t]
        result[t+1,1] = two_N - result[t+1,0]      # Compute 2*N - X[t+1]
    return result


wf_path = simulate_wright_fisher([300,100], 1000)
print wf_path

pylab.figure()
pylab.plot(wf_path[:,0], 'bs-', wf_path[:,1], 'rs-')
pylab.show()
```

Alternatively, you can use the following R program.

```
##################################
#      R code for question 8     #
##################################

# Simulates a path of length n+1 from the Wright-Fisher model
# Initial distribution is given by the array initial (length 2)
simulate.wright.fisher <- function(initial, n) {
  result <- matrix(nrow=n+1, ncol=2)             # Create matrix to store result
  result[1,] <- initial                          # Set first line to initial dist'n
  two.N <- sum(initial)                          # Compute 2*N
  for (t in 1:n) {
    result[t+1,1] <- rbinom(1, size=two.N, prob=result[t,1]/two.N)
                                                # Draw X[t+1]|X[t]=x[t]
    result[t+1,2] <- two.N - result[t+1,1]      # Compute 2*N - X[t+1]
  }
  result
}

wf.path <- simulate.wright.fisher(c(300,100), 1000)
wf.path

matplot(wf.path, type="o", col=3:4, pch=15:16, lty=1,
        xlab="Time", ylab="Count")
legend("topleft", col=3:4, lty=1, pch=15:16,
       legend=c("Allele 1","Allele 2"))
```