

SOLUTIONS MANUAL

FOR

Korosteleva, O. (2022). *Stochastic Processes with R: An Introduction*

By

OLGA KOROSTELEVA

Department of Mathematics and Statistics

California State University, Long Beach

TABLE OF CONTENTS

CHAPTER 1	3
CHAPTER 2	31
CHAPTER 3	41
CHAPTER 4	48
CHAPTER 5	60
CHAPTER 6	67
CHAPTER 7	74
CHAPTER 8	81
CHAPTER 9	87

CHAPTER 1

EXERCISE 1.1. For a Markov chain with a one-step transition probability matrix $\begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \\ 0.8 & 0.1 & 0.1 \end{bmatrix}$ we compute:

- (a) $P(X_3 = 2 | X_0 = 1, X_1 = 2, X_2 = 3) = P(X_3 = 2 | X_2 = 3)$ (by the Markov property)
 $= P_{32} = 0.1.$
- (b) $P(X_4 = 3 | X_0 = 2, X_3 = 1) = P(X_4 = 3 | X_3 = 1)$ (by the Markov property)
 $= P_{13} = 0.3.$
- (c) $P(X_0 = 1, X_1 = 2, X_2 = 3, X_3 = 1) = P(X_3 = 1 | X_0 = 1, X_1 = 2, X_2 = 3) P(X_2 = 3 | X_0 = 1, X_1 = 2) P(X_1 = 2 | X_0 = 1) P(X_0 = 1)$ (by conditioning)
 $= P(X_3 = 1 | X_2 = 3) P(X_2 = 3 | X_1 = 2) P(X_1 = 2 | X_0 = 1) P(X_0 = 1)$ (by the Markov property)
 $= P_{31} P_{23} P_{12} P(X_0 = 1) = (0.8)(0.5)(0.4)(1) = 0.16.$
- (d) We first compute the two-step transition probability matrix. We obtain

$$\mathbf{P}^{(2)} = \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \\ 0.8 & 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \\ 0.8 & 0.1 & 0.1 \end{bmatrix} = \begin{bmatrix} 0.41 & 0.27 & 0.32 \\ 0.52 & 0.22 & 0.26 \\ 0.34 & 0.36 & 0.30 \end{bmatrix}.$$

Now we write

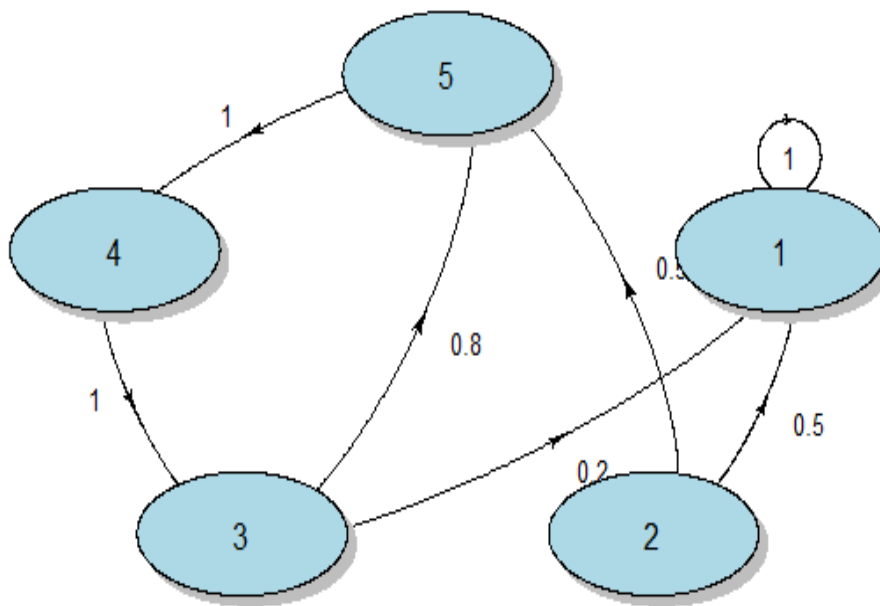
$$\begin{aligned} P(X_0 = 1, X_1 = 2, X_3 = 3, X_5 = 1) &= P(X_5 = 1 | X_0 = 1, X_1 = 2, X_3 = 3) P(X_3 = 3 | X_0 = 1, X_1 = 2) P(X_1 = 2 | X_0 = 1) P(X_0 = 1) \text{ (by conditioning)} \\ &= P(X_5 = 1 | X_3 = 3) P(X_3 = 3 | X_1 = 2) P(X_1 = 2 | X_0 = 1) P(X_0 = 1) \text{ (by the Markov property)} \\ &= P_{31}^{(2)} P_{23}^{(2)} P_{12} P(X_0 = 1) = (0.34)(0.26)(0.4)(1) = 0.03536. \end{aligned}$$

EXERCISE 1.2. (a) We plot a diagram of the Markov chain.

```
#specifying transition probability matrix
tm<- matrix(c(1, 0, 0, 0, 0, 0.5, 0, 0, 0, 0.5, 0.2, 0, 0, 0, 0.8,
0, 0, 1, 0, 0, 0, 0, 0, 1, 0), nrow=5, ncol=5, byrow=TRUE)

#transposing transition probability matrix
tm.tr<- t(tm)

#plotting diagram
library(diagram)
plotmat(tm.tr, arr.length=0.25, arr.width=0.1, box.col="light blue",
box.lwd=1, box.prop=0.5, box.size=0.12, box.type="circle", cex.txt=0.8,
lwd=1, self.cex=0.3, self.shiftx=0.01, self.shifty=0.09)
```



(b) From the diagram, the Markov chain consists of three classes $\{1\}$, $\{2\}$, and $\{3, 4, 5\}$. State 1 is absorbing. Once the chain enters state 1, it cannot leave it, and will forever go through the loop. This makes $\{1\}$ a separate recurrent class. The existence of the loop means that its period is one, making it an aperiodic class.

State 2 is reflective. The chain leaves that state in one step. Therefore, it forms a separate transient class that has an infinite period.

Finally, states 3, 4, and 5 communicate and thus belong to the same class. The chain can return to either state in this class in 3, 6, 9, etc. steps, thus the period is equal to 3. Since there is a positive probability to leave this class, it is transient.

The R output supports these findings.

```

#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm,states=c("1", "2", "3", "4", "5"))

#computing Markov chain characteristics
recurrentClasses(mc)

"1"

transientClasses(mc)

"2"

"3" "4" "5"

absorbingStates(mc)

"1"

```

(c) Below we simulate three trajectories of the chain that start at a randomly chosen state.

```

#specifying total number of steps
nsteps<- 25

#specifying seed
set.seed(4955145)

#specifying initial probability
p0<- c(0.2, 0.2, 0.2, 0.2, 0.2)

#specifying matrix containing states
MC.states<- matrix(NA, nrow=nsteps, ncol=3)

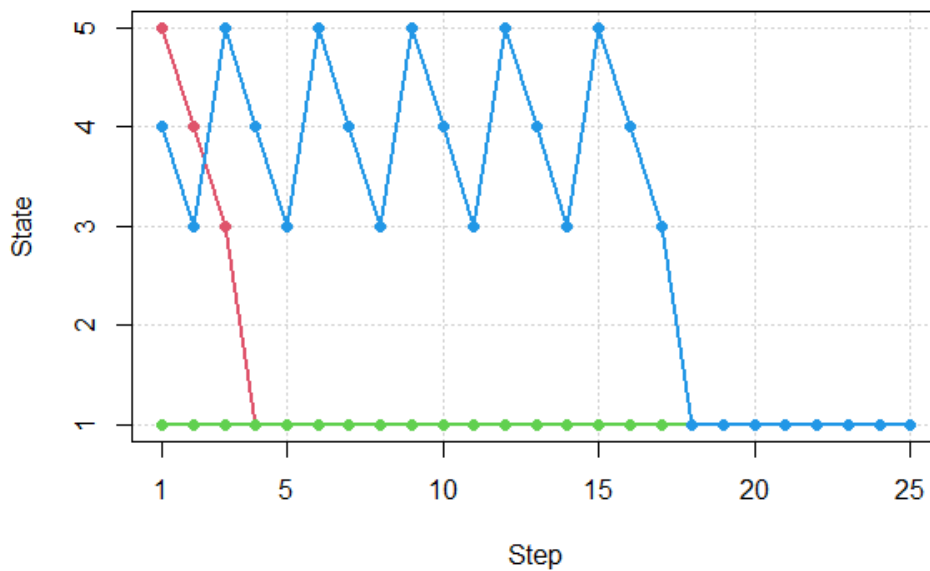
#simulating states
for (i in 1:3) {
  state0<- sample(1:5, 1, prob=p0)
  MC.states[,i]<- rmarkovchain(n=nsteps-1, object=mc, t0=state0,
    include.t0=TRUE)
}

#plotting simulated trajectories
matplot(MC.states, type="l", lty=1, lwd=2, col=2:4, xaxt="n", ylim=c(1,5),
  xlab="Step", ylab="State", panel.first=grid())

axis(side=1, at=c(1,5,10,15,20,25))

points(1:nsteps, MC.states[,1], pch=16, col=2)
points(1:nsteps, MC.states[,2], pch=16, col=3)
points(1:nsteps, MC.states[,3], pch=16, col=4)

```



Since state 1 is an absorbing state, sooner or later, the trajectories transition into this state and don't leave it.

(d) To find the steady-state probabilities, we need to solve the following equations:

$$(\pi_1, \pi_2, \pi_3, \pi_4, \pi_5) = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5) \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.5 \\ 0.2 & 0 & 0 & 0 & 0.8 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \text{ with the additional condition}$$

that $\pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 = 1$.

Written out, the system becomes
$$\begin{cases} \pi_1 = \pi_1 + 0.5\pi_2 + 0.2\pi_3 \\ \pi_2 = 0 \\ \pi_3 = \pi_4 = \pi_5 = 0 \\ \pi_1 + \pi_2 + \pi_3 + \pi_4 + \pi_5 = 1 \end{cases}$$
. It has the degenerate solution

$\pi_1 = 1, \pi_2 = \pi_3 = \pi_4 = \pi_5 = 0$. This solution is expected because state 1 is an absorbing state, and so the chain ends up spending 100% of the time there. Having a unique stationary distribution, it is an ergodic Markov chain.

Using R, we obtain:

```
steadyStates(mc)
```

```
1 2 3 4 5
1 0 0 0 0
```

(e) Here we plot the unconditional probabilities at time n against the time.

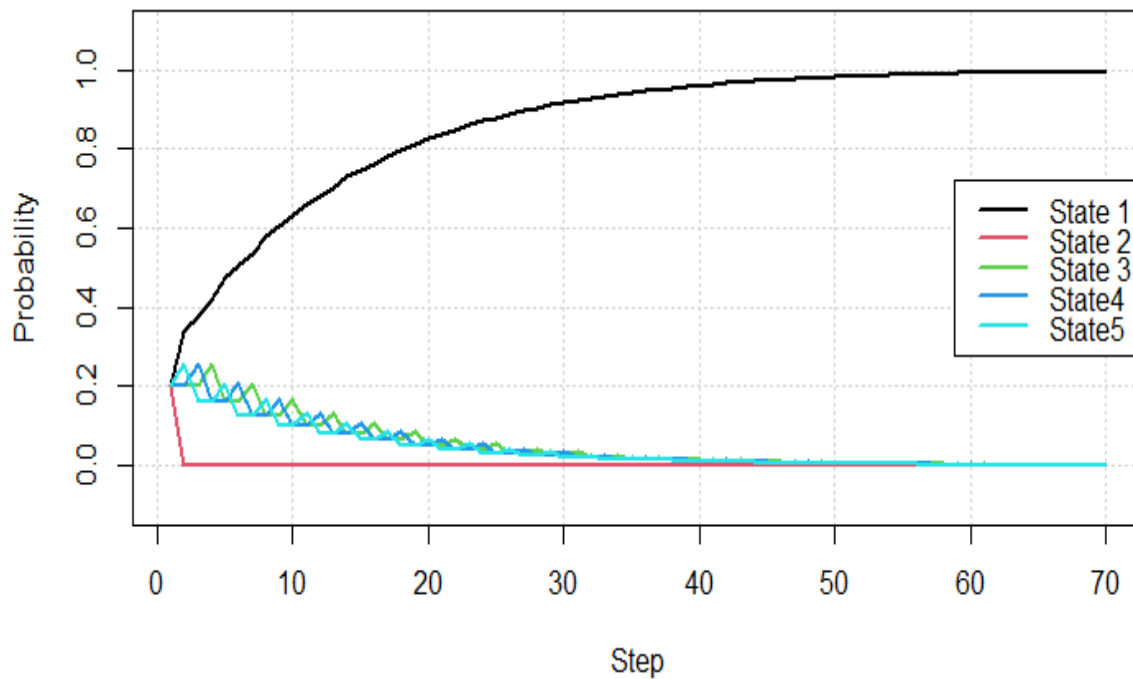
```
#specifying total number of steps
nsteps<- 70

#specifying matrix containing probabilities
probs<- matrix(NA, nrow=nsteps, ncol=5)

#computing probabilities
probs[1,] <- p0
for(n in 2:nsteps)
  probs[n,]<- probs[n-1,]%*%tm

#plotting probabilities vs. step by state
matplot(probs, type="l", lty=1, lwd=2, col=1:5, ylim=c(-0.1, 1.1),
xlab="Step", ylab="Probability", panel.first=grid())

legend("right", c("State 1", "State 2", "State 3", "State4", "State5"), lty=1,
lwd=2, col=1:5)
```



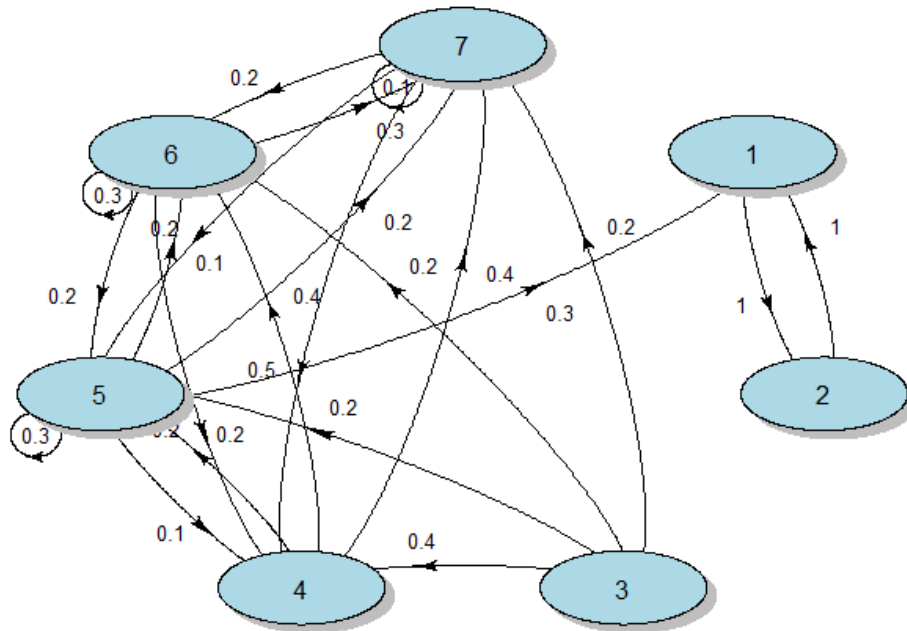
The convergence to the steady-state distribution is apparent after 60 steps.

EXERCISE 1.3. (a) We plot a diagram of the Markov chain.

```
#specifying transition probability matrix
tm<- matrix(c(0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0.4,0.2,0.2,0.2,
0,0,0,0,0.2,0.4,0.4,0.3,0,0,0.1,0.3,0.1,0.2,0,0,0,0.2,0.2,0.3,0.3,
0,0,0,0.5,0.2,0.2,0.1),nrow=7, ncol=7, byrow=TRUE)

#transposing transition probability matrix
tm.tr<- t(tm)

#plotting diagram
library(diagram)
plotmat(tm.tr, arr.length=0.3, arr.width=0.1, arr.pos=0.58, box.col="light blue",
box.lwd=1, box.prop=0.5, box.size=0.09, box.type="circle", cex.txt=0.8, lwd=1,
self.cex=0.3, self.shiftx=-0.07, self.shifty=-0.05)
```



(b) States 1 and 2 form a class and it is recurrent. The period is 2. Once the chain transitions into this class, it never leaves it and will bounce between the two states.

State 3 is reflecting. The chain leaves this state in one step. This state forms a class of its own. It is a transient class and its period is infinite.

States 4, 5, 6, and 7 communicate and thus form a class. Its period is one because of the loops. This class is transient because with positive probability the chain can leave this state and transition into the $\{1, 2\}$ class.

From R, we obtain:

```
#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("1", "2", "3", "4", "5",
"6", "7"))

#computing Markov chain characteristics
recurrentClasses(mc)

"1" "2"

transientClasses(mc)

"3"

"4" "5" "6" "7"

absorbingStates(mc)
character(0)

#creating irreducible Markov chain objects
tm.ir<- matrix(c(0,1,1,0),nrow=2, ncol=2, byrow=TRUE)
mc.ir<-new("markovchain", transitionMatrix=tm.ir, states=c("1","2"))
```



```
#finding periods of irreducible Markov chains
period(mc.ir)
```

2

(c) Below we simulate two trajectories of the chain that start at a randomly selected state.

```
#specifying total number of steps
nsteps<- 25

#specifying seed
set.seed(3339964)

#specifying initial probability
p0<- c(1/7, 1/7, 1/7, 1/7, 1/7, 1/7, 1/7)

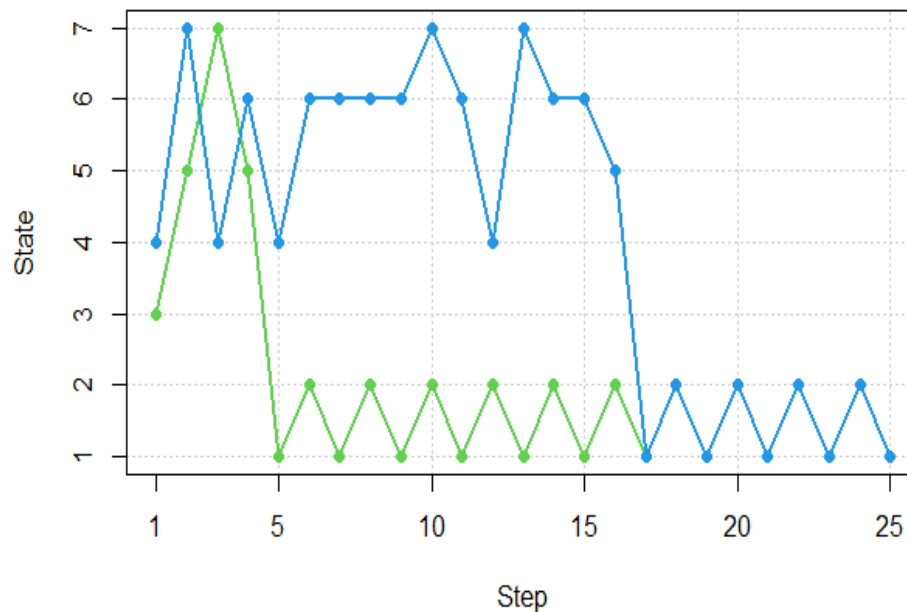
#specifying matrix containing states
MC.states<- matrix(NA, nrow=nsteps, ncol=2)

#simulating states
for (i in 1:2){
  state0<- sample(1:7, 1, prob=p0)
  MC.states[,i]<- rmarkovchain(n=nsteps-1, object=mc, t0=state0,
    include.t0=TRUE)
}

#plotting simulated trajectories
matplot(MC.states, type="l", lty=1, lwd=2, col=3:4, ylim=c(1,7), xaxt="n",
  xlab="Step", ylab="State", panel.first=grid())

axis(side=1, at=c(1,5,10,15,20,25))

points(1:nsteps, MC.states[,1], pch=16, col=3)
points(1:nsteps, MC.states[,2], pch=16, col=4)
```



Both simulated trajectories transition to the class $\{1, 2\}$ sooner or later.

(d) Below we calculate the limiting probabilities.

In R:

```
#finding steady-state distribution
round(steadyStates(mc), digits=4)
```

```
      1  2 3 4 5 6 7
0.5 0.5 0 0 0 0 0
```

There is a single limiting distribution which means that the chain is ergodic. States 1 and 2 absorb the chain and then the chain spends 50% of the time in state 1 and the other 50%, in state 2.

(e) Here we plot the unconditional probability vectors p_n against n .

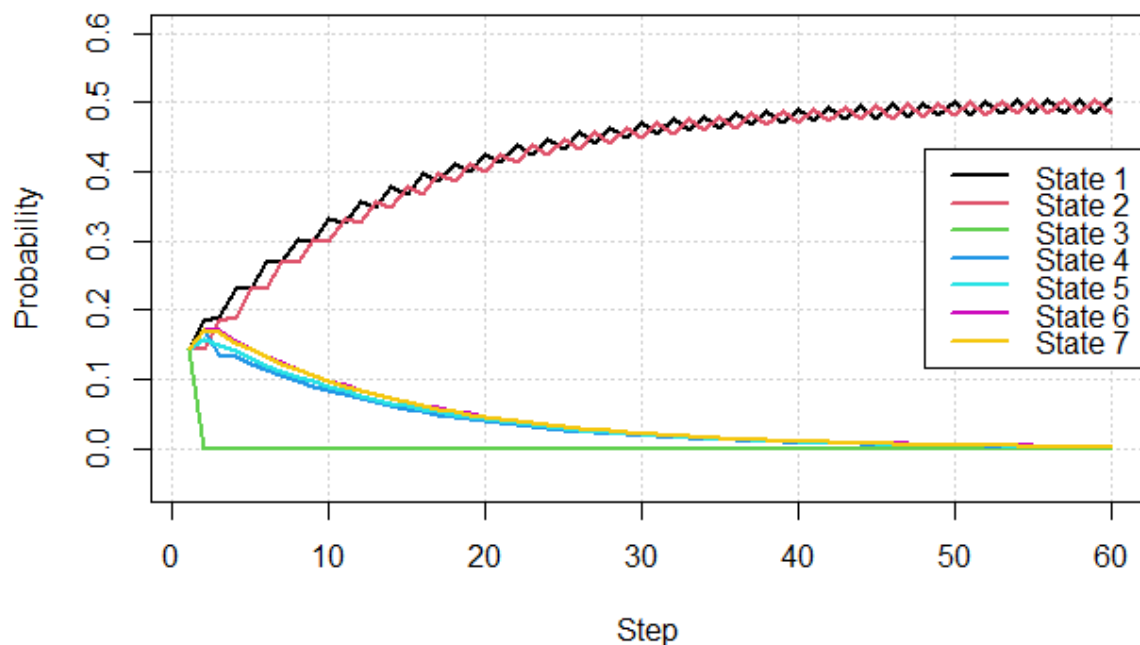
```
#specifying total number of steps
nsteps<- 60
```

```
#specifying matrix containing probabilities
probs<- matrix(NA, nrow=nsteps, ncol=7)
```

```
#computing probabilities
probs[1,] <- p0
for(n in 2:nsteps)
  probs[n,]<- probs[n-1,]%*%tm
```

```
#plotting probabilities vs. step by state
matplot(probs, type="l", lty=1, lwd=2, col=1:7, ylim=c(-0.05, 0.6),
xlab="Step", ylab="Probability", panel.first=grid())
```

```
legend("right", c("State 1","State 2","State 3","State 4","State 5","State 6",
"State 7"), lty=1, lwd=2, col=1:7)state 2","state 3","state 4","state 5","state
6", "state 7"), lty=1, col=1:7)
```



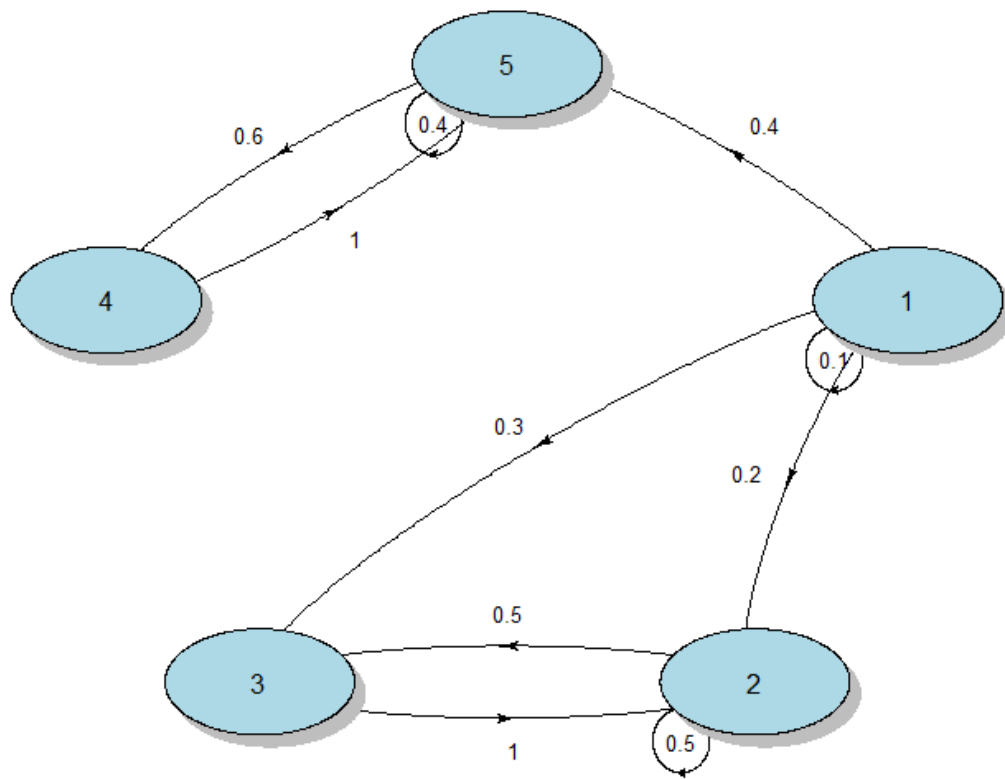
For state 1 and 2 the probabilities converge to 0.5, whereas for all the other states, the probabilities converge to zero. The curves settle around step 50.

EXERCISE 1.4. (a) We plot the diagram of the Markov chain.

```
#specifying the transition probability matrix
tm<- matrix(c(0.1,0.2,0.3,0,0.4,0,0.5,0.5,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0.6,0.4),
nrow=5, ncol=5, byrow=TRUE)

#transposing the transition probability matrix
tm.tr<- t(tm)

#plotting the diagram for the Markov chain
library(diagram)
plotmat(tm.tr, arr.length=0.3, arr.width=0.1, box.col="light blue", box.lwd=1,
box.prop=0.5, box.size=0.09, box.type="circle", cex.txt=0.8, lwd=1, self.cex=0.3,
self.shiftx=-0.07, self.shifty=-0.05)
```



(b) State 1 is a reflecting state because the chain leaves it with a positive probability. It might stay for a while in state 1 because of the loop, but then it leaves either to enter the irreducible class $\{2, 3\}$ or $\{4, 5\}$. These two classes are recurrent, with period 1 (because of the loops). There are no absorbing states.

In R:

```
#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("1", "2", "3", "4", "5"))

#computing Markov chain characteristics
recurrentClasses(mc)
```

```
"2" "3"
"4" "5"
```

```
transientClasses(mc)
```

```
"1"
```

```
absorbingStates(mc)
```

```
character(0)
```

```
#creating irreducible Markov chain objects
tm.ir1<- matrix(c(0,1,0.6,0.4),nrow=2, ncol=2, byrow=TRUE)
mc.ir1<-new("markovchain", transitionMatrix=tm.ir, states=c("4","5"))
```

```
#finding periods of irreducible Markov chains
period(mc.ir1)
```

```
1
```

```
#creating irreducible Markov chain objects
tm.ir2<- matrix(c(0.5, 0.5, 1, 0),nrow=2, ncol=2, byrow=TRUE)
mc.ir2<-new("markovchain", transitionMatrix=tm.ir, states=c("2","3"))
```

```
#finding periods of irreducible Markov chains
period(mc.ir2)
```

```
1
```

(c) We simulate two trajectories of the Markov chain.

```
#specifying total number of steps
nsteps<- 25
```

```
#specifying seed
set.seed(202870)
```

```
#specifying matrix containing states
MC.states<- matrix(NA, nrow=nsteps, ncol=2)
```

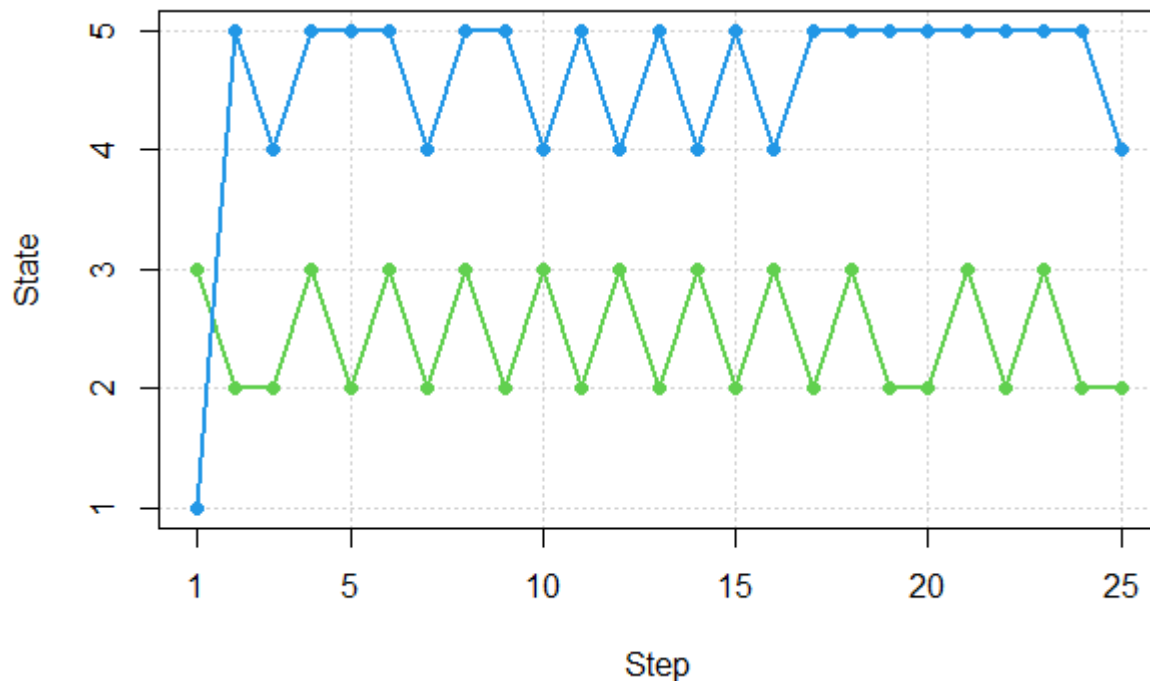
```
#simulating states
for (i in 1:2){
  state0<- sample(1:5, 1, prob=c(1/5, 1/5, 1/5, 1/5, 1/5))
  MC.states[,i]<- rmarkovchain(n=nsteps-1, object=mc, t0=state0,
    include.t0=TRUE)
}
```

```
#plotting simulated trajectories
matplot(MC.states, type="l", lty=1, lwd=2, col=3:4, ylim=c(1,5), xaxt="n",
  xlab="Step", ylab="State", panel.first=grid())
```

```
axis(side=1, at=c(1,5,10,15,20,25))
```

```
points(1:nsteps, MC.states[,1], pch=16, col=3)
points(1:nsteps, MC.states[,2], pch=16, col=4)
```

The trajectories enter either class {2, 3} or {4, 5} and keep bouncing between the two states within each class, possibly remaining for a little bit in state 2 or state 5 because of the loops.



(d) In R, we compute the invariant probability measures.

```
round(steadyStates(mc), digits=4)
```

	1	2	3	4	5
0	0.0000	0.0000	0.375	0.625	
0	0.6667	0.3333	0.000	0.000	

There are two invariant probability measures: $(0, 0, 0, 0.375, 0.625)$ and $(0, 0.6667, 0.3333, 0, 0)$. The chain will settle for one of these distributions, depending on what recurrent class it happens to enter $\{2,3\}$ or $\{4,5\}$. Neither of these two invariant measures is considered to be the stationary distribution because the chain is non-ergodic and the limiting distribution would depend on the initial state of the chain.

(e) We plot the graphs of unconditional probabilities against time, assuming successively that the chain starts in states 1, 2, 3, 4, and 5.

We run the following R code five times, each time changing the initial state.

```
#specifying total number of steps
nsteps<- 20

#specifying matrix containing probabilities
probs<- matrix(NA, nrow=nsteps, ncol=5)

#computing probabilities (initial state 1)
probs[1,] <- c(1,0,0,0,0)
#(state 2) c(0,1,0,0,0) (state 3) c(0,0,1,0,0) (state 4) c(0,0,0,1,0)
#(state 5) c(0,0,0,0,1)
for(n in 2:nsteps)
  probs[n,]<- probs[n-1,]%*%tm

#plotting probabilities vs. step by state
matplot(probs, main="Initial State 1", type="l", lty=1, lwd=2, col=1:5,
```

```
ylim=c(-0.05, 1.1), xlab="Step", ylab="Probability", panel.first=grid())

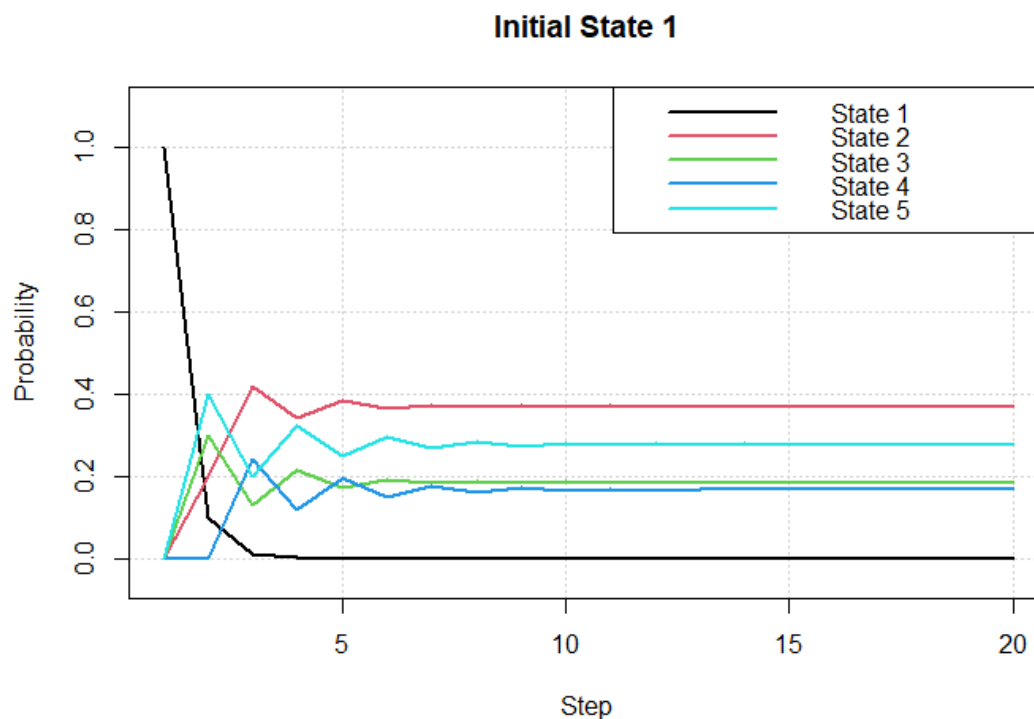
legend("topright", c("State 1", "State 2", "State 3", "State 4", "State 5"),
lty=1, lwd=2, col=1:5)
```

We obtain the following five graphs.

For the initial state 1:

```
> probs

      [,1]      [,2]      [,3]      [,4]      [,5]
[20,] 1e-19 0.3703699 0.1851856 0.1666536 0.2777908
```



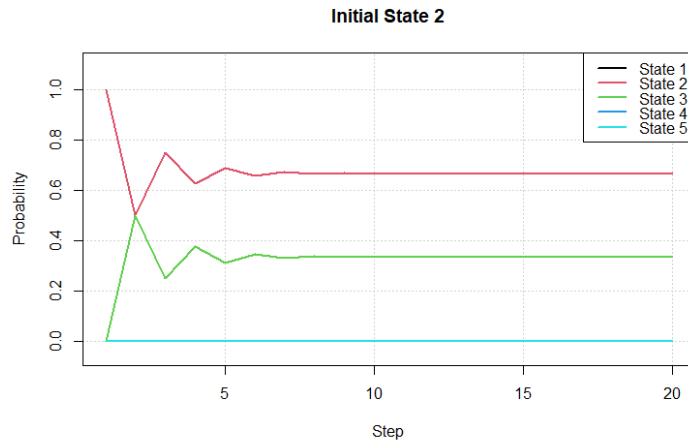
The probability of remaining in state 1 after n steps is $1 - 0.1^n$, so, as n increases, the probability converges to zero. For the other states, the probabilities converge to a linear combination of the two invariant vectors: the chain enters the class $\{2, 3\}$ with probability $(0.5)(1 + 0.1 + 0.1^2 + \dots) = \frac{0.5}{1-0.1} = \frac{5}{9}$ and class $\{4, 5\}$ with probability $1 - \frac{5}{9} = \frac{4}{9}$, thus the limiting probabilities are $\left(\frac{5}{9}\right)(0, 0.6667, 0.3333, 0, 0) + \left(\frac{4}{9}\right)(0, 0, 0, 0.375, 0.625) = (0, 0.3704, 0.1852, 0.1667, 0.2778)$.

For the initial state 2:

```
> probs
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[20,]	0	0.6666660	0.3333340	0	0

If the chain starts in state 2, it will remain within the class $\{2, 3\}$, and the respective probabilities will converge to the invariant vector $(0, 0.6667, 0.3333, 0, 0)$.

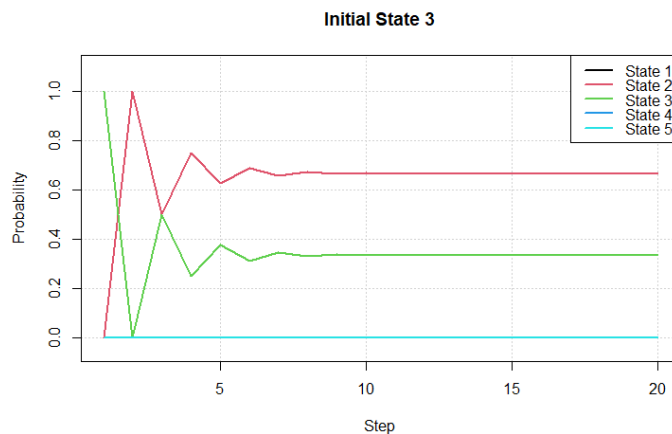


For the initial state 3:

```
> probs
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[20,]	0	0.6666679	0.3333321	0	0

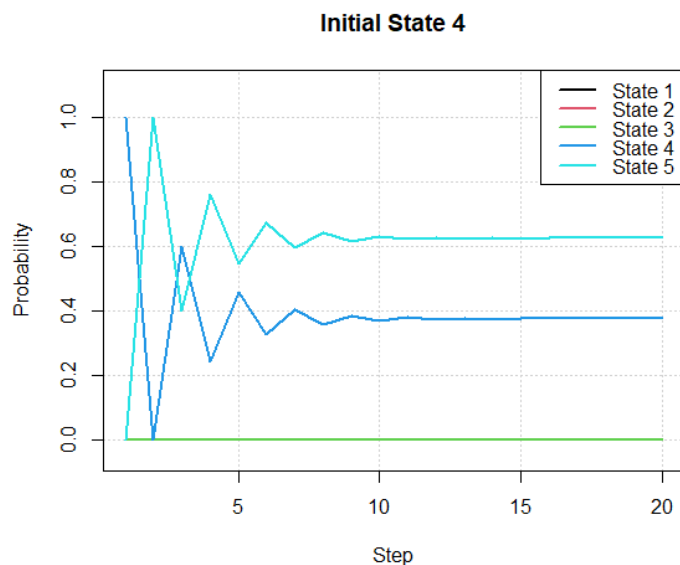
If the chain starts in state 3, it first goes to state 2 with probability one, and then will transition within the class $\{2, 3\}$, and the respective probabilities will converge to the invariant vector $(0, 0.6667, 0.3333, 0, 0)$.



For the initial state 4:

```
> probs
```

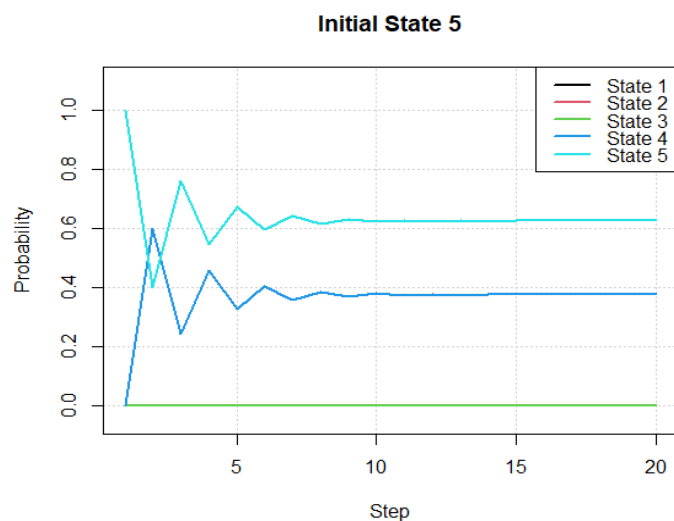
	[,1]	[,2]	[,3]	[,4]	[,5]
[20,]	0	0	0	0.3749619	0.6250381



If the chain starts in state 4, it first transitions into state 5 with probability one, and then will stay within the class $\{4, 5\}$, and the respective probabilities will converge to the invariant vector $(0, 0, 0, 0.375, 0.625)$.

For the initial state 5:

```
> probs
      [,1] [,2] [,3] [,4] [,5]
[20,]  0    0    0 0.3750229 0.6249771
```



If the chain starts in state 5, it transitions within the class $\{4, 5\}$, and the respective probabilities will converge to the invariant vector $(0, 0, 0, 0.375, 0.625)$.

EXERCISE 1.5. (a) In a box, there are two red (R), four blue (B), and eight green (G) balls. One ball is drawn at a time without replacement and its color is noted. The stochastic process $\{X_n, n = 1, 2, \dots\}$ with the state space $S = \{R, B, G\}$ doesn't satisfy the Markovian property. It can be proved, for example, as $P(X_3 = G | X_1 = R, X_2 = B) = \frac{8}{12}$, whereas $P(X_3 = G | X_1 = G, X_2 = B) = \frac{7}{12}$, thus, the color of the ball drawn at the third step depends on the colors of all previously drawn balls, not just the one drawn at step two.

(b) If the drawing is done with replacement, the process is a Markov chain. Since the balls are put back into the box, the colors of drawn balls are independent of each other. Let C stand for any of the three colors: red, blue, or green. Then we can write

$$P(X_3 = C | X_1 = C, X_2 = C) = P(X_3 = C) = P(X_3 = C | X_2 = C),$$

and thus, the Markov property always holds. Note that a sequence of independent trials is a special case of a Markov chain.

EXERCISE 1.6. Let O denote any outcome of a coin flip. The flips are considered independent, therefore, we obtain

$$P(X_3 = O | X_1 = O, X_2 = O) = P(X_3 = O) = P(X_3 = O | X_2 = O),$$

that is, the Markovian property always holds. The coin is fair, hence, the transition probability matrix is

$$\begin{array}{c} H \\ T \end{array} \begin{array}{cc} H & T \\ \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \end{array}. \text{ To derive the limiting probabilities, we solve } (\pi_H, \pi_T) = (\pi_H, \pi_T) \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

where $\pi_H + \pi_T = 1$. The solution is $\pi_H = \pi_T = 0.5$. Note that a sequence of independent trials is a special case of a Markov chain.

EXERCISE 1.7. (a) Below we find the transition probability matrix for Chapter 1 of “Moby Dick” by Herman Melville.

```
library(tidyverse)

chapter1 <- read_file("./Loomings.txt")

#cleaning the text
lowercase<- tolower(chapter1)
no.blanks<- gsub(" ", "", lowercase)
no.line.breaks<- gsub("\r\n", "", no.blanks)
#removing all punctuation
clean.string<- gsub("[[:punct:]]", "", no.line.breaks)

#splitting the string into characters
x2<- strsplit(clean.string, "")

#shifting the text by one place
no.last<- substr(clean.string, 1, nchar(clean.string)-1)
first.blank<- str_c(" ", no.last)
x1<- strsplit(first.blank, "")

vowels<-c("a","e","i","o","u")
consonants<- c("b","c","d","f","g","h","j","k","l","m","n","p","q","r","s","t",
"v","w","x","y","z")

for (counter in 1:nchar(x2)){
  v<- ifelse(x2[[counter]] %in% vowels,1,0)
  c<- ifelse(x2[[counter]] %in% consonants,1,0)
```

```

vv<- ifelse(x1[[counter]] %in% vowels & x2[[counter]] %in% vowels,1,0)
vc<- ifelse(x1[[counter]] %in% vowels & x2[[counter]] %in% consonants,1,0)
cv<- ifelse(x1[[counter]] %in% consonants & x2[[counter]] %in% vowels,1,0)
cc<- ifelse(x1[[counter]] %in% consonants & x2[[counter]] %in% consonants,1,0)
}

```

```
sum(v)
```

3647

```
sum(c)
```

5871

```
sum(vv)
```

572

```
sum(vc)
```

3075

```
sum(cv)
```

3075

```
sum(cc)
```

2795

We check quickly that these numbers add up properly. Since the first chapter of “Moby Dick” starts and ends with consonants, all vowels are transitioned into and transitioned from. Thus, $\text{sum}(v) = 3647 = \text{sum}(vv) + \text{sum}(vc) = 572 + 3075 = \text{sum}(vv) + \text{sum}(cv)$. Also, all but the last consonant are transitioned from, therefore, $\text{sum}(c) - 1 = 5870 = \text{sum}(cv) + \text{sum}(cc) = 3075 + 2795$, and all but the first consonant are transitioned to, so $\text{sum}(c) - 1 = 5870 = \text{sum}(vc) + \text{sum}(cc) = 3075 + 2795$. The transition probability matrix is

$$\begin{array}{cc}
 & \begin{array}{c} v \\ c \end{array} & \begin{array}{c} v \\ c \end{array} \\
 \begin{array}{c} v \\ c \end{array} & \begin{bmatrix} \frac{572}{3647} = 0.15684 & \frac{3075}{3647} = 0.84316 \\ \frac{3075}{5870} = 0.52385 & \frac{2795}{5870} = 0.47615 \end{bmatrix}
 \end{array}$$

The code below computes the limiting probabilities and the proportions of vowels and consonants in the text.

```

#specifying the transition probability matrix
tm<- matrix(c(sum(vv)/sum(v), sum(vc)/sum(v), sum(cv)/(sum(c)-1),
sum(cc)/(sum(c)-1)), nrow=2, ncol=2, byrow=TRUE)

#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("v","c"))

#computing limiting probabilities
steadyStates(mc)

```

0.383209^v 0.616791^c

```
#computing proportions of vowels and consonants
print(prop.vowels<- sum(v)/(sum(v)+sum(c)))
```

0.3831687

```
print(prop.cons<- sum(c)/(sum(v)+sum(c)))
```

0.6168313

From the output, the limiting probabilities are equal to the actual proportions of vowels and consonants.

(b) Now, we run the same code as in part (a), but with the text of Chapter 2. The code and output are

```
library(tidyverse)
chapter2 <- read_file("./The Carpet-Bag.txt")

#cleaning the text
lowercase<- tolower(chapter2)
no.blanks<- gsub(" ","",lowercase)
no.line.breaks<- gsub("\r\n", "", no.blanks)
#removing all punctuation
clean.string<- gsub("[[:punct:]]","",no.line.breaks)

#splitting the string into characters
x2<- strsplit(clean.string, "")

#shifting the text by one place
no.last<- substr(clean.string, 1, nchar(clean.string)-1)
first.blank<- str_c(" ", no.last)
x1<- strsplit(first.blank,"")

vowels<-c("a","e","i","o","u")
consonants<- c("b","c","d","f","g","h","j","k","l","m","n","p","q","r","s","t",
"v","w","x","y","z")

for (counter in 1:nchar(x2)){
  v<- ifelse(x2[[counter]] %in% vowels,1,0)
  c<- ifelse(x2[[counter]] %in% consonants,1,0)
  vv<- ifelse(x1[[counter]] %in% vowels & x2[[counter]] %in% vowels,1,0)
  vc<- ifelse(x1[[counter]] %in% vowels & x2[[counter]] %in% consonants,1,0)
  cv<- ifelse(x1[[counter]] %in% consonants & x2[[counter]] %in% vowels,1,0)
  cc<- ifelse(x1[[counter]] %in% consonants & x2[[counter]] %in% consonants,1,0)
}

sum(v)

2319

sum(c)

3899

sum(vv)
```

340

$\text{sum}(vc)$

1978

$\text{sum}(cv)$

1978

$\text{sum}(cc)$

1921

Chapter 2 starts and ends with vowels, all consonants are transitions into and transitioned from. Therefore, we must have $\text{sum}(c) = 3899 = \text{sum}(vc) + \text{sum}(cc) = 1978 + 1921 = \text{sum}(cv) + \text{sum}(cc)$. Also, all but the last vowel are transitioned from, so $\text{sum}(v) - 1 = 2318 = \text{sum}(vv) + \text{sum}(vc) = 340 + 1978$, and all but the first vowel are transitioned into, so $\text{sum}(v) - 1 = 2318 = \text{sum}(vv) + \text{sum}(cv) = 340 + 1978$. The transition probability matrix is

$$\begin{array}{c} \begin{array}{cc} & v & c \\ v & \left[\begin{array}{cc} \frac{340}{2318} = 0.14668 & \frac{1978}{2318} = 0.85332 \\ \frac{1978}{3899} = 0.50731 & \frac{1921}{3899} = 0.49269 \end{array} \right] & \\ c & & \end{array} \end{array}.$$

These transition probabilities are not exactly equal to the ones in Chapter 1 but are within 1/100th, which is very close.

EXERCISE 1.8. The sentence “The quick brown fox jumped over the lazy dog” repeated 500 times is a deterministic sequence of vowels and consonants and cannot be modeled as a Markov chain. To prove this mathematically, we do the following calculations.

thequickbrownfoxjumpedoverthelazydog|thequick...
ccvcvccccvccccvccvccvccvccvccvccvcc|ccvcvcc...

Let $k = 500$ denote the number of repetitions. There are 0 vvv subsequences, k vvc subsequences, k cvv subsequences, and $10k$ cvc subsequences. Thus, we compute

$$P(X_3 = v | X_2 = v, X_1 = v) = \frac{P(vvv)}{P(vvv) + P(vvc)} = \frac{0}{0+k} = 0, \text{ whereas } P(X_3 = v | X_2 = v) = \frac{P(vvv) + P(cvv)}{P(vvv) + P(cvv) + P(vvc) + P(cvc)} = \frac{0+k}{0+k+k+10k} = \frac{1}{12} \neq 0, \text{ and so, the Markovian property doesn't hold.}$$

EXERCISE 1.9. (a) We show that the genotypes of the direct descendant and the second parent follow a Markov chain with the state space $S = \{(AA, AA), (AA, Aa), (AA, aa), (Aa, AA), (Aa, Aa), (Aa, aa), (aa, AA), (aa, Aa), (aa, aa)\}$ and the transition probability matrix

	(AA, AA)	(AA, Aa)	(AA, aa)	(Aa, AA)	(Aa, Aa)	(Aa, aa)	(aa, AA)	(aa, Aa)	(aa, aa)
(AA, AA)	1/3	1/3	1/3	0	0	0	0	0	0
(AA, Aa)	1/6	1/6	1/6	1/6	1/6	1/6	0	0	0
(AA, aa)	0	0	0	1/3	1/3	1/3	0	0	0
(Aa, AA)	1/6	1/6	1/6	1/6	1/6	1/6	0	0	0
(Aa, Aa)	1/12	1/12	1/12	1/6	1/6	1/6	1/12	1/12	1/12
(Aa, aa)	0	0	0	1/6	1/6	1/6	1/6	1/6	1/6
(aa, AA)	0	0	0	1/3	1/3	1/3	0	0	0
(aa, Aa)	0	0	0	1/6	1/6	1/6	1/6	1/6	1/6
(aa, aa)	0	0	0	0	0	0	1/3	1/3	1/3

- If both parents have the combination AA of genes, their direct descendant will have genes AA with probability one, and will equally likely choose the second parent with genes AA, Aa, or aa. Therefore, the state (AA, AA) transitions into states (AA, AA), (AA, Aa), and (AA, aa) with probabilities equal to 1/3.
- If the parents have genes (AA, Aa), their direct descendant will have genes AA with probability 1/2 or genes Aa with probability 1/2. And will choose the second parent with either of the three types with probability 1/3. Thus, (AA, Aa) transitions into states (AA, AA), (AA, Aa), (AA, aa), (Aa, AA), (Aa, Aa), or (Aa, aa), with probability $\left(\frac{1}{2}\right)\left(\frac{1}{3}\right) = \frac{1}{6}$ each.
- If the parents have genes (Aa, Aa), their offspring will have genetic type AA with probability 1/4, Aa with probability 1/2, and aa with probability 1/4. Combining with a randomly chosen second parent's gene type, we see that the state (Aa, Aa) transitions into states (AA, AA), (AA, Aa), (AA, aa), (aa, AA), (aa, Aa), or (aa, aa) with probability $\left(\frac{1}{4}\right)\left(\frac{1}{3}\right) = \frac{1}{12}$ each, and states (Aa, AA), (Aa, Aa), or (Aa, aa) with probability $\left(\frac{1}{2}\right)\left(\frac{1}{3}\right) = \frac{1}{6}$ each.

The other cases are proven similarly.

(b) Below we determine the transient and recurrent classes of the Markov chain.

```
#specifying transition probability matrix
tm<- matrix(c(1/3, 1/3, 1/3, 0, 0, 0, 0, 0, 0, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 0, 0,
0, 0, 0, 1/3, 1/3, 1/3, 0, 0, 0, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 0, 0, 0,
1/12, 1/12, 1/12, 1/6, 1/6, 1/6, 1/12, 1/12, 1/12, 0, 0, 0, 1/6, 1/6, 1/6, 1/6,
1/6, 1/6, 0, 0, 0, 1/3, 1/3, 1/3, 0, 0, 0, 0, 0, 1/6, 1/6, 1/6, 1/6, 1/6,
0, 0, 0, 0, 0, 0, 1/3, 1/3, 1/3), nrow=9, ncol=9, byrow=TRUE)
```

```
#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("AAAA", "AAaA", "AAaa",
"AaAA", "AaAa", "Aaaa", "aaAA", "aaAa", "aaaa"))
```

```
#computing Markov chain characteristics
transientStates(mc)
```

```
character(0)
```

```
recurrentClasses(mc)
```

```
"AAAA" "AAaA" "AAaa" "AaAA" "AaAa" "Aaaa" "aaAA" "aaAa" "aaaa"
```

All states are recurrent and none are transient.

(c) Next, we find the stationary distribution.

```
#finding stationary distribution
steadyStates(mc)
```

```
      AAAA      AAAa      AAaa      AaAA
0.08333333 0.08333333 0.08333333 0.16666667
      AaAa      Aaaa      aaAA      aaAa
0.16666667 0.16666667 0.08333333 0.08333333
      aaaa
0.08333333
```

In a long run, there will be about 8.3% of each gene type (AA, AA) , (AA, Aa) , (AA, aa) , (aa, AA) , (aa, Aa) , and (aa, aa) , and about 16.7% of each gene type (Aa, AA) , (Aa, Aa) , and (Aa, aa) .

(d) Here we find the initial state that achieves the stationary distribution in the smallest number of generations. To answer this question, we run the following R code nine times with the initial state ranging from $(1, 0, 0, \dots, 0)$ to $(0, 0, \dots, 0, 1)$. We look for the smallest generation number for which the steady-state distribution has probabilities that round to 0.0833 and 0.1667.

```
#specifying total number of steps
nsteps<- 20

#specifying matrix containing probabilities
probs<- matrix(NA, nrow=nsteps, ncol=9)

#computing probabilities
probs[1,] <- c(1, 0, 0, 0, 0, 0, 0, 0, 0)

for(n in 2:nsteps) {
  print(n)
  print (probs[n,]<- round(probs[n-1,]%*%tm, 6))
}
```

```
15
[1,] 0.083353 0.083353 0.083353 0.166665 0.166665 0.166665 0.083312 0.083312 0.083312
```

```
16
[1,] 0.083343 0.083343 0.083343 0.166665 0.166665 0.166665 0.083322 0.083322 0.083322
```

The initial state (AA, AA) achieves the steady-state probabilities in the 16th generation. We repeat this code for the other 8 initial states and get the following result:

Initial State	(AA, AA)	(AA, Aa)	(AA, aa)	(Aa, AA)	(Aa, Aa)	(Aa, aa)	(aa, AA)	(aa, Aa)	(aa, aa)
Smallest Generation To Achieve the Steady-State Distribution	16	15	3	15	2	15	3	15	16

The conclusion is that the most genetically diversified type (Aa, Aa) in the first generation results in the limiting distribution already in the second generation, whereas the least diversified genetic types (AA, AA) and (aa, aa) need to wait until the 16th generation to see convergence.

EXERCISE 1.10. The code below selects the weather conditions for Detroit, and computes the empirical conditional probability of clouds tomorrow, given clouds today and yesterday, and the conditional probability of clouds tomorrow, given clouds today.

```
weather.data<- read.csv("./weather_description.csv", header=TRUE, sep=",")
DET<- weather.data$Detroit
```

```
table(DET)
```

```
DET
      broken clouds      drizzle      few clouds
      3975              96      1775
      fog              freezing rain      haze
      585                3      768
heavy intensity drizzle heavy intensity rain heavy shower snow
      8              437      107
      heavy snow light intensity drizzle light intensity drizzle rain
      249              396      2
light intensity shower rain light rain light shower sleet
      193              3873      1
      light shower snow light snow mist
      172              1383      3414
      moderate rain overcast clouds proximity shower rain
      1450              6470      208
proximity thunderstorm scattered clouds shower rain
      104              3940      7
      shower snow sky is clear smoke
      4              15249      1
      snow squalls thunderstorm
      222              1      66
thunderstorm with heavy rain thunderstorm with light rain thunderstorm with rain
      14              41      34
      very heavy rain
      4
```

```
X3<- ifelse(DET=="sky is clear", "no clouds", ifelse(DET %in% c("broken clouds",
"few clouds", "overcast clouds", "scattered clouds", "smoke"), "clouds",
ifelse(DET %in% c("heavy shower snow", "heavy snow", "light shower snow", "light
snow", "shower snow", "snow"), "snow", "rain")))
```

```
table(X3)
```

```
X3
clouds no clouds skyclear snow
16161 11705 15249 2137
```

```
library(Hmisc)
```

```
X2<- Lag(X3,shift=1)
```

```
X1<- Lag(X3,shift=2)
```

```
library(Hmisc)
```

```
X2<- Lag(X3,shift=1)
```

```
X1<- Lag(X3,shift=2)
```

```
#computing P(X3=c|X2=c,X1=c)
```

```
ccc<- ifelse(X1=="clouds" & X2=="clouds" & X3=="clouds",1,0)
```

```
ccn<- ifelse(X1=="clouds" & X2=="clouds" & X3=="no clouds",1,0)
```

```
ccr<- ifelse(X1=="clouds" & X2=="clouds" & X3=="rain",1,0)
```

```
ccs<- ifelse(X1=="clouds" & X2=="clouds" & X3=="snow",1,0)
```

```
sum(ccc)/sum(ccc+ccn+ccr+ccs)
```

0.8143572

```
#computing P(X3=c|X2=c)
```

```
ncc<- ifelse(X1=="no clouds" & X2=="clouds" & X3=="clouds",1,0)
```

```
rcc<- ifelse(X1=="rain" & X2=="clouds" & X3=="clouds",1,0)
```

```

scc<- ifelse(X1=="snow" & X2=="clouds" & X3=="clouds",1,0)
ncn<- ifelse(X1=="no clouds" & X2=="clouds" & X3=="no clouds",1,0)
ncr<- ifelse(X1=="no clouds" & X2=="clouds" & X3=="rain",1,0)
ncs<- ifelse(X1=="no clouds" & X2=="clouds" & X3=="snow",1,0)
rcn<- ifelse(X1=="rain" & X2=="clouds" & X3=="no clouds",1,0)
rcr<- ifelse(X1=="rain" & X2=="clouds" & X3=="rain",1,0)
rcs<- ifelse(X1=="rain" & X2=="clouds" & X3=="snow",1,0)
scn<- ifelse(X1=="snow" & X2=="clouds" & X3=="no clouds",1,0)
scr<- ifelse(X1=="snow" & X2=="clouds" & X3=="rain",1,0)
scs<- ifelse(X1=="snow" & X2=="clouds" & X3=="snow",1,0)

sum(ccc+ncc+rcc+scc)/sum(ccc+ccn+ccr+ccs+ncc+ncn+ncr+ncs+rcc+rcn
+rcr+rcs+scc+scn+scr+scs)

```

0.7662892

The state space of this process is $S = \{c = \text{"clouds"}, n = \text{"no clouds"}, r = \text{"rain"}, s = \text{"snow"}\}$. From the above output,

$$\begin{aligned}
 P(X_3 = c \mid X_1 = c, X_2 = c) &= \frac{P(X_1 = c, X_2 = c, X_3 = c)}{P(X_1 = c, X_2 = c)} \\
 &= \frac{P(ccc)}{P(ccc) + P(ccn) + P(ccr) + P(ccs)} = 0.8143572,
 \end{aligned}$$

whereas

$$P(X_3 = c \mid X_2 = c) = \frac{P(ccc) + P(ncc) + P(rcc) + P(scc)}{P(ccc) + P(ccn) + P(ccr) + \dots + P(scs)} = 0.7662892.$$

Since the two quantities are not the same, the process is not a Markov chain.

EXERCISE 1.11. Let s denote the air quality status (good/unhealthy/hazardous). We are given that $P(X_{n+1} = s \mid X_n = s, X_{n-1} = s, \dots, X_1 = s) = P(X_{n+1} = s \mid X_n = s, X_{n-1} = s)$. We consider states as air quality statuses in two consecutive days: (X_1, X_2) , (X_2, X_3) , etc. We show that the Markov property holds:

$$\begin{aligned}
 &P((X_n, X_{n+1}) = (s, s) \mid (X_{n-1}, X_n) = (s, s), \dots, (X_1, X_2) = (s, s)) \\
 &= P(X_{n+1} = s, X_n = s \mid X_n = s, X_{n-1} = s, \dots, X_1 = s) \\
 &= \frac{P(X_{n+1} = s, X_n = s, X_{n-1} = s, \dots, X_1 = s)}{P(X_n = s, X_{n-1} = s, \dots, X_1 = s)} \\
 &= P(X_{n+1} = s \mid X_n = s, X_{n-1} = s, \dots, X_1 = s) \\
 &= P(X_{n+1} = s \mid X_n = s, X_{n-1} = s) \text{ (by the assumption of the problem)} \\
 &= \frac{P(X_{n+1} = s, X_n = s, X_{n-1} = s)}{P(X_n = s, X_{n-1} = s)} \\
 &= P((X_n, X_{n+1}) = (s, s) \mid (X_{n-1}, X_n) = (s, s)).
 \end{aligned}$$

EXERCISE 1.12. (a) Let state 1 give income \$200, state 2 give income \$0, state 3 give income -\$75, state 4 give income \$105, and state 5 give income -\$130. Since the rolls of the die are independent, the next state will depend only on the present state. We use the fact that the die is fair and that the states are traversed circularly, and write the transition probability matrix:

	1	2	3	4	5
1	1/6	1/3	1/6	1/6	1/6
2	1/6	1/6	1/3	1/6	1/6
3	1/6	1/6	1/6	1/3	1/6
4	1/6	1/6	1/6	1/6	1/3
5	1/3	1/6	1/6	1/6	1/6

(b) We compute the steady-state probability of each square and find the long-run winning of the player.

```
#specifying transition probability matrix
tm<- matrix(c(1/6, 1/3, 1/6, 1/6, 1/6,
              1/6, 1/6, 1/3, 1/6, 1/6,
              1/6, 1/6, 1/6, 1/3, 1/6,
              1/6, 1/6, 1/6, 1/6, 1/3,
              1/3, 1/6, 1/6, 1/6, 1/6),
            nrow=5, ncol=5, byrow=TRUE)

#creating Markov chain object
library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("1", "2", "3", "4", "5"))

steadyStates(mc)

      1      2      3      4      5
0.2 0.2 0.2 0.2 0.2
```

In the long-run, the chain will be uniformly distributed between the five states, and thus the expected winning of the player will be $E(\text{winning}) = (\$200)(0.2) + (\$0)(0.2) + (-\$75)(0.2) + (\$105)(0.2) + (-\$130)(0.2) = \20 .

EXERCISE 1.13. (a) Assuming that the traffic starts with the light state at 1PM, we find the distribution of the states at 6PM. Traffic conditions change every 20 minutes, therefore between 1PM and 4PM there will be 9 transitions between the states (light/heavy/jammed), and between 4PM and 6PM there will be 6 transitions. We run the following R code to find the distribution of states at 6PM:

```
#specifying the transition probability matrices
tm1<- matrix(c(0.4, 0.4, 0.2, 0.3, 0.5, 0.2, 0, 0.5, 0.5), nrow=3, ncol=3,
              byrow=TRUE)
tm2<- matrix(c(0.1, 0.5, 0.4, 0.1, 0.3, 0.6, 0, 0.1, 0.9), nrow=3, ncol=3,
              byrow=TRUE)

#computing the unconditional distribution at 6pm
library(expm)
state1pm<- c(1, 0, 0)
state4pm<- state1pm%*%(tm1%^9)
print(state6pm<- state4pm%*%(tm2%^6))

      [,1]      [,2]      [,3]
0.01504877 0.1332313 0.85172
```

$P(\text{light traffic}) = 0.01504877$, $P(\text{heavy traffic}) = 0.1332313$, and $P(\text{jammed traffic}) = 0.85172$.

(b) Below we simulate 10,000 trajectories to verify the result of part (a).

```
#creating Markov chain objects
library(markovchain)
mc1<- new("markovchain", transitionMatrix=tm1, states=c("light", "heavy",
"jammed"))

mc2<- new("markovchain", transitionMatrix=tm2, states=c("light", "heavy",
"jammed"))

#simulating states between 1pm and 4pm
MC.states4pm<- matrix(NA, nrow=9, ncol=10000)

for (i in 1:10000)
MC.states4pm[,i]<- rmarkovchain(n=9, object=mc1, t0="light")

#simulating states between 4pm and 6pm
MC.states6pm<- matrix(NA, nrow=6, ncol=10000)

for (i in 1:10000)
MC.states6pm[,i]<- rmarkovchain(n=6, object=mc2, t0=MC.states4pm[9,i])

#concatenating two matrices
MC.states<- rbind(MC.states4pm, MC.states6pm)

#computing frequencies of states at 6pm
table(MC.states[15,])

heavy jammed light
1316 8533 151
```

Thus, the estimates are $\hat{P}(\text{light traffic}) = 0.0151$, $\hat{P}(\text{heavy traffic}) = 0.1316$, and $\hat{P}(\text{jammed traffic}) = 0.8533$.

EXERCISE 1.14. (a) Assuming that a shrub is initially sustainable, we simulate three trajectories of the Markov chain.

```
#specifying transition probability matrix
tm<- matrix(c(0.6, 0.2, 0.1, 0.1, 0.7, 0.2, 0.1, 0, 0.1, 0.3, 0.4, 0.2, 0,
0, 0, 1), nrow=4, ncol=4, byrow=TRUE)

library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("1", "2", "3", "4"))

#specifying total number of steps
nsteps<- 25

#specifying seed
set.seed(912332)

#specifying matrix containing states
MC.states<- matrix(NA, nrow=nsteps, ncol=3)
```

```

#simulating states

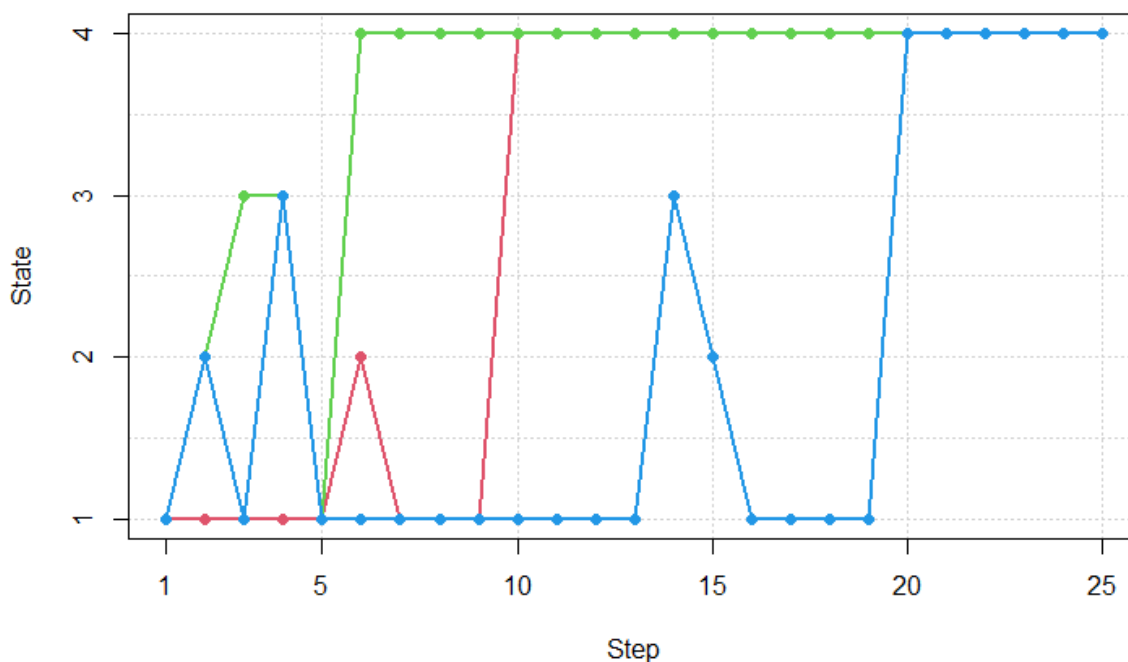
for (i in 1:3){
  state0<- 1
  MC.states[,i]<- rmarkovchain(n=nsteps-1, object=mc, t0=state0,
    include.t0=TRUE)
}

#plotting simulated trajectories
matplot(MC.states, type="l", lty=1, lwd=2, col=2:4, xaxt="n", yaxt="n",
  ylim=c(1,4), xlab="Step", ylab="State", panel.first=grid())

axis(side=1, at=c(1,5,10,15,20,25))
axis(side=2, at=c(1,2,3,4))

points(1:nsteps, MC.states[,1], pch=16, col=2)
points(1:nsteps, MC.states[,2], pch=16, col=3)
points(1:nsteps, MC.states[,3], pch=16, col=4)

```



(b) Here we find the probability that an initially sustainable shrub will eventually become extinct.

```
steadyStates(mc)
```

```

1 2 3 4
0 0 0 1

```

State 4 (“extinct”) is an absorbing state, thus, eventually, a sustainable shrub will become extinct with probability one.

EXERCISE 1.15. (a) The number of music instruments on Tuesday morning can assume values 3, 4, 5, 6, or 7. Therefore, the state space consists of these five states.

Let $N \sim \text{Poisson}(4)$ be the number of instruments bought during the week. Then

$$\begin{aligned}
P(3 \text{ instruments this week} \mid 3 \text{ instruments previous week}) &= P(N = 0) = \exp(-4) = 0.0183 \\
P(4 \text{ instruments this week} \mid 3 \text{ instruments previous week}) &= 0 \\
P(5 \text{ instruments this week} \mid 3 \text{ instruments previous week}) &= 0 \\
P(6 \text{ instruments this week} \mid 3 \text{ instruments previous week}) &= 0 \\
P(7 \text{ instruments this week} \mid 3 \text{ instruments previous week}) &= P(N \geq 1) = 1 - 0.0183 = 0.9817
\end{aligned}$$

$$\begin{aligned}
P(3 \text{ instruments this week} \mid 4 \text{ instruments previous week}) &= P(N = 1) = 4 \exp(-4) = 0.0733 \\
P(4 \text{ instruments this week} \mid 4 \text{ instruments previous week}) &= P(N = 0) = \exp(-4) = 0.0183 \\
P(5 \text{ instruments this week} \mid 4 \text{ instruments previous week}) &= 0 \\
P(6 \text{ instruments this week} \mid 4 \text{ instruments previous week}) &= 0 \\
P(7 \text{ instruments this week} \mid 4 \text{ instruments previous week}) &= P(N \geq 2) = 1 - 0.0733 - 0.0183 \\
&= 0.9084
\end{aligned}$$

$$\begin{aligned}
P(3 \text{ instruments this week} \mid 5 \text{ instruments previous week}) &= P(N = 2) = \frac{4^2}{2!} \exp(-4) = 0.1465 \\
P(4 \text{ instruments this week} \mid 5 \text{ instruments previous week}) &= P(N = 1) = 4 \exp(-4) = 0.0733 \\
P(5 \text{ instruments this week} \mid 5 \text{ instruments previous week}) &= P(N = 0) = \exp(-4) = 0.0183 \\
P(6 \text{ instruments this week} \mid 5 \text{ instruments previous week}) &= 0 \\
P(7 \text{ instruments this week} \mid 5 \text{ instruments previous week}) &= P(N \geq 3) = 1 - 0.1465 - 0.0733 \\
&- 0.0183 = 0.7619
\end{aligned}$$

$$\begin{aligned}
P(3 \text{ instruments this week} \mid 6 \text{ instruments previous week}) &= P(N = 3) = \frac{4^3}{3!} \exp(-4) = 0.1954 \\
P(4 \text{ instruments this week} \mid 6 \text{ instruments previous week}) &= P(N = 2) = \frac{4^2}{2!} \exp(-4) = 0.1465 \\
P(5 \text{ instruments this week} \mid 6 \text{ instruments previous week}) &= P(N = 1) = 4 \exp(-4) = 0.0733 \\
P(6 \text{ instruments this week} \mid 6 \text{ instruments previous week}) &= P(N = 0) = \exp(-4) = 0.0183 \\
P(7 \text{ instruments this week} \mid 6 \text{ instruments previous week}) &= P(N \geq 4) = 1 - 0.1954 - 0.1465 \\
&- 0.0733 - 0.0183 = 0.5665
\end{aligned}$$

$$\begin{aligned}
P(3 \text{ instruments this week} \mid 7 \text{ instruments previous week}) &= P(N = 4) = \frac{4^4}{4!} \exp(-4) = 0.1954 \\
P(4 \text{ instruments this week} \mid 7 \text{ instruments previous week}) &= P(N = 3) = \frac{4^3}{3!} \exp(-4) = 0.1954 \\
P(5 \text{ instruments this week} \mid 7 \text{ instruments previous week}) &= P(N = 2) = \frac{4^2}{2!} \exp(-4) = 0.1465 \\
P(6 \text{ instruments this week} \mid 7 \text{ instruments previous week}) &= P(N = 1) = 4 \exp(-4) = 0.0733 \\
P(7 \text{ instruments this week} \mid 7 \text{ instruments previous week}) &= P(N = 0) + P(N \geq 5) = \\
&1 - 0.1954 - 0.1954 - 0.1465 - 0.0733 = 0.5665 = 0.3894
\end{aligned}$$

The one-step transition probability matrix is

	3	4	5	6	7
3	0.0183	0	0	0	0.9817
4	0.0733	0.0183	0	0	0.9084
5	0.1465	0.0733	0.0183	0	0.7619
6	0.1954	0.1465	0.0733	0.0183	0.5665
7	0.1954	0.1954	0.1465	0.0733	0.3894

(b) The following code generates inventory trajectories, assuming that the initial inventory size is randomly chosen.

```
#specifying transition probability matrix
tm<- matrix(c(0.0183, 0, 0, 0, 0.9817, 0.0733, 0.0183, 0, 0, 0.9084, 0.1465,
0.0733, 0.0183, 0, 0.7619, 0.1954, 0.1465, 0.0733, 0.0183, 0.5665, 0.1954,
0.1954, 0.1465, 0.0733, 0.3894), nrow=5, ncol=5, byrow=TRUE)

library(markovchain)
mc<- new("markovchain", transitionMatrix=tm, states=c("3", "4", "5", "6", "7"))

#specifying total number of steps
nsteps<- 25

#specifying seed
set.seed(8596943)

#specifying matrix containing states
MC.states<- matrix(NA, nrow=nsteps, ncol=2)

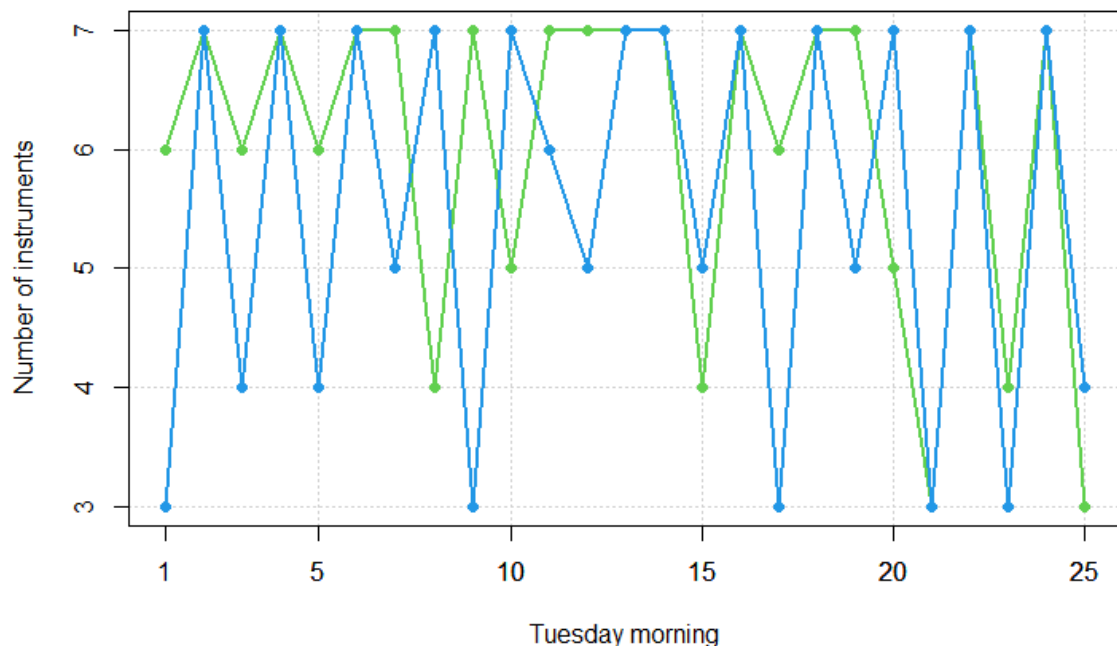
#simulating states

for (i in 1:2) {
  state0<- sample(3:7, 1, prob=c(0.2,0.2,0.2,0.2,0.2))
  MC.states[,i]<- rmarkovchain(n=nsteps-1, object=mc, t0=state0,
include.t0=TRUE)
}

#plotting simulated trajectories
matplot(MC.states, type="l", lty=1, lwd=2, col=3:4, axat="n", ylim=c(3,7),
xlab="Tuesday morning", ylab="Number of instruments", panel.first=grid())

axis(side=1, at=c(1,5,10,15,20,25))

points(1:nsteps, MC.states[,1], pch=16, col=3)
points(1:nsteps, MC.states[,2], pch=16, col=4)
```



(c) Using the definition of conditional probability and Markov property, we can write

$$\begin{aligned}
 P(X_4 = 7, X_3 = 7, X_2 = 7 \mid X_1 = 7) &= \frac{P(X_4 = 7, X_3 = 7, X_2 = 7, X_1 = 7)}{P(X_1 = 7)} \\
 &= \frac{P(X_4 = 7 \mid X_3 = 7, X_2 = 7, X_1 = 7) P(X_3 = 7, X_2 = 7, X_1 = 7)}{P(X_1 = 7)} \\
 &= \frac{P(X_4 = 7 \mid X_3 = 7) P(X_3 = 7 \mid X_2 = 7, X_1 = 7) P(X_2 = 7, X_1 = 7)}{P(X_1 = 7)} \\
 &= \frac{P(X_4 = 7 \mid X_3 = 7) P(X_3 = 7 \mid X_2 = 7) P(X_2 = 7 \mid X_1 = 7) P(X_1 = 7)}{P(X_1 = 7)} \\
 &= P_{77}P_{77}P_{77} = P_{77}^3 = (0.3894)^3 = 0.059.
 \end{aligned}$$

(d) Below we find the steady-state probability distribution for this Markov chain.

steadyStates(mc)

0.1487365³ 0.1300719⁴ 0.09080707⁵ 0.04379828⁶ 0.5865863⁷

Using these values, we compute the expected weekly storage cost:

$$\begin{aligned}
 E(\text{weekly storage cost}) &= \$5 \left((3)(0.1487365) + (4)(0.1300719) + (5)(0.09080707) \right. \\
 &\quad \left. + (6)(0.04379828) + (7)(0.5865863) \right) = (\$5)(5.789426) = \$28.94713.
 \end{aligned}$$

Thus, in the long run, there will be, on average, 5.789426 instruments in the store on Tuesday morning, and the average storage cost will amount to \$28.95.

CHAPTER 2

EXERCISE 2.1. In theory, $E(X_{50}) = ((2)(0.3) - 1)(50) = -20$ and $Var(X_{50}) = (4)(0.3)(1 - 0.3)(50) = 42$.

Next, we run an R code that simulates 10,000 trajectories of length 50 steps and computes the mean and variance of the last values.

```
#specifying parameters
p<- 0.3
n<- 50
ntraj<- 10000

#setting seed number
set.seed(546675)

#defining walk as matrix
walk<- matrix(NA, nrow=n, ncol=ntraj)

#simulating trajectories
for (j in 1:ntraj) {
  walk[1,j]<- 0
  for (k in 2:n) {
    walk[k,j]<- ifelse(runif(1)<p, walk[k-1,j]+1, walk[k-1,j]-1)
  }
}

mean(walk[50,])

-19.5824

var(walk[50,])

42.16583
```

The empirical values are pretty close to the theoretical ones.

EXERCISE 2.2. (a) The R script below simulates 10,000 trajectories and counts how many of them have a value of 0 at the 1,000th step.

```
#setting counter to zero
nzeros<- 0

#specifying seed
set.seed(675572)

#defining walk as matrix
walk<- c()

#simulating trajectories
for (j in 1:10000)
{
  walk[1]<- 0
  for (i in 2:1001)
```

```

walk[i]<- ifelse(runif(1)<0.5, walk[i-1]+1, walk[i-1]-1)
  if (walk[1001]==0) nzeros=nzeros+1
}

print(nzeros)

```

253

(b) The theoretical probability of returning to 0 on the 1,000th step is

$$P(X_{1000} = 0 \mid X_0 = 0) = \binom{1000}{500} \left(\frac{1}{2}\right)^{1000} = 0.025. \text{ This quantity was computed in R:}$$

```
choose(1000, 500)*0.5^1000
```

0.02522502

The estimated probability from part (a) is $\hat{P}(X_{1000} = 0 \mid X_0 = 0) = \frac{253}{10000} = 0.0253$, which is a pretty accurate estimate of the theoretical value.

EXERCISE 2.3. (a) The code below simulates the 10,000 trajectories of one-, two-, and three-dimensional symmetric random walks that start at the origin and continue for at most 1,000 steps. A trajectory that reaches the origin is terminated.

```

#setting counters to zero
n1D<- 0
n2D<- 0
n3D<- 0

#specifying seed
set.seed(300799)

#defining 1D walk as vector
walk1D<- c()
nsteps1D<- c()

#simulating 1D trajectories
for (j in 1:10000)
{
  walk1D[1]<- 0 #setting initial value to zero
  for (i in 2:1001)
  {
    walk1D[i]<- ifelse(runif(1)<0.5, walk1D[i-1]+1, walk1D[i-1]-1)
    if (walk1D[i]==0) {
      n1D=n1D+1
      break    }
  }
  nsteps1D[j]=i
}

#defining 2D walk as matrix
walk2D<- matrix(NA, nrow=1001, ncol=2)
nsteps2D<- c()

#defining random steps

```



```

rstep2D<- matrix(c(1, 0, -1, 0, 0, 1, 0, -1), nrow=4, ncol=2, byrow=TRUE)

#simulating 2D trajectories
for (j in 1:10000)
{
  walk2D[1,]<- c(0,0) #setting initial value to the origin
  for (i in 2:1001)
  {
    walk2D[i,]<- walk2D[i-1,]+rstep2D[sample(1:4, size=1),]

    if (walk2D[i,1]==0 & walk2D[i,2]==0) {
      n2D=n2D+1
      break
    }
  }
  nsteps2D[j]=i
}

#defining 3D walk as matrix
walk3D<- matrix(NA, nrow=1001, ncol=3)
nsteps3D<- c()

#defining random steps
rstep3D<- matrix(c(1, 0, 0,-1, 0, 0, 0, 1, 0, 0, -1, 0, 0, 0, 1, 0, 0, -1),
nrow=6, ncol=3, byrow=TRUE)

#simulating 3D trajectories
for (j in 1:10000)
{
  walk3D[1,]<- c(0,0,0) #setting initial value to the origin
  for (i in 2:1001)
  {
    walk3D[i,]<- walk3D[i-1,]+rstep3D[sample(1:6, size=1),]

    if (walk3D[i,1]==0 & walk3D[i,2]==0 & walk3D[i,3]==0) {
      n3D=n3D+1
      break
    }
  }
  nsteps3D[j]=i
}

print(n1D)

9756

print(n2D)

6759

print(n3D)

3329

```

Roughly 97.6% of the 1D trajectories returned to 0, about 67.6% of the 2D trajectories returned to (0, 0), and only 33.3% of the 3D trajectories returned to (0, 0, 0).

(a) The average number of steps it took those trajectories to return to the origin is computed as

```
mean(nsteps1D[nsteps1D!=1001])
```

27.47499

```
mean(nsteps2D[nsteps2D!=1001])
```

61.47625

```
mean(nsteps3D[nsteps3D!=1001])
```

27.83689

The 97.6% of the 1D trajectories that returned to the origin, did it in 27.47 steps, on average.
The 67.6% of the 2D trajectories that returned to the origin, did it in 61.48 steps, on average.
The 33.3% of the 3D trajectories that returned to the origin, did it in 27.84 steps, on average.

EXERCISE 2.4. (a) The R script below simulates the trajectories and terminates them if the barrier is hit. Otherwise, trajectories continue for 1,000 steps. The total number of trajectories that hit the barrier is counted. We also record the number of steps (for part (b)) and the y-coordinate (for part (c)).

```
#setting counter to zero
nhits<- 0

#specifying seed
set.seed(50118)

#defining walk as matrix
walk<- matrix(NA, nrow=1001, ncol=2)
nsteps<- c()
ycoord<- c()

#defining random steps
rstep<- matrix(c(1, 0, -1, 0, 0, 1, 0, -1), nrow=4, ncol=2, byrow=TRUE)

#simulating trajectories
for (j in 1:10000)
{
  walk[1,]<- c(0,0) #setting initial value to the origin
  for (i in 2:1001)
  {
    walk[i,]<- walk[i-1,] + rstep[sample(1:4, size=1),]

    if (walk[i,1]==30) {
      nhits=nhits+1
      break
    }
  }
  nsteps[j]<- i
  ycoord[j]<- ifelse(i==1001, 99999, walk[i,2])
}

print(nhits)
```

1764

So, of the 10,000 trajectories, 1,764 hit the vertical barrier. Thus, the estimated probability to hit the barrier is 0.1764.

(b) The average number of steps it takes a trajectory to hit the barrier, provided it did hit the barrier within the 1,000 steps, is estimated as

```
mean(nsteps[nsteps!=1001])
```

623.2053

It took on average 623.2 steps to hit the barrier for the 17.64% of the trajectories that terminated at the barrier.

(a) Estimate the expected value of the y-coordinate at the time when the random walk hits the barrier. What should this value be from the theoretical point of view? Hint: deduce from a symmetry argument.

```
mean(ycoord[ycoord!=99999])
```

0.1066364

The estimated average y-coordinate for 17.64% of the trajectories that hit the barrier was 0.1066. From the theoretical viewpoint, using the symmetry of the random walk, we can argue that the y-coordinate should be equal to 0.

EXERCISE 2.5. By running the following script, we simulate trajectories and calculate the number of those that hit the barrier. The plot is given below.

```
Nhits<- c()

#specifying seed
set.seed(96770)

#defining walk as matrix
walk<- matrix(NA, nrow=1001, ncol=2)

#defining random steps
rstep<- matrix(c(1, 0, -1, 0, 0, 1, 0, -1), nrow=4, ncol=2, byrow=TRUE)

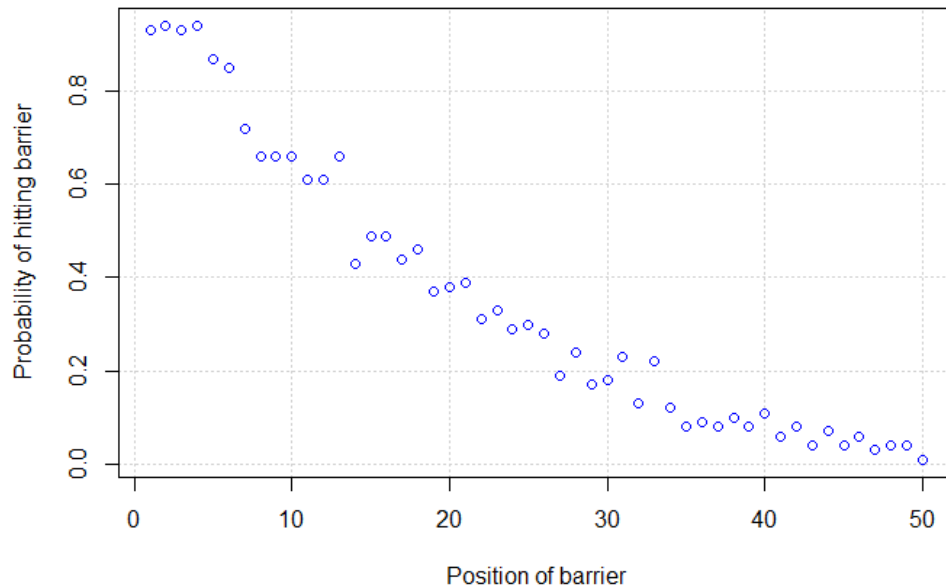
#varying the barrier value
for (barrier in 1:50) {
  nhits<- 0
  #simulating trajectories
  for (j in 1:100)
  {
    walk[1,]<- c(0,0) #setting initial value to the origin
    for (i in 2:1001)
    {
      walk[i,]<- walk[i-1,] + rstep[sample(1:4, size=1),]

      if (walk[i,1]==barrier) {
        nhits=nhits+1
        break
      }
    }
  }
  Nhits[barrier]=nhits
}
```

```
print(Nhits)
```

```
[1] 93 94 93 94 87 85 72 66 66 66 61 61
[13] 66 43 49 49 44 46 37 38 39 31 33 29
[25] 30 28 19 24 17 18 23 13 22 12 8 9
[37] 8 10 8 11 6 8 4 7 4 6 3 4
[49] 4 1
```

```
plot(1:50, Nhits/100, col="blue", xlab="Position of barrier", ylab="Probability of
hitting barrier", panel.first=grid())
```



We see that as the barrier value increases from 1 to 50, the estimated probability of hitting this barrier decreases from 0.93 to 0.01, in a slightly curvilinear (convex downward) manner.

EXERCISE 2.6. The lines of code given below terminate each trajectory if it reaches a side of the square. The total number of steps required is recorded for each trajectory. The average value is computed at the end.

```
walk<- data.frame()
nsteps<- c()

set.seed(37440)

#defining random steps
rstep<- matrix(c(1, 0, -1, 0, 0, 1, 0, -1), nrow=4, ncol=2, byrow=TRUE)

#simulating trajectories
for (j in 1:1000)
{
  walk[1,1]<- 0
  walk[1,2]<- 0
  i<- 2

  repeat{
    walk[i,]<- walk[i-1,] + rstep[sample(1:4, size=1),]
```

```

    if (walk[i,1]==10 | walk[i,1]==-10 | walk[i,2]==10 | walk[i,2]==-10)
    { break }
    else i=i+1
    }

    nsteps[j]=i
}

mean(nsteps)

120.898

```

Hence, the average number of steps it takes for the random walk to reach the square is estimated as 120.898 steps.

EXERCISE 2.7. (a) Conditioning on the outcome of the first step, we see that the probability \mathbf{P}_i solves the recurrence relation $\mathbf{P}_i = p\mathbf{P}_{i+1} + q\mathbf{P}_{i-1}$ with the border constraints $\mathbf{P}_A = 0$ and $\mathbf{P}_B = 1$. Assuming first that $\frac{q}{p} \neq 1$, we look for the solution in the form $\mathbf{P}_i = c\left(\frac{q}{p}\right)^i + d$ where c and d are some constants that can be found from the boundary conditions: $\mathbf{P}_A = 0 = c\left(\frac{q}{p}\right)^A + d$ and $\mathbf{P}_B = 1 = c\left(\frac{q}{p}\right)^B + d$. From here, $c = -\frac{1}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B}$ and $d = \frac{\left(\frac{q}{p}\right)^A}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B}$, and thus, $\mathbf{P}_i = \frac{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^i}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B}$.

Now assume $\frac{q}{p} = 1$. We look for the solution of the recurrence relation in the form $\mathbf{P}_i = ci + d$. Again, from the boundary conditions, $\mathbf{P}_A = 0 = cA + d$ and $\mathbf{P}_B = 1 = cB + d$. Hence, $c = \frac{1}{B-A}$ and $d = -\frac{A}{B-A}$, leading to $\mathbf{P}_i = \frac{i-A}{B-A}$.

(b) By conditioning on the first step, we see right away that the expectation satisfies the recurrence relation $\mathbf{E}_i = p\mathbf{E}_{i+1} + q\mathbf{E}_{i-1} + 1$ with the boundary conditions $\mathbf{E}_A = \mathbf{E}_B = 0$. Because of the additive constant term, this equation is referred to as a non-homogeneous relation and the general solution is sought in the form $\mathbf{E}_i = c\left(\frac{q}{p}\right)^i + d + \frac{i}{q-p}$, if $\frac{q}{p} \neq 1$, and $\mathbf{E}_i = ci + d - i^2$, if $\frac{q}{p} = 1$. The constants c and d are found from the boundary conditions. In the former case, they satisfy $\mathbf{E}_A = 0 = c\left(\frac{q}{p}\right)^A + d + \frac{A}{q-p}$ and $\mathbf{E}_B = 0 = c\left(\frac{q}{p}\right)^B + d + \frac{B}{q-p}$. Whence,

$$c = \frac{B-A}{q-p} \cdot \frac{1}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B},$$

and

$$d = -\frac{B-A}{q-p} \cdot \frac{\left(\frac{q}{p}\right)^B}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B} - \frac{B}{q-p},$$

resulting in

$$E_i = \frac{B-A}{q-p} \cdot \frac{\left(\frac{q}{p}\right)^i - \left(\frac{q}{p}\right)^B}{\left(\frac{q}{p}\right)^A - \left(\frac{q}{p}\right)^B} - \frac{B-i}{q-p}.$$

In the latter case, c and d solve $E_A = 0 = cA + d - A^2$ and $E_B = 0 = cB + d - B^2$. From here, $c = A + B$ and $d = -AB$. Thus, $E_i = (A + B)i - AB - i^2 = (B - i)(i - A)$.

(c) We use the formulas derived above with $p = 0.47$, $q = 0.53$, $A = 10$, $i = 40$, and $B = 80$. We obtain

$$P_{40} = \frac{\left(\frac{0.53}{0.47}\right)^{10} - \left(\frac{0.53}{0.47}\right)^{40}}{\left(\frac{0.53}{0.47}\right)^{10} - \left(\frac{0.53}{0.47}\right)^{80}} = 0.007962,$$

and

$$E_{40} = \frac{80-10}{0.53-0.47} \cdot \frac{\left(\frac{0.53}{0.47}\right)^{40} - \left(\frac{0.53}{0.47}\right)^{80}}{\left(\frac{0.53}{0.47}\right)^{10} - \left(\frac{0.53}{0.47}\right)^{80}} - \frac{80-40}{0.53-0.47} = 490.7115.$$

The probability of doubling the fortune in this rigged game is very small (about 0.008), and the gambler will play, on average, about 491 games before he walks out of the casino.

Below we give the code that simulates 10,000 trajectories and computes the proportion of them that ended in \$80 (as opposed to \$10) and averages the number of games in each trajectory.

```
#specifying parameters
p<- 0.47
ntraj<- 10000

#setting seed number
set.seed(314159)

#defining walk as matrix
walk<- data.frame(NULL)

#setting counters
n80<- 0
n10<- 0
ngames<- 0

#simulating trajectories
for (j in 1:ntraj) {
  walk[1,j]<- 40
  k<- 2
  repeat {
    walk[k,j]<- ifelse(runif(1)<p, walk[k-1,j]+1, walk[k-1,j]-1)
    if(walk[k,j]==80) {
      n80<- n80+1
      break
    }
  }
}
```

```

    }
    if(walk[k,j]==10) {
      n10<- n10+1
      break
    }
    k<- k+1
    ngames<- ngames+1
  }
}

print(prop.n80<- n80/ntraj)

0.0084

print(avg.ngames<- ngames/ntraj)

488.4926

```

EXERCISE 2.8. The student's visit to the museum can be modeled as a random walk on a graph with the state space $S = \{Exit, A, B, C, D, E, F\}$, and the transition probability matrix \mathbf{P}

	<i>Exit</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>
<i>Exit</i>	1	0	0	0	0	0	0
<i>A</i>	1/3	0	1/3	0	1/3	0	0
<i>B</i>	0	1/2	0	1/2	0	0	0
<i>C</i>	0	0	1/3	0	1/3	1/3	0
<i>D</i>	0	1/2	0	1/2	0	0	0
<i>E</i>	0	0	0	1/2	0	0	1/2
<i>F</i>	0	0	0	0	0	1	0

We assume that at the beginning of the walk, the student enters the museum and finds himself in Room A. The expected number of transitions between the rooms until he reaches the exit is given by the formula

$$\begin{aligned}
 E(\# \text{ of transitions}) &= (0, 1, 0, 0, 0, 0, 0) \left((1)(\mathbf{P}) + (2)(\mathbf{P}^2 - \mathbf{P}) + (3)(\mathbf{P}^3 - \mathbf{P}^2) + (4)(\mathbf{P}^4 - \mathbf{P}^3) \right. \\
 &\quad \left. + \dots \right) (1, 0, 0, 0, 0, 0, 0)^{-1}
 \end{aligned}$$

We submit the following R code that approximates this sum. The convergence is achieved with 146 terms.

```

#specifying the transition probability matrix
tm<- matrix(c(1,0,0,0,0,0,0,1/3,0,1/3,0,1/3,0,0,0,1/2,0,1/2,0,0,0,0,0,0,
1/3,0,1/3,1/3,0,0,1/2,0,1/2,0,0,0,0,0,1/2,0,0,1/2,0,0,0,0,0,1,0),
nrow=7, ncol=7, byrow=TRUE)

#setting counter
ntrans<- 0

#computing expected number of transitions
p<- matrix(NA, nrow=146, ncol=7)
p[1,]<- c(0,1,0,0,0,0,0)

```

```
for (i in 2:146) {  
  p[i,]<- p[i-1,]*%tm  
  ntrans<- ntrans+(i-1)*(p[i,1]-p[i-1,1])  
}
```

```
print(ntrans)
```

12.94225

Thus, the student will make, on average, 12.94225 transitions between the rooms. Since he spends 30 minutes in each room, the total average length of visit will be $(30)(12.94225) = 388.2675$ minutes (or 6 hours and 28.3 minutes). On an “average” visit, he will be done before the museum closes for the day.

CHAPTER 3

EXERCISE 3.1. We use the independence and stationarity of increments of a Poisson process to derive the expression for the joint probability distribution. We write

$$\begin{aligned}
 P(N(s) = m, N(t) = n) &= P(N(t) - N(s) = n - m, N(s) = m) \\
 &= P(N(t) - N(s) = n - m)P(N(s) = m) = P(N(t - s) = n - m)P(N(s) = m) \\
 &= \frac{(\lambda(t - s))^{n-m}}{(n - m)!} e^{-\lambda(t-s)} \cdot \frac{(\lambda s)^m}{m!} e^{-\lambda s} = \frac{(t - s)^{n-m} s^m}{(n - m)! m!} \lambda^n e^{-\lambda t} \\
 &= \binom{n}{m} \left(\frac{s}{t}\right)^m \left(1 - \frac{s}{t}\right)^{n-m} \frac{(\lambda t)^n}{n!} e^{-\lambda t}.
 \end{aligned}$$

EXERCISE 3.2. Assume that $s < t$. We compute the covariance function, using the independence and stationarity of the increments. We have

$$\begin{aligned}
 \text{Cov}(N(s), N(t)) &= E[N(s)N(t)] - E[N(s)]E[N(t)] \\
 &= E[(N(t) - N(s) + N(s))N(s)] - E[N(s)]E[N(t)] \\
 &= E[(N(t) - N(s))N(s)] + E[N(s)]^2 - E[N(s)]E[N(t)] \\
 &= E[N(t) - N(s)]E[N(s)] + \text{Var}[N(s)] + [E(N(s))]^2 - E[N(s)]E[N(t)] \\
 &= E[N(t - s)]E[N(s)] + \text{Var}[N(s)] + [E(N(s))]^2 - E[N(s)]E[N(t)] \\
 &= \lambda(t - s)\lambda s + \lambda s + (\lambda s)^2 - \lambda s \lambda t = \lambda s.
 \end{aligned}$$

EXERCISE 3.3. (a) $P(N(5) = 16 \mid N(1) = 2, N(2) - N(1) = 3)$

$$\begin{aligned}
 &= \frac{P(N(5) - N(2) = 11, N(2) - N(1) = 3, N(1) = 2)}{P(N(2) - N(1) = 3, N(1) = 2)} \\
 &= \frac{P(N(3) = 11)P(N(1) = 3)P(N(1) = 2)}{P(N(1) = 3)P(N(1) = 2)} = P(N(3) = 11) = \frac{((5)(3))^{11}}{11!} e^{-(5)(3)} = 0.0663.
 \end{aligned}$$

(b) Since $E(S_{100}) = (100) \left(\frac{1}{5}\right) = 20$, the 100th claim is expected to be seen on the 20th business day, that is, on January 27th.

JANUARY					
Monday	2	9	16	23	30
Tuesday	3	10	17	24	31
Wednesday	4	11	18	25	

Thursday	5	12	19	26	
Friday	6	13	20	27	
Saturday	7	14	21	28	
Sunday	8	15	22	29	

EXERCISE 3.4. (a) Phone calls that result in sales occur with rate $(0.15) \left(\frac{60}{5}\right) = 1.8$ per hour.

Therefore, in the next two hours, there will be, on average $(2)(1.8)=3.6$ successful sales.

(b) The total number of phone calls is a Poisson process with a rate of $60/5=12$ per hour. Phone calls that result in a sale and those that don't form independent Poisson processes with rates 1.8 and 10.2 per hour, respectively. Therefore,

$$\begin{aligned} P(N(1) = 15, N_{\text{sale}}(1) = 5) &= P(N_{\text{sale}}(1) = 5, N_{\text{no sale}}(1) = 10) \\ &= P(N_{\text{sale}}(1) = 5)P(N_{\text{no sale}}(1) = 10) = \frac{(1.8)^5}{5!} e^{-1.8} \frac{(10.2)^{10}}{10!} e^{-10.2} = 0.00325. \end{aligned}$$

$$(c) P(N(4) = 10 | N(1) = 3) = P(N(4) - N(1) = 7) = P(N(3) = 7) = \frac{((1.8)(3))^7}{7!} e^{-(1.8)(3)} = 0.119987.$$

EXERCISE 3.5. (a) $N_1(t)$ and $N_2(t)$ are splitted Poisson processes with the means

$$\begin{aligned} E(N_1(t)) &= \lambda \int_0^t P(\text{disease is contracted at time } s, \text{ symptoms show by time } t) ds \\ &= \lambda \int_0^t F(t-s) ds = \{u = t-s\} = \lambda \int_0^t F(u) du, \end{aligned}$$

and

$$\begin{aligned} E(N_2(t)) &= \lambda \int_0^t P(\text{disease is contracted at time } s, \text{ no symptoms show by time } t) ds \\ &= \lambda \int_0^t (1 - F(t-s)) ds = \{u = t-s\} = \lambda \int_0^t (1 - F(u)) du. \end{aligned}$$

(b) Suppose by a fixed time t , $\hat{E}(N_1(t))$ individuals are observed who show symptoms of a disease. From here, we can estimate the rate of contracting the disease as $\hat{\lambda} = \frac{\hat{E}(N_1(t))}{\int_0^t F(u) du}$. Plugging this into the expression for the expected value of $N_2(t)$, we can calculate the estimated number of individuals infected but not yet showing symptoms by time t as

$$\hat{E}(N_2(t)) = \frac{\hat{E}(N_1(t)) \int_0^t (1 - F(u)) du}{\int_0^t F(u) du}.$$

(c) Suppose the incubation period until symptoms show is an exponentially distributed random variable with a mean of 2 days. Thus, $F(u) = 1 - e^{-u/2}$, $u \geq 0$. Given that $\hat{E}(N_1(10)) = 1000$, we estimate the number of individuals who are infected but haven't shown the symptoms yet as

$$\hat{E}(N_2(10)) = \frac{1000 \int_0^{10} e^{-\frac{u}{2}} du}{\int_0^{10} \left(1 - e^{-\frac{u}{2}}\right) du} = \frac{(1000)(2) \left(1 - e^{-\frac{10}{2}}\right)}{10 - (2) \left(1 - e^{-\frac{10}{2}}\right)} = 247.8979,$$

or about 248 individuals.

EXERCISE 3.6. (a) Let $N(t)$ denote the number of high road surface distress areas on a t -mile stretch of the road. It is a Poisson process with a rate $\lambda = 2.8$. So, $E(N(10)) = (2.8)(10) = 28$.

(b) The code below simulates 30 distances between distressed surface areas. These distances are independent and exponentially distributed with mean $\frac{1}{2.8} = 0.357143$ miles.

```
#specifying parameters
lambda<- 2.8
Nareas<- 30

#defining states
N<- 0:Nareas

#setting distance as vector
dist<- c()

#setting initial value for distance
dist[1]<- 0

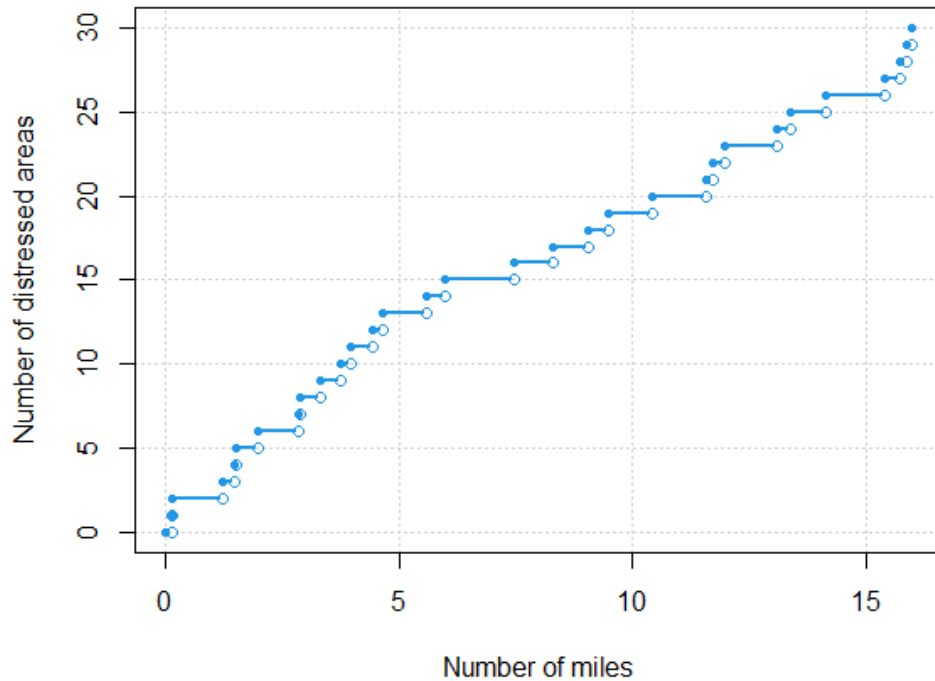
#specifying seed
set.seed(777754)

for (i in 2:(Nareas+1))
  dist[i]<- dist[i-1] + round((-1/lambda)*log(runif(1)),2)

#plotting trajectory
plot(dist, N, type="n", xlab="Number of miles", ylab="Number of distressed areas",
panel.first = grid())

segments(dist[-length(dist)],N[-length(dist)], dist[-1]-0.07, N[-length(dist)],
lwd=2, col=4)

points(dist, N, pch=20, col=4)
points(dist[-1],N[-length(dist)],pch=1, col=4)
```



In this simulation, the total length of the road that contains 30 distressed surface areas is the last value in the vector `dist`, that is, 15.99 miles.

```
dist[length(dist)]
```

15.99

(c) Given that $N(10) = 30$, the distances between distressed surface areas are distributed as order statistics of the uniform distribution on the interval $(0,10)$. The R code below simulates the locations of those areas.

```
#specifying parameters
D<- 10
Nareas<- 30

#specifying seed
set.seed(87998)

#defining states
N<- 0:Nareas

#generating N standard uniforms
u<- c()
u[1]<- 0

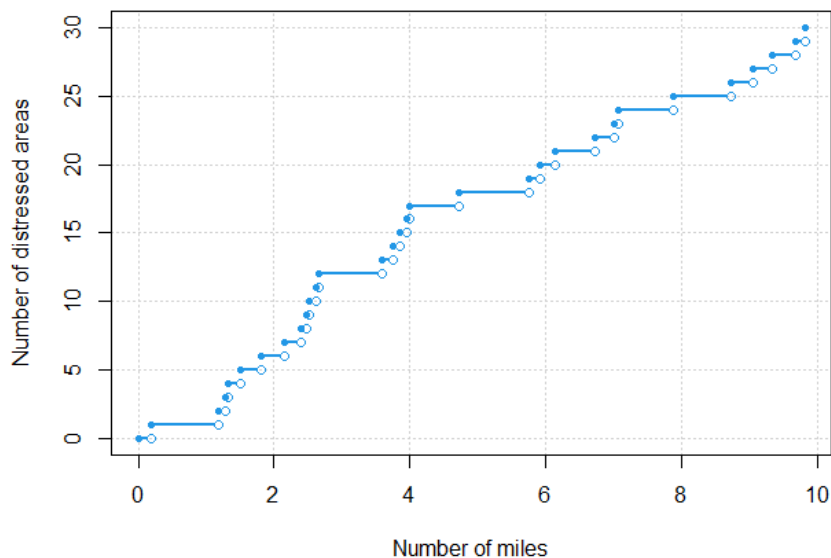
for(i in 2:(Nareas+1))
  u[i]<- runif(1)

#computing event distances
dist<- D*sort(u)

#plotting trajectory
plot(dist, N, type = "n", xlab="Number of miles", ylab="Number of distressed
areas", panel.first = grid())
```

```
segments(dist[-length(dist)],N[-length(dist)], dist[-1]-0.07, N[-length(dist)],
lwd=2, col=4)
```

```
points(dist, N, pch=20, col=4)
points(dist[-1],N[-length(dist)],pch=1, col=4)
```



EXERCISE 3.7. For the data set on significant volcanic eruptions between 1920 and 2020, the code below calculates interarrival times, plots a histogram, and conducts the goodness-of-fit test. The p-value for the test is larger than 0.05, indicating that the Poisson process models the data well.

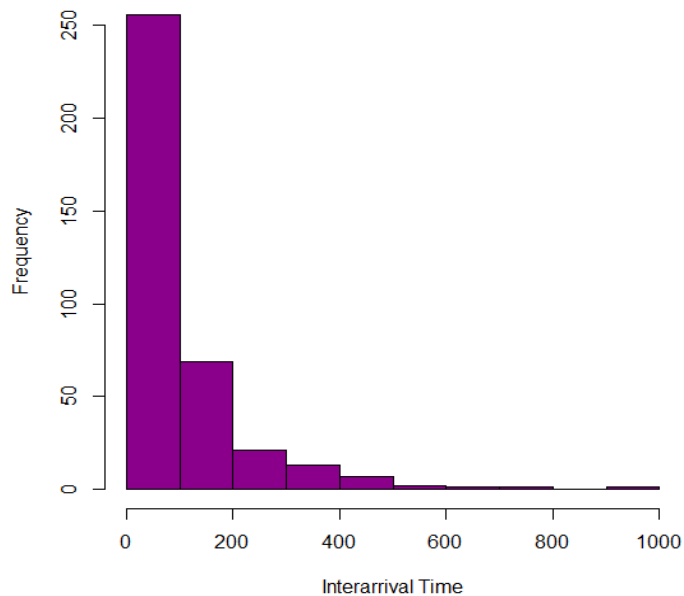
```
volcanoes.data<- read.csv(file="./volcanoesdata.csv", header=TRUE, sep=",")

#creating date-time variable
datetime<- as.POSIXct(paste(as.Date(volcanoes.data$DATE), volcanoes.data$TIME))

#computing lag
datetime.lag<- c(0,head(datetime, -1))

#computing interarrival times (in hours)
int<- (as.numeric(datetime)-as.numeric(datetime.lag))/(3600*24)
int<- int[-1] # removing first value

#plotting histogram
hist(int, main="", xlab="Interarrival Time", col="dark magenta")
```



```
#binning interarrival times
binned.int<- as.factor(ifelse(int<25,"1", ifelse(int>=25 & int<50,"2",
ifelse(int>=50 & int<100,"3", ifelse(int>=100 & int<150,"4",ifelse(int>=150 &
int<200,"5", ifelse(int>=200 & int<250,"6", "7"))))))))
```

```
#computing observed frequencies
obs<- table(binned.int)
```

```
#estimating mean for exponential distribution
mean.est<- mean(int)
```

```
#computing expected frequencies
exp<- c(1:7)
exp[1]<- length(int)*(1-exp(-25/mean.est))
exp[2]<- length(int)*(exp(-25/mean.est)-exp(-50/mean.est))
exp[3]<- length(int)*(exp(-50/mean.est)-exp(-100/mean.est))
exp[4]<- length(int)*(exp(-100/mean.est)-exp(-150/mean.est))
exp[5]<- length(int)*(exp(-150/mean.est)-exp(-200/mean.est))
exp[6]<- length(int)*(exp(-200/mean.est)-exp(-250/mean.est))
exp[7]<- length(int)*exp(-250/mean.est)
```

```
obs
```

```
1      2      3      4      5      6      7
96     64     94     42     29     13     33
```

```
round(exp,1)
```

```
83.3  64.6  88.9  53.5  32.2  19.3  29.2
```

```
#computing chi-squared statistic
print(chi.sq<- sum((obs-exp)^2/exp))
```

```
7.581284
```

```
#computing p-value
print(p.value<- 1-pchisq(chi.sq, df=5))
```

```
0.1808718
```

EXERCISE 3.8. (a) Team A scores as a Poisson process with a rate $\lambda_A = (0.25 + 0.40 + 0.20)(0.5) = (0.85)(0.5) = 0.425$ per minute. Team B scores as a Poisson process with a rate $\lambda_B = (0.25 + 0.50 + 0.15)(0.4) = (0.9)(0.4) = 0.36$ per minute. Hence,

$$E(\text{time until a team scores}) = \frac{1}{\lambda_A + \lambda_B} = \frac{1}{0.425 + 0.36} = 1.273885 \text{ minutes.}$$

$$(b) E(\text{time until team A scores}) = \frac{1}{\lambda_A} = \frac{1}{0.425} = 2.352941 \text{ minutes.}$$

$$E(\text{time until team B scores}) = \frac{1}{\lambda_B} = \frac{1}{0.36} = 2.777778 \text{ minutes.}$$

$$(c) P(\text{team A scores before team B}) = \frac{\lambda_A}{\lambda_A + \lambda_B} = \frac{0.425}{0.425 + 0.36} = 0.541401,$$

$$P(\text{team B scores before team A}) = 1 - 0.541401 = 0.458599.$$

(d) For team A, 1-pointers occur as an independent Poisson process with rate $(0.25)(0.5) = 0.125$ per minute; 2-pointers occur as an independent Poisson process with rate $(0.4)(0.5) = 0.2$ per minute; and 3-pointers occur as a Poisson process with rate $(0.20)(0.5) = 0.1$ per minute. For team B, the respective rates are $(0.25)(0.4) = 0.1$ per minute, $(0.5)(0.4) = 0.2$ per minute, and $(0.15)(0.4) = 0.06$ per minute. Therefore,

$$\begin{aligned} & P(\text{teams score same \# of 1-pointers, 2-pointers, and 3-pointers}) \\ &= P(\text{same \# of 1-pointers})P(\text{same \# of 2-pointers})P(\text{same \# of 3-pointers}) \\ &= \left[e^{-(0.125+0.1)(48)} \sum_{n=0}^{\infty} \frac{((0.125)(0.1)(48)^2)^n}{(n!)^2} \right] \left[e^{-(0.2+0.2)(48)} \sum_{n=0}^{\infty} \frac{((0.2)(0.2)(48)^2)^n}{(n!)^2} \right] \\ &\quad \times \left[e^{-(0.1+0.06)(48)} \sum_{n=0}^{\infty} \frac{((0.1)(0.06)(48)^2)^n}{(n!)^2} \right] = 0.0012. \end{aligned}$$

#computing probability of same number of 1-pointers, 2-pointers, and 3-pointers

```
sum1<- 0
for(n in 0:16)
  sum1<- sum1+(0.125*0.1*48^2)^n/(factorial(n))^2

print(p1<- sum1*exp(-(0.125+0.1)*48))
```

0.1152988

```
sum2<- 0
for(n in 0:23)
  sum2<- sum2+(0.2*0.2*48^2)^n/(factorial(n))^2
```

```
print(p2<- sum2*exp(-(0.2+0.2)*48))
```

0.09165684

```
sum3<- 0
for(n in 0:12)
  sum3<- sum3+(0.1*0.06*48^2)^n/(factorial(n))^2
```

```
print(p3<- sum3*exp(-(0.1+0.06)*48))
```

0.1167362

```
p1*p2*p3
```

0.001233658

EXERCISE 3.9. (a) The spider will need $\tau = 30$ minutes = 0.5 hours to reach the top. The rate of rain is $\lambda = 2$ per hour. Denote by T the total time it takes the spider to reach the top. The expected value of T is $E(T) = \frac{1}{2}(e^{(0.5)(2)} - 1) = 0.859141$ hours or 51.5 minutes.

(b) Let N denote the number of times the spider will be washed down before it reaches the top. Then, $E(N) = e^{(0.5)(2)} - 1 = 1.718282$.

CHAPTER 4

EXERCISE 4.1. (a) The number of broken calculators can be modeled according to a nonhomogeneous Poisson process $\{N(t), t \geq 0\}$ with the intensity rate function

$$\lambda(t) = \begin{cases} 3, & \text{if } 0 \leq t \leq 3, \\ 2t - 3, & \text{if } 3 \leq t \leq 10. \end{cases}$$

The integrated rate function is

$$\Lambda(t) = \int_0^t \lambda(u) du = \begin{cases} \int_0^t 3 du = 3t, & \text{if } 0 \leq t \leq 3, \\ 9 + \int_3^t (2u - 3) du = t^2 - 3t + 9, & \text{if } 3 \leq t \leq 10. \end{cases}$$

The probability mass function is

$$P(N(t) - N(s) = n) = \frac{(\Lambda(t) - \Lambda(s))^n}{n!} e^{-(\Lambda(t) - \Lambda(s))} = \frac{(t^2 - s^2 - 3(t - s))^n}{n!} e^{-(t^2 - s^2 - 3(t - s))}.$$

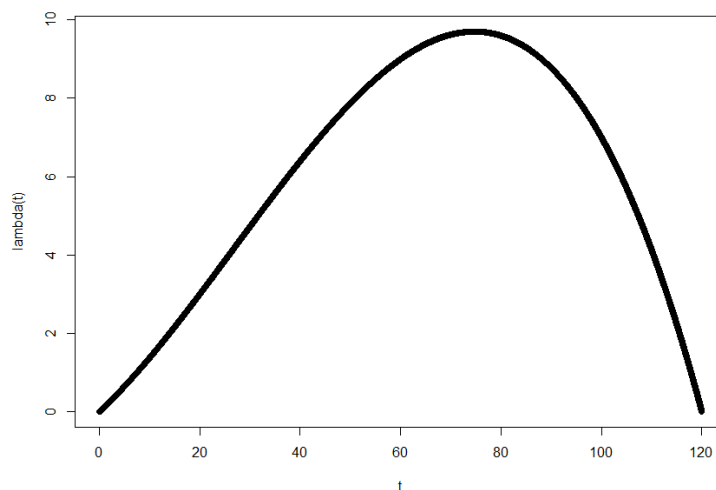
$$(b) P(N(8) - N(4) = 50) = \frac{(8^2 - 4^2 - 3(8-4))^{50}}{50!} e^{-(8^2 - 4^2 - 3(8-4))} = 0.004983.$$

$$(c) E(N(10) - N(2)) = \Lambda(10) - \Lambda(2) = 10^2 - (3)(10) + 9 - (3)(2) = 73.$$

EXERCISE 4.2. (a) Below is the R code and plot of the intensity function $\lambda(t) = -0.000025 t^3 + 0.002 t^2 + 0.12t$ against t on the interval $[0, 120]$.

```
lambda<- function(t) -0.000025*t^3+0.002*t^2+0.12*t
t<- seq(0, 120, by = 0.01)

plot(t, lambda(t))
```

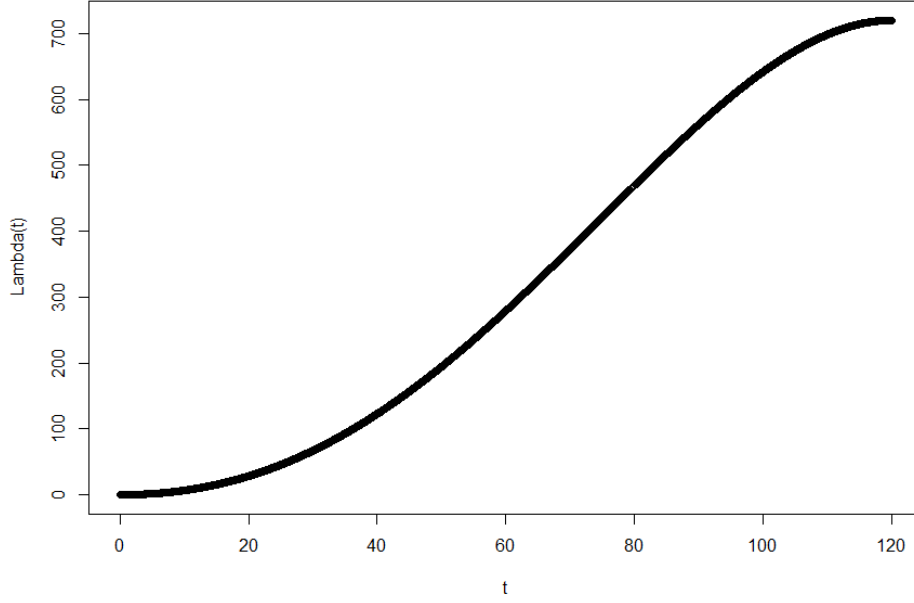


The intensity function looks like a skewed upside-down parabola that is equal to 0 at $t = 0$ and $t = 120$. It achieves the maximum at time t that solves the equation $\lambda'(t) = 0$. Therefore, t solves the quadratic equation $(-0.000025)(3)t^2 + (0.002)(2)t + 0.12 = 0$, which simplifies to $-0.00075t^2 + 0.4t + 12 = 0$. We need the solution that lies between 0 and 120, therefore, $t = \frac{0.4 + \sqrt{0.52}}{0.015} = 74.74068$ days. Thus the peak intensity rate occurs 74.74 days into the fire season, and the maximum number of fires per day is $= -0.000025 (74.74068)^3 + 0.002 (74.74068)^2 + 0.12(74.74068) = 9.703286$, or roughly 9.7 fires per day.

(b) We write $\Lambda(t) = \int_0^t \lambda(u) du = \int_0^t (-0.000025 u^3 + 0.002 u^2 + 0.12u) du = -0.00000625t^4 + 0.0006667t^3 + 0.06t^2, 0 \leq t \leq 120$. The R code and the graph are given below.

```
Lambda<- function(t) -0.00000625*t^4+0.0006667*t^3+0.06*t^2
t<- seq(0, 120, by = 0.01)

plot(t, Lambda(t))
```



To find the average number of wildfires per season we compute $\Lambda(120) = -(0.00000625)(120)^4 + (0.0006667)(120)^3 + (0.06)(120)^2 = 720.0576$, so, on average, about 720 fires occur in this area every season.

(a) The middle 50% of the fire season falls between day 30 and day 90. The mean number of wildfires in this period is $\Lambda(90) - \Lambda(30) = -(0.00000625)(90)^4 + (0.0006667)(90)^3 + (0.06)(90)^2 - (-(0.00000625)(30)^4 + (0.0006667)(30)^3 + (0.06)(30)^2) = 561.9618 - 66.9384 = 495.0234$.

EXERCISE 4.3. (a) The integrated intensity function is

$$\Lambda(t) = \int_0^t \lambda(u) du = \int_0^t \frac{A}{\sqrt{u}} du = 2A\sqrt{t}, t \geq 0.$$

We know that $\Lambda(1) = 30$. Thus, $A = 15$.

(b) Given that the n th injury occurred at the time s_n , the conditional cumulative distribution function of the time until the next injury T_{n+1} , $n \geq 1$, is $F_{T_{n+1}|s_n}(t|s_n) = 1 - \exp\{-30(\sqrt{t+s_n} - \sqrt{s_n})\}$, $t \geq 0$. The cdf of T_1 is $F_{T_1}(t) = 1 - \exp\{-30\sqrt{t}\}$, $t \geq 0$. To simulate observations from this distribution, we first generate standard uniform random variables U_n , $n \geq 1$, and then sequentially solve $F_{T_{n+1}|s_n}(t|s_n) = U_{n+1}$, i.e., we solve for t the identities $1 - \exp\{-30(\sqrt{t+s_n} - \sqrt{s_n})\} = U_n$. Replacing $1 - U_n$ by U_n since both have the same standard uniform distribution, we obtain

$$S_1 = \left(-\frac{1}{30}\ln(U_n)\right)^2, \quad \text{and} \quad S_{n+1} = \left(\sqrt{S_n} - \frac{1}{30}\ln(U_n)\right)^2, \quad n \geq 1.$$

The code and the simulated trajectory follow.

```
#specifying parameters
```

```

ninjrs<- 100

#defining states
N<- 0:ninjrs

#defining times as vectors
time<- c()

#specifying seed
set.seed(933541)

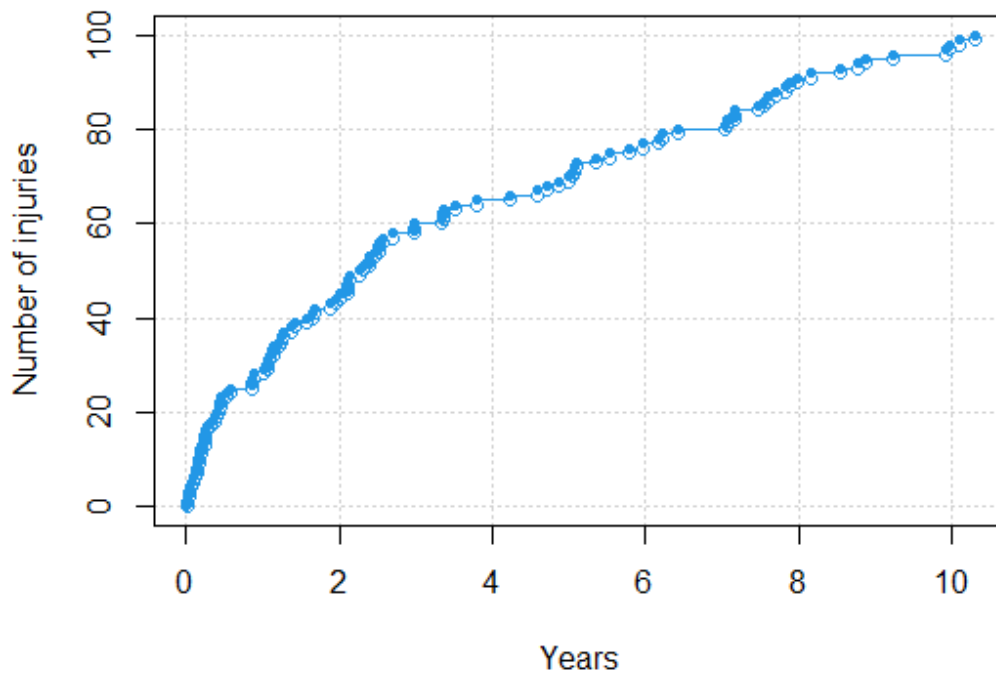
#computing event times
time[1]<- (-log(runif(1))/30)^2
for(i in 2:(ninjrs+1))
  time[i]<- (sqrt(time[i-1])-log(runif(1))/30)^2

#plotting simulated trajectory
plot(time, N, type="n", xlab="Years", ylab="Number of injuries",
panel.first=grid())

segments(time[-length(time)], N[-length(time)], time[-1]-0.07,
N[-length(time)], col=4)

points(time, N, pch=20, col=4)
points(time[-1], N[-length(time)], pch=1, col=4)

```



```

time[201]
9.870481

```

The time range of the trajectory is [0, 9.870481].

(c) We are given that $S_{100} = 12.25$ years. The times of injuries S_1, S_2, \dots, S_{99} are then represented by order statistics from the distribution with the cumulative distribution function $\frac{\Lambda(s)}{\Lambda(S_{100})} = \frac{30\sqrt{s}}{30\sqrt{S_{100}}} = \sqrt{s/S_{100}}$. The code that simulates a trajectory and the graph follow.

```

t<- 12.25
Lambda<- 30*sqrt(t)

#generating N(t)
set.seed(1133664)
ninjrs<- rpois(1,Lambda)

#defining states
N<- 0:ninjrs

#generating standard uniforms
u<- c()
u[1]<- 0
for(i in 2:(ninjrs+1))
u[i]<- runif(1)

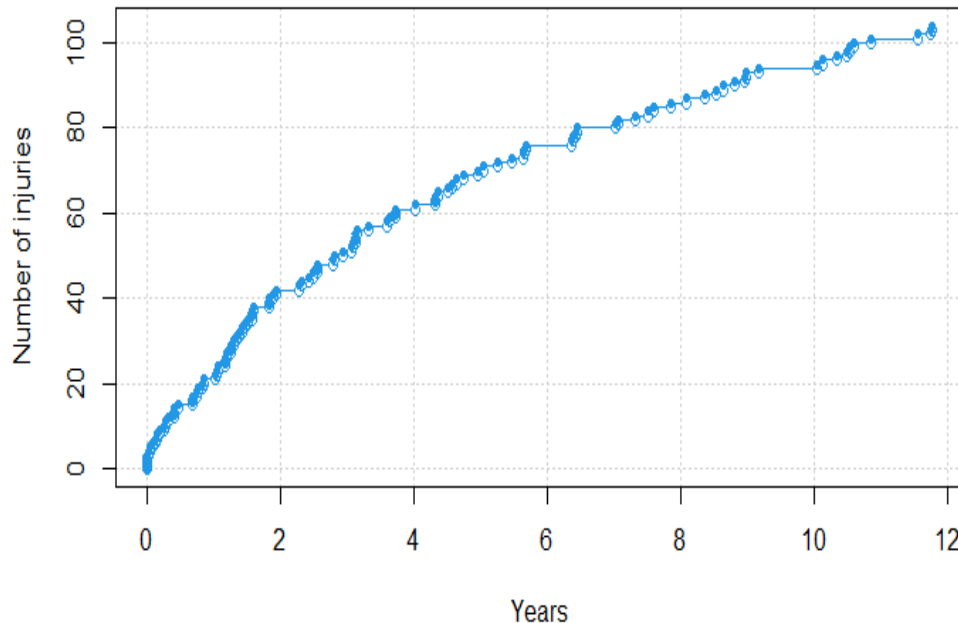
#computing event times
time<- round(t*sort(u)^2,4)

#plotting simulated trajectory
plot(time, N, type="n", xlab="Years", ylab="Number of injuries",
panel.first=grid())

segments(time[-length(time)], N[-length(time)], time[-1]-0.07,
N[-length(time)], col=4)

points(time, N, pch=20, col=4)
points(time[-1], N[-length(time)], pch=1, col=4)

```



EXERCISE 4.4. Below are the code and plot of the simulated trajectory. We are using the thinning simulation method where we uniformly dominate the intensity rate function by 20.

```

#specifying parameters
lambda<- function(s) 10+10*cos(2*pi*s)
lambda.star<- function(s) 20
Lambda.star<- function(s) 20*s

```

```

#specifying seed
set.seed(2866514)

#generating N(t)
njumps<- rpois(1, Lambda.star(10))

#generating N(t) standard uniforms
u<- c()
u[1]<- 0

for(i in 2:(njumps+1))
  u[i]<- runif(1)

#computing event times
time.star<- 10*sort(u)

#thinning event times
accepted<- c()
time<- c()
accepted[1]<- 1
time[1]<- 0

for (i in 2:(njumps+1)) {
  if (runif(1)<= lambda(time.star[i])/lambda.star(time.star[i]))
    accepted[i]=1 else accepted[i]=0
}

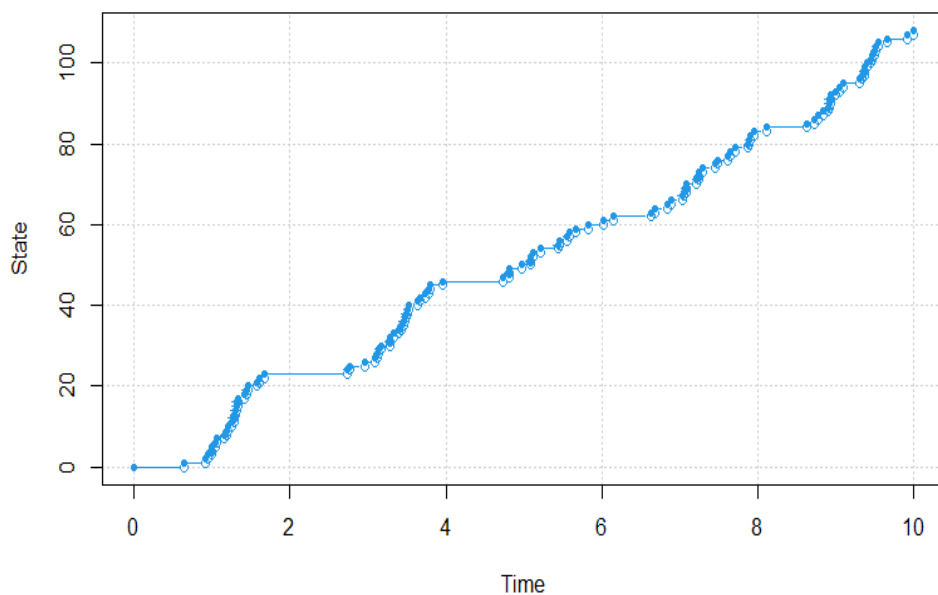
time<- time.star[-which(accepted==0)]
N<- 0:(length(time)-1)

#plotting trajectory
plot(time, N, type="n", xlab="Time", ylab="State", panel.first = grid())

segments(time[-length(time)],N[-length(time)], time[-1]-0.07,
          N[-length(time)], col=4)

points(time, N, ylim=c(0,120), pch=20, col=4)
points(time[-1],N[-length(time)],pch=1, col=4)

```



EXERCISE 4.5. In the process of radioactive decay, photons are emitted according to a nonhomogeneous Poisson process with the intensity rate $\lambda(t) = 100e^{-0.5t}$, $t \geq 0$. The integrated intensity rate function is $\Lambda(t) = \int_0^t \lambda(s)ds = 200(1 - e^{-0.5t})$, $t \geq 0$.

METHOD 1 (EXPONENTIAL INTERARRIVALS). To simulate event times, we solve the recurrence equations $1 - e^{-\Lambda(S_1)} = U_1$ and $1 - e^{-(\Lambda(S_{n+1}) - \Lambda(S_n))} = U_{n+1}$, $n \geq 1$. Equivalently, we can replace $1 - U_1$ by U_1 and solve $e^{-\Lambda(S_1)} = U_1$ and $e^{-(\Lambda(S_{n+1}) - \Lambda(S_n))} = U_{n+1}$, $n \geq 1$. In this example, the equations are: $e^{-200(1 - e^{-0.5S_1})} = U_1$ and $e^{-200(e^{-0.5S_n} - e^{-0.5S_{n+1}})} = U_{n+1}$, $n \geq 1$. Solving, we get

$$S_1 = -2\ln\left(1 + \left(\frac{1}{200}\right)\ln(U_1)\right) \text{ and } S_{n+1} = -2\ln\left(e^{-0.5S_n} + \left(\frac{1}{200}\right)\ln(U_{n+1})\right), n \geq 1.$$

The code below simulates a trajectory with 20 events.

```
#specifying parameters
njumps<- 20

#defining states
N<- 0:njumps

#defining times as vectors
time<- c()

#specifying seed
set.seed(40556002)

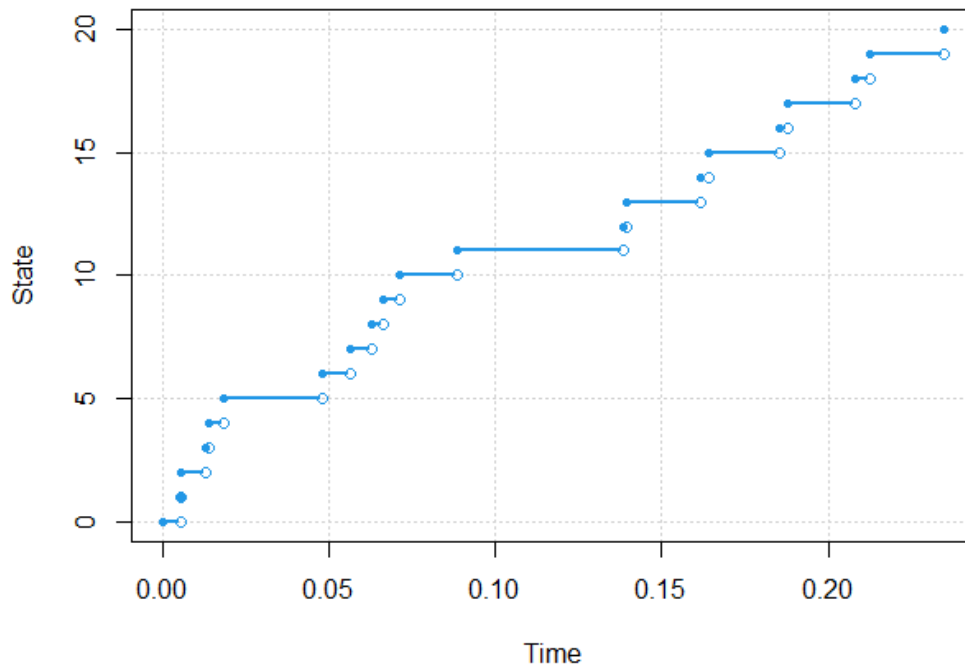
#generating standard uniforms
u<- c()
for(i in 1:njumps)
u[i]<- runif(1)

#computing event times
time[1]<- 0
time[2]<- -2*log(1+(1/200)*log(u[1]))

for(i in 3:(njumps+1)) {
time[i]<- -2*log(exp(-0.5*time[i-1])+(1/200)*log(u[i-1]))
}

#plotting trajectory
plot(time, N, type="n", xlab="Time", ylab="State", panel.first = grid())
segments(time[-length(time)],N[-length(time)], time[-1]-0.001, N[-length(time)],
lwd=2, col=4)

points(time, N, ylim=c(0,120), pch=20, col=4)
points(time[-1],N[-length(time)], pch=1, col=4)
```



METHOD 2 (UNIFORM ORDER STATISTICS). In this method, an event time S is found as the solution of the equation $\frac{\Lambda(S)}{\Lambda(0.25)} = U$ where by U we denote a standard uniform random variable from an ordered sample. The equation takes the form: $\frac{200(1-e^{-0.5S})}{200(1-e^{-(0.5)(0.25)})} = U$. The solution is $S = -2 \ln(1 - (1 - e^{-(0.5)(0.25)})U)$. The code and graphical output are presented below.

```
#specifying parameters
t<- 0.25
Lambda<- 200*(1-exp(-0.5*t))

#specifying seed
set.seed(492231)

#generating N(t)
njumps<- rpois(1,Lambda)

#defining states
N<- 0:njumps

#generating N(t) standard uniforms
u<- c()
u[1]<- 0

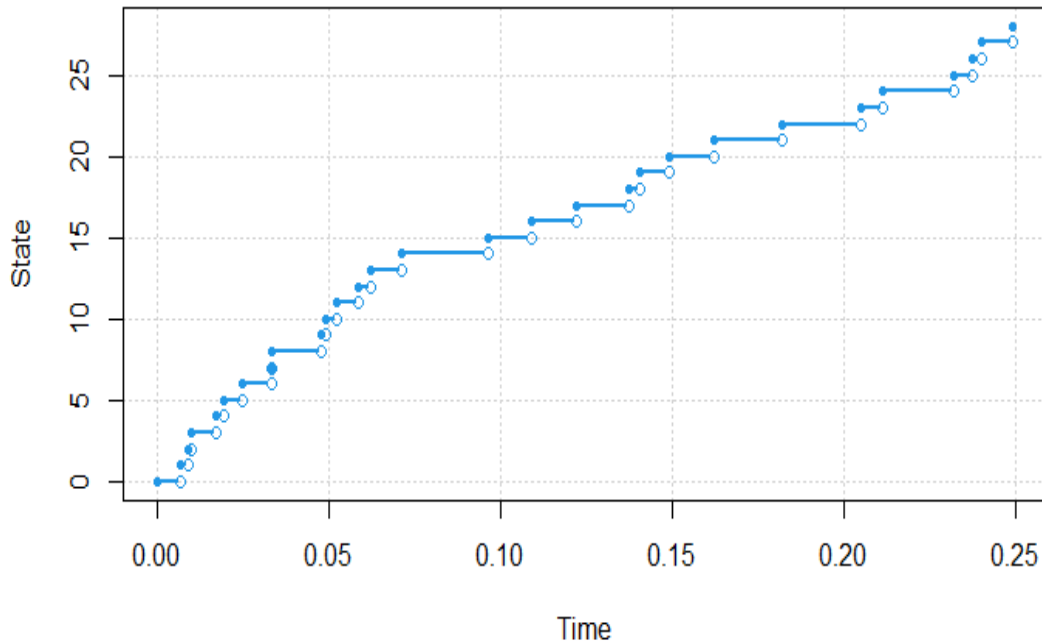
for(i in 2:(njumps+1))
  u[i]<- runif(1)

#computing event times
time<- -2*log(1-(1-exp(-0.5*t))*sort(u))

#plotting trajectory
plot(time, N, type="n", xlab="Time", ylab="State", panel.first = grid())
segments(time[-length(time)],N[-length(time)], time[-1]-0.001, N[-length(time)]],
lwd=2, col=4)

points(time, N, ylim=c(0,120), pch=20, col=4)
```

```
points(time[-1],N[-length(time)], pch=1, col=4)
```



METHOD 3 (THINNING). We bound the intensity rate function $\lambda(t) = 100e^{-0.5t}$, $t \geq 0$, uniformly by 100. The code below simulates event times of a homogeneous Poisson process and then applies the algorithm of the thinning method to select only those event times that belong to the nonhomogeneous Poisson process. The plot follows.

```
#specifying parameters
lambda<- function(s) 100*exp(-0.5*s)
lambda.star<- function(s) 100
Lambda.star<- function(s) 100*s

#specifying seed
set.seed(2866514)

#generating N(t)
njumps<- rpois(1, Lambda.star(0.25))

#generating N(t) standard uniforms
u<- c()
u[1]<- 0

for(i in 2:(njumps+1))
  u[i]<- runif(1)

#computing event times
time.star<- 0.25*sort(u)

#thinning event times
accepted<- c()
time<- c()
accepted[1]<- 1
time[1]<- 0

for (i in 2:(njumps+1)) {
  if (runif(1)<= lambda(time.star[i])/lambda.star(time.star[i]))
    accepted[i]=1 else accepted[i]=0
}
```



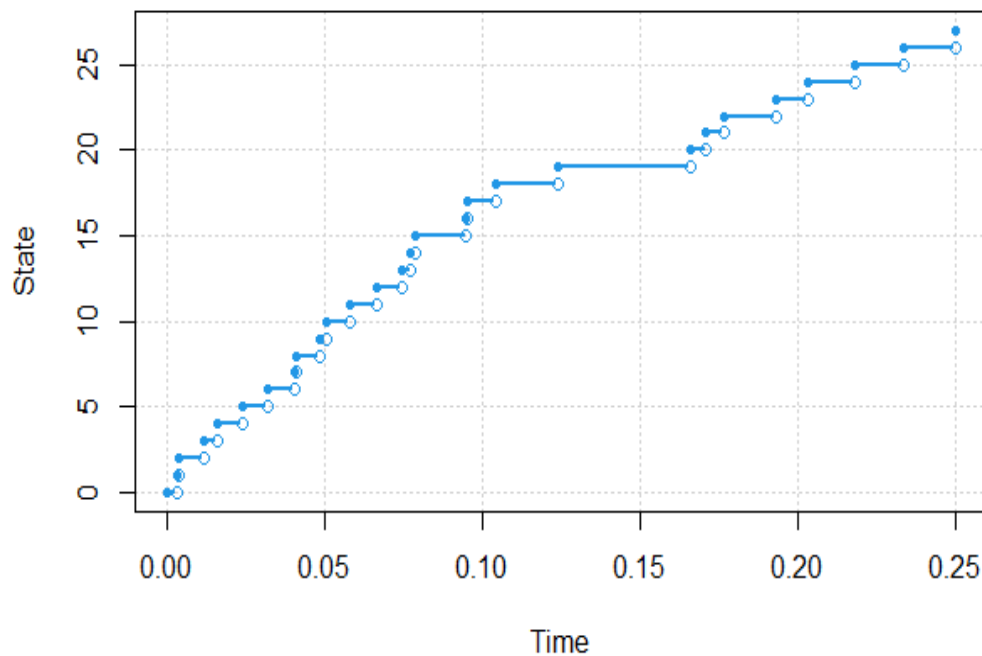
```

time<- time.star[-which(accepted==0)]
N<- 0:(length(time)-1)

#plotting trajectory
plot(time, N, type="n", xlab="Time", ylab="State", panel.first = grid())
segments(time[-length(time)],N[-length(time)], time[-1]-0.001, N[-length(time)],
lwd=2, col=4)

points(time, N, ylim=c(0,120), pch=20, col=4)
points(time[-1],N[-length(time)], pch=1, col=4)

```

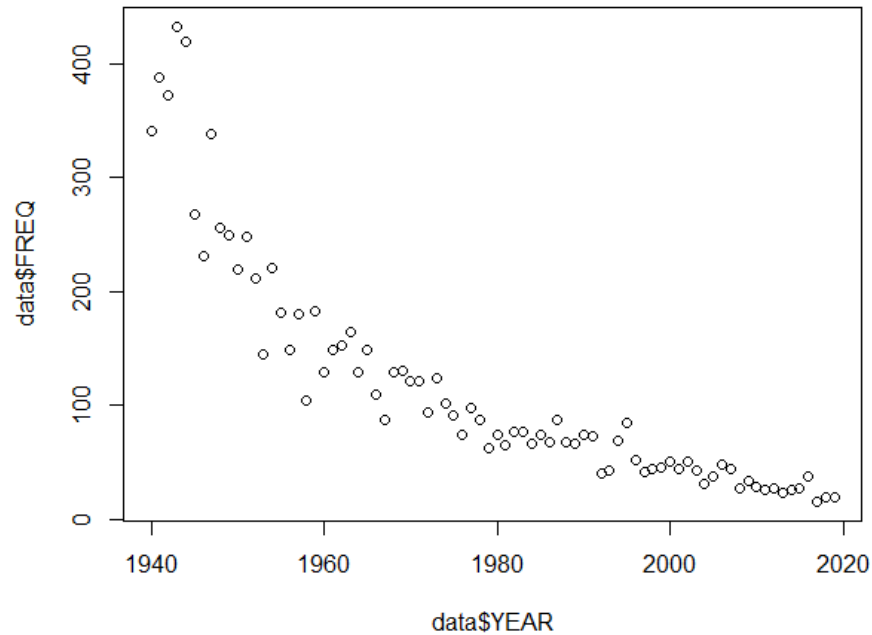


EXERCISE 4.6. (a) We run the following code to plot the counts of lightning deaths against year.

```

data<- read.csv(file="./lightningdata.csv", header=TRUE, sep=",")
plot(data$YEAR, data$FREQ)

```



We can see from the plot that the intensity rate decreases over time roughly exponentially. A possible explanation for it is that over the years more awareness has been created among citizens through educational efforts, so fewer people are exposed to the hazard.

(b) We can see that lightning strikes are essentially a seasonal phenomenon. The majority of them happen between May and September. It means that some interarrival times have very large values not inherent to an exponential distribution. Moreover, some incidents resulted in multiple fatalities which would be an event of probability zero under the Poisson law.

EXERCISE 4.7. (a) The code and output below estimate the parameters of the model using the regression approach.

```
port.data<- read.csv(file="./Exercise4.4Data.csv", header=TRUE, sep=",")

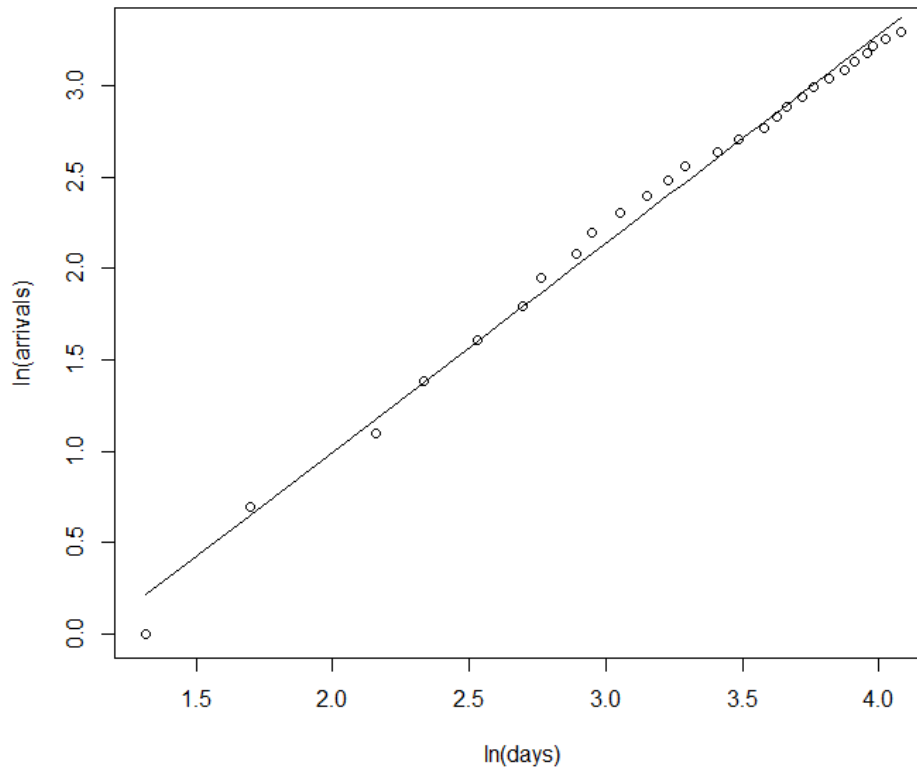
x<- log(port.data$days)
y<- log(port.data$arrivals)

glm(y~x)

plot(x,y, xlab="ln(days)", ylab="ln(arrivals)")

Coefficients:
(Intercept)          x
      -1.294         1.145

lines(x, -1.294+1.145*x)
```



The estimates of the model parameters are $\hat{\alpha} = e^{-1.294} = 0.274172$, and $\hat{\beta} = 1.145$.

(b) The code that follows estimates the parameters using the maximum likelihood approach.

```
port.data<- read.csv(file="./Exercise4.4Data.csv", header=TRUE, sep=",")

x<- log(port.data$days)
y<- log(port.data$arrivals)
N<- 27

print(beta.hat<- N/(N*x[N]-sum(x)))

1.162782

print(alpha.hat<- N/exp(x[N]*beta.hat))

0.2351766
```

The MLEs are $\hat{\alpha} = 0.2351766$, and $\hat{\beta} = 1.162782$.

(c) To predict when the next 10,000 TEUs arrive at the port, we submit the following lines of code.

```

port.data<- read.csv(file="./Exercise4.4Data.csv", header=TRUE, sep=",")

x<- log(port.data$days)
y<- log(port.data$arrivals)
N<- 27

alpha.hat<- c(0.274172, 0.2351766)
beta.hat<- c(1.145, 1.162782)
S.hat<- c()

library(pracma)

for(i in 1:2)
print(S.hat[i]<- alpha.hat[i]^(-
1/beta.hat[i])*exp(alpha.hat[i]*exp(x[N])^beta.hat[i])*
gammainc(alpha.hat[i]*exp(x[N])^beta.hat[i], 1/beta.hat[i]+1)[2])

```

60.85581
60.97308

According to the data, the 27th arrival was on day 59.1. The 28th arrival is predicted to be on day 60.85581 (by the linear regression), or 60.97308 (by the maximum likelihood).

CHAPTER 5

EXERCISE 5.1. (a) Let $X(t) = \sum_{i=1}^{N(t)} Y_i$ be the total amount paid in prizes up to time t hours. We know that it is a compound Poisson process with $N(t) \sim \text{Poisson}(1.5t)$, and Y_i independent of each other and $N(t)$. The first two moments of Y_1 are $E(Y_1) = (\$5000)(0.15) + (2000)(0.35) + (\$500)(0.2) + (\$100)(0.3) = \$1,580$, and $E(Y_1^2) = (\$5000)^2(0.15) + (2000)^2(0.35) + (\$500)^2(0.2) + (\$100)^2(0.3) = \25,203,000 .

Therefore, the mean of $X(200)$ is $E(X(200)) = (1.5)(200)(\$1,580) = \$474,000$. The variance is $\text{Var}(X(200)) = (1.5)(200)(\$^25,203,000) = \21,560,900,000 , and the standard deviation is

$$\sqrt{\text{Var}(X(200))} = \sqrt{\$^21,560,900,000} = \$39,508.23.$$

The budget for 100 games should be $E(X(200)) + \sqrt{\text{Var}(X(200))} = \$474,000 + \$39,508.23 = \$513,508.23$.

(b) Below are the codes, all relevant output, and the graph for the simulated 100 games.

```
#specifying parameters
lambda<- 1.5
total.hours<- 200
amount<- c(5000, 2000, 500, 100)
p<- c(0.15, 0.35, 0.2, 0.3)

#specifying seed
set.seed(704661)

#generating number of prizes
nprizes<- rpois(1,lambda*total.hours)

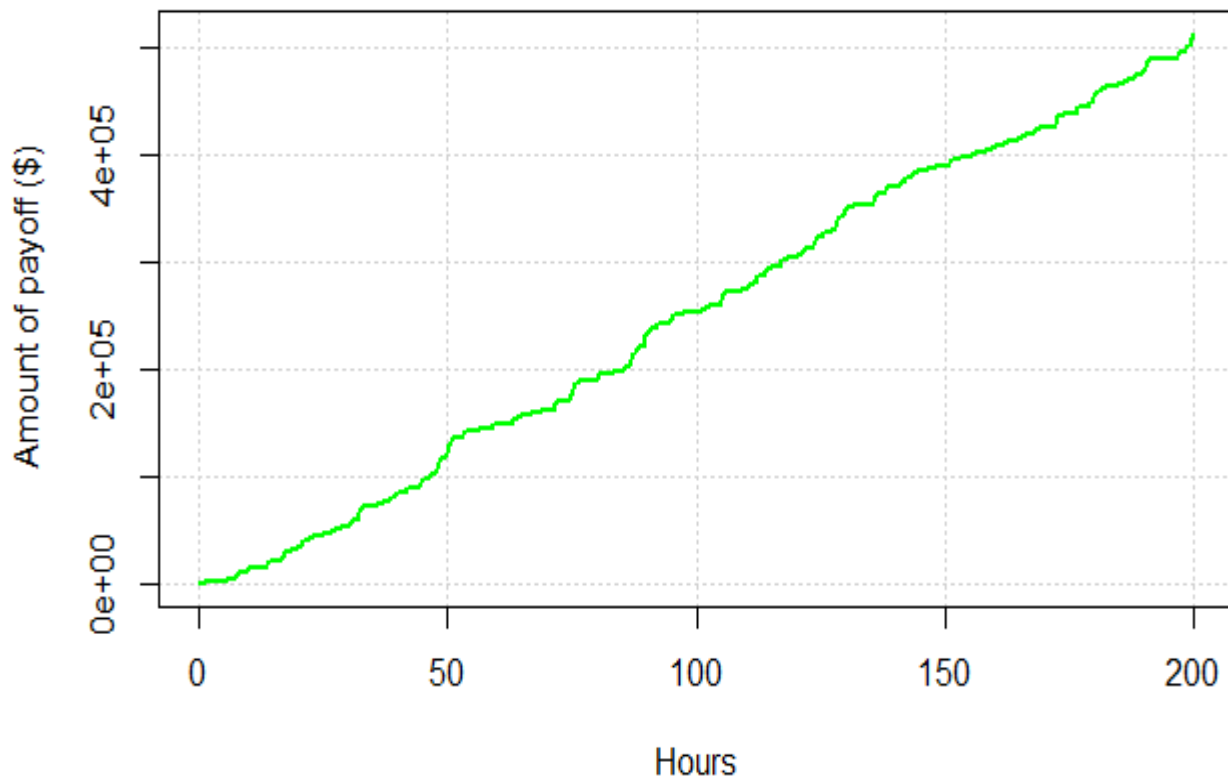
#defining vectors
payoff<- c()
time<- c()
u<- c()

#setting initial values
payoff[1]<- 0
u[1]<- 0

#generating standard uniforms
for(i in 2:(nprizes+1)) {
  u[i]<- runif(1)
  payoff[i]<- payoff[i-1] + amount[sample(1:4, 1, prob=p)]
}

#computing event times
hour<- total.hours*sort(u)

#simulating trajectory
plot(hour, payoff, type="l", lty=1, lwd=2, col="green", xlab="Hours", ylab="Amount
of payoff ($)", panel.first = grid())
```



```
nprizes
```

```
308
```

```
payoff[length(payoff)]
```

```
557100
```

There were a total of 308 prizes given out during the 200 hours of the 100 games. The total payoff was \$557,100.00. Since the budget that the producer had for the 100 games was \$513,508.23, the producer ran out of money before the 100th game.

```
payoff
```

```
[289] 511200 513200 513300 515300 515400
```

The producer ran out of budget after the 291st prize was given out.

```
hour[291]
```

```
190.8272
```

The 291st prize was given out at hour 190.8272 of the show, that is, during the first half of the 96th show.

EXERCISE 5.2. (a) Let $Y_i \sim \text{Unif}(\$30, \$300)$ be the i th claim amount. The two first moments of Y_1 are $E(Y_1) = \frac{\$30 + \$300}{2} = \$165$, and $E(Y_1^2) = \frac{1}{\$300 - \$30} \int_{\$30}^{\$300} u^2 du = \frac{(\$300)^3 - (\$30)^3}{3(\$270)} = \$^2 33,300$.

Let $X(t) = \sum_{i=1}^{N(t)} Y_i$ be the aggregate claim process. Its mean for $t = 30$ is $E(X(30)) = (60)(30)(\$165) = \$297,000$, and the standard deviation is $\sqrt{Var(X(30))} = \sqrt{(60)(30)(\$^2 33,300)} = \$7,742.093$.

(b) By the Central Limit Theorem, $Z = \frac{X(30) - E(X(30))}{\sqrt{Var(X(30))}} = \frac{X(30) - 297000}{7742.093}$ has approximately a $N(0,1)$ distribution, and therefore, $P(X(30) > 300000) = P\left(Z > \frac{300000 - 297000}{7742.093}\right) = P(Z > 0.387492) = 0.349196$.

EXERCISE 5.3. (a) Let $X(t) = \sum_{i=1}^{N(t)} Y_i$ denote the aggregate number of light photons that are generated up to t seconds. We are given that $N(t) \sim \text{Poisson}(\lambda t)$ and $Y_i \sim \text{Poisson}(\tilde{\lambda})$. The mean and standard deviation of $X(t)$ are $E(X(t)) = (\lambda)(t)(\tilde{\lambda})$ and $\sqrt{Var(X(t))} = \sqrt{(\lambda)(t)(EY_1^2)} = \sqrt{(\lambda)(t)(\tilde{\lambda} + \tilde{\lambda}^2)}$.

(b) The code below simulates 100 values of the aggregate number of light photons generated within 10 seconds.

```
#specifying parameters
total.time<- 10
lambda<- 50
lambda.tilde<- 5

total.photons<- c()

for (j in 1:100) {

  nphotons<- c()
  nphotons[1]<- 0

  #generating N(t)
  set.seed(150*j)
  N<- rpois(1,lambda*total.time)

  #simulating trajectory
  for (i in 2:N)
    nphotons[i]<- nphotons[i-1]+rpois(1,lambda.tilde)

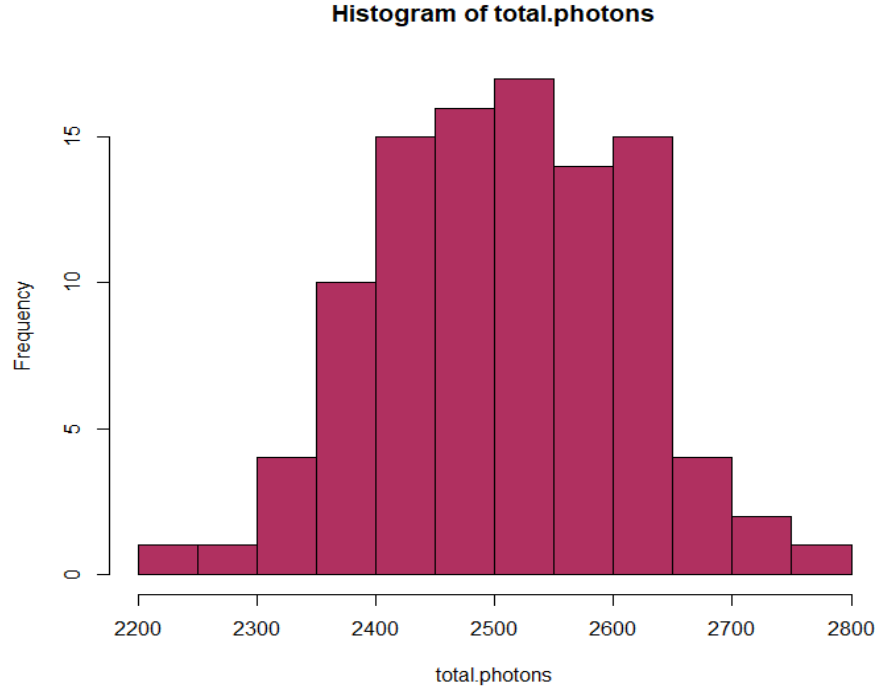
  total.photons[j]<- nphotons[N]
}
```

```
total.photons
[1] 2333 2737 2532 2392 2612 2484 2474 2440
[9] 2638 2549 2549 2361 2459 2603 2633 2537
[17] 2379 2314 2496 2630 2410 2233 2350 2610
[25] 2506 2412 2444 2444 2572 2430 2477 2472
[33] 2538 2470 2648 2641 2333 2425 2475 2549
[41] 2382 2599 2451 2386 2477 2262 2673 2627
[49] 2493 2490 2789 2449 2572 2537 2566 2427
[57] 2591 2469 2414 2508 2610 2587 2523 2384
[65] 2695 2578 2633 2523 2355 2597 2577 2558
[73] 2550 2396 2406 2496 2576 2627 2434 2552
```

```
[81] 2425 2393 2504 2664 2485 2408 2625 2500
[89] 2595 2609 2521 2524 2420 2569 2659 2741
[97] 2360 2617 2546 2505
```

(c) Below we construct a histogram for these 100 values.

```
hist(total.photons, col="maroon")
```



The histogram does resemble a bell shape. It should be the case since λ is large, and so, the Central Limit Theorem should be applicable.

EXERCISE 5.4. The average present value of the total claim amount is computed by conditioning on the value of $N(t)$. We proceed as follows: $E[P(t)] = EE[P(t)|N(t)] = EE\left[\sum_{i=1}^{N(t)} X_i e^{-\delta S_i} \mid N(t)\right] = E\left[\sum_{i=1}^{N(t)} E(X_i)E(e^{-\delta S_i})\right]$. The claim arrival times $S_1, \dots, S_{N(t)}$ are order statistics from a uniform distribution on $[0, t]$ and we can write $S_i = tU_{(i)}$ where $U_{(i)}$ is the i th order statistic from the standard uniform distribution. We continue $E[P(t)] = E\left[\sum_{i=1}^{N(t)} E(X_i)E(e^{-\delta t U_{(i)}})\right] = E(X_1)E\left[\sum_{i=1}^{N(t)} E(e^{-\delta t U_i})\right] = E(X_1)E(N(t))E(e^{-\delta t U_1}) = E(X_1)\lambda \int_0^1 e^{-\delta t u} du = E(X_1)\left(\frac{\lambda}{\delta}\right)(1 - e^{-\delta t})$.

EXERCISE 5.5. (a) Random variables Y_i 's are iid $\sim \text{Poisson}(\beta)$. Therefore, $\sum_{i=1}^n Y_i \sim \text{Poisson}(\beta n)$. We write $P(X(t) = x) = P(\sum_{i=1}^{N(t)} Y_i = x) = \sum_{n=0}^{\infty} P(\sum_{i=1}^n Y_i = x \mid N(t) = n)P(N(t) = n) = \sum_{n=0}^{\infty} \frac{(\beta n)^x}{x!} e^{-\beta n} \frac{(\lambda t)^n}{n!} e^{-\lambda t} = \frac{\beta^x}{x!} e^{-\lambda t} \sum_{n=0}^{\infty} \frac{n^x (\lambda t)^n}{n!} e^{-\beta n}$.

(b) $E(X(t)) = \lambda t E(Y_1) = \lambda t \beta$, $\text{Var}(X(t)) = \lambda t E Y_1^2 = \lambda t (\beta + \beta^2)$.

$$(c) P(X(t) = 0) = \frac{\beta^0}{0!} e^{-\lambda t} \sum_{n=0}^{\infty} \frac{n^0 (\lambda t)^n}{n!} e^{-\beta n} = e^{-\lambda t} \sum_{n=0}^{\infty} \frac{(\lambda t e^{-\beta})^n}{n!} = e^{-\lambda t + \lambda t e^{-\beta}} = e^{-\lambda t(1 - e^{-\beta})}.$$

The ratio between the variance and mean is $\frac{Var(X(t))}{E(X(t))} = 1 + \beta$, thus, β can be estimated as

$$\hat{\beta} = \frac{\hat{Var}(X(t))}{\hat{E}(X(t))} - 1. \text{ Also, } \ln P(X(t) = 0) = \ln P(0) = -\lambda t(1 - e^{-\beta}). \text{ Hence, we can estimate } \lambda \text{ by}$$

$$\hat{\lambda} = -\frac{\ln \hat{P}(0)}{t(1 - e^{-\hat{\beta}})}.$$

EXERCISE 5.6. (a) The total dollar amount can be modeled by a compound Poisson process $X(t) = \sum_{i=1}^{N(t)} Y_i$ where $N(t)$ is the process governing the number of cars that come to the gas station up to time t , and Y_i is the dollar amount that the i th car driver pays. We are given that $N(t) \sim \text{Poisson}(\lambda t)$, and $Y_i \sim \text{Gamma}(\alpha, \beta)$ with mean $E(Y_i) = \alpha\beta$, and variance $Var(Y_i) = \alpha\beta^2$.

(b) Denote by $T_i \sim \text{Exp}(mean = \frac{1}{\lambda})$ the interarrival times between car arrivals. The method of moments estimator of λ is $\hat{\lambda} = 1/\bar{T}$, the reciprocal of the sample mean of the interarrival times. The method of moments estimators of α and β are $\hat{\alpha} = \frac{n\bar{Y}^2}{\sum_{i=1}^n Y_i^2 - n\bar{Y}^2}$ and $\hat{\beta} = \frac{\bar{Y}}{\hat{\alpha}} = \frac{\sum_{i=1}^n Y_i^2 - n\bar{Y}^2}{n\bar{Y}}$. They solve the system of two equations: $\bar{Y} = \hat{E}(Y_1) = \hat{\alpha}\hat{\beta}$ and $\frac{1}{n}\sum_{i=1}^n Y_i^2 = \hat{E}(Y_1^2) = \hat{Var}(Y_1) + (\hat{E}(Y_1))^2 = \hat{\alpha}\hat{\beta}^2 + (\hat{\alpha}\hat{\beta})^2 = \hat{\beta}\bar{Y} + \bar{Y}^2$.

(a) The code below produces numeric values of the estimators and plots histograms with fitted curves.

```
gas.data<- read.csv(file="./Exercise5.6Data.csv", header=TRUE, sep=",")

#computing lag
gas.data$ArrivalTime.lag<- c(0,head(gas.data$ArrivalTime, -1))
#gas.data<-gas.data[-1,] #removing first row

#computing interarrival times
interarrival.time<- gas.data$ArrivalTime-gas.data$ArrivalTime.lag

#estimating lambda of Poisson arrival
print(lambda.hat<- 1/mean(interarrival.time))

0.6029832

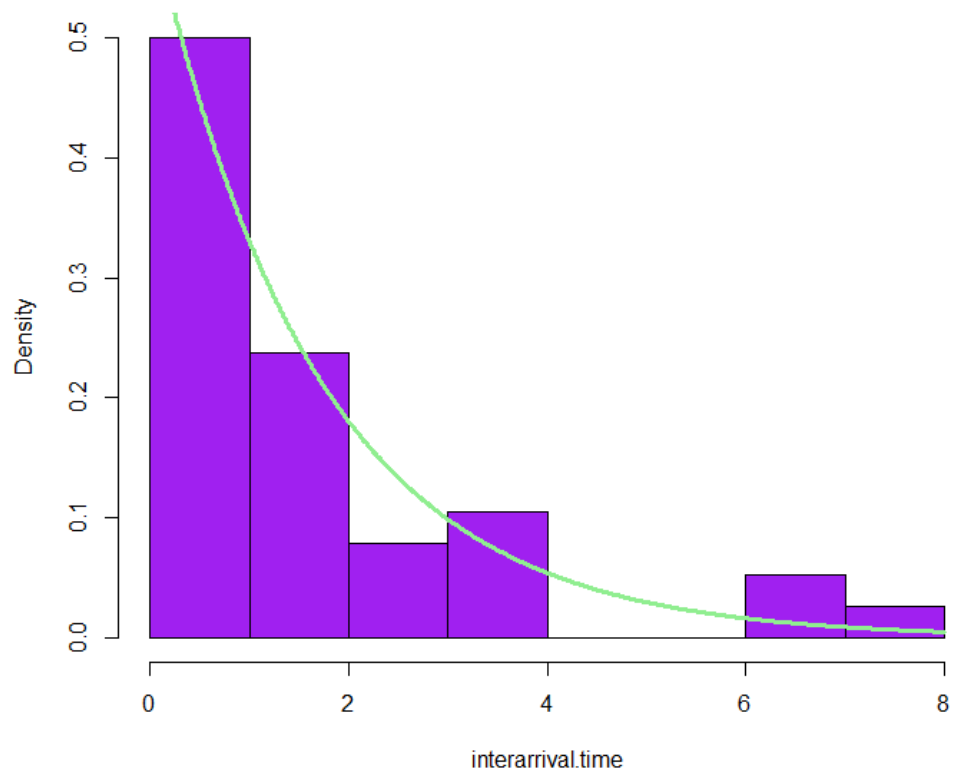
print(1/lambda.hat)
```

1.658421

There are, on average, 0.6029832 car arrivals every minute. The average wait time between two arrivals is 1.658421 minutes.

```
#overlaying histogram and fitted exponential density curve
hist(interarrival.time,freq=FALSE, col="purple")
x<- seq(0, 8, by=0.01)
y<- dexp(x,lambda.hat)
lines(x, y, lty=1, col="light green", lwd=3)
```

Histogram of interarrival.time



```
#estimating parameters of gamma distribution
amount<- gas.data$AmountSpent
print(alpha.hat<- length(amount)*mean(amount)^2/(sum(amount^2)-
length(amount)*mean(amount)^2))
```

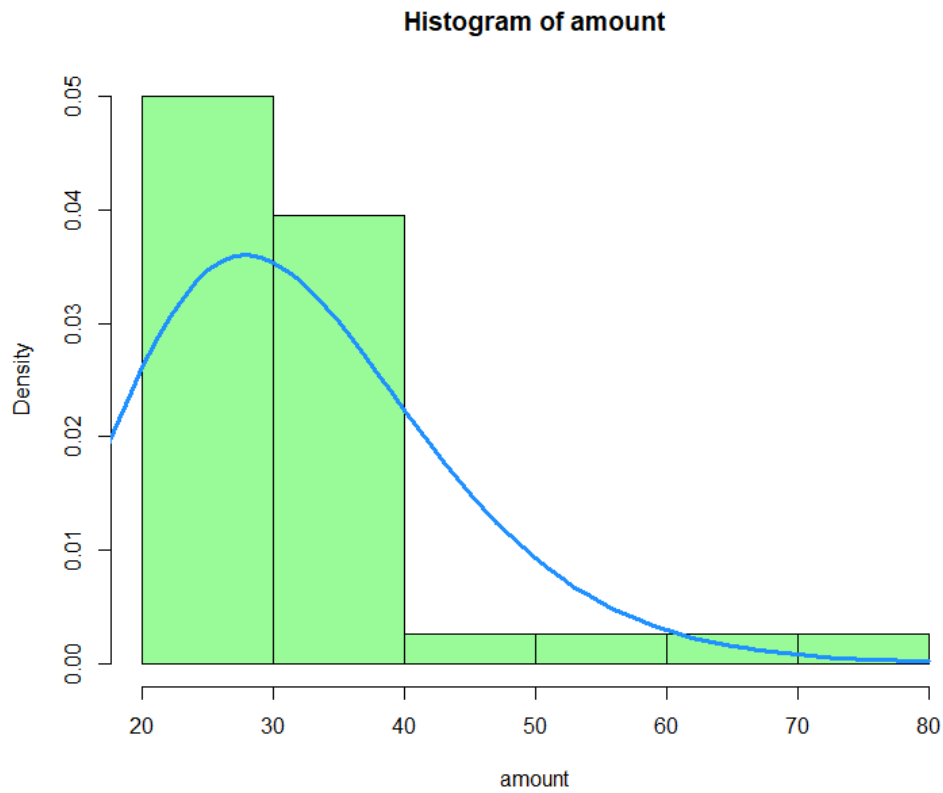
7.48998

```
print(beta.hat<- mean(amount)/alpha.hat)
```

4.29609

```
#overlaying histogram and fitted density
hist(amount, freq=FALSE, col = "pale green")

x<- 0:80
y<- dgamma(x, alpha.hat, 1/beta.hat)
lines(x, y, lty=1, col="dodger blue", lwd=3)
```



(d) The estimated mean of the total dollar amount at one hour is $\hat{E}(X(60)) = (\hat{\lambda})(t)(\hat{\alpha})(\hat{\beta}) = (0.6029832)(60)(7.48998)(4.29609) = \$1,164.154$. The estimated standard deviation is

$$\sqrt{\hat{Var}(X(60))} = \sqrt{(\hat{\lambda})(t)(\hat{\alpha})(\hat{\beta})^2} = \sqrt{(0.6029832)(60)(7.48998)(4.29609)^2} = \$70.71995.$$

CHAPTER 6

EXERCISE 6.1. (a) $Cov(N(s), N(t) - N(s)) = E[N(s)N(t-s)] - E[N(s)]E[N(t-s)] = EE[N(s)N(t-s) | \Lambda] - EE[N(s) | \Lambda] \cdot EE[N(t-s) | \Lambda] = E[(\Lambda s)(\Lambda)(t-s)] - E[\Lambda s]E[\Lambda(t-s)] = s(t-s)E(\Lambda^2) - s(t-s)(E(\Lambda))^2 = s(t-s)Var(\Lambda).$

(b) $Cov(N(s), N(t)) = E[N(s)N(t)] - E[N(s)]E[N(t)] = EE[N(s)(N(t) - N(s) + N(s)) | \Lambda] - E[N(s)]E[N(t)] = EE[N(s)N(t-s) | \Lambda] + EE[(N(s))^2 | \Lambda] - EE[N(s) | \Lambda] \cdot EE[N(t) | \Lambda] = E[(\Lambda s)(\Lambda)(t-s)] + E[\Lambda s + (\Lambda s)^2] - E(\Lambda s)E(\Lambda t) = s(t-s)E(\Lambda^2) + sE(\Lambda) + s^2E(\Lambda^2) - st(E(\Lambda))^2 = stE(\Lambda^2) - st(E(\Lambda))^2 + sE(\Lambda) = stVar(\Lambda) + sE(\Lambda).$

EXERCISE 6.2. (a) $F_{\Lambda|N(t)}(\lambda|n) = P(\Lambda \leq \lambda | N(t) = n) = \frac{P(N(t)=n, \Lambda \leq \lambda)}{P(N(t)=n)}$

$$= \frac{\int_0^\lambda P(N(t) = n | \Lambda = u) f_\Lambda(u) du}{\int_0^\infty P(N(t) = n | \Lambda = u) f_\Lambda(u) du} = \frac{\int_0^\lambda \frac{(ut)^n}{n!} e^{-ut} f_\Lambda(u) du}{\int_0^\infty \frac{(ut)^n}{n!} e^{-ut} f_\Lambda(u) du} = \frac{\int_0^\lambda u^n e^{-ut} f_\Lambda(u) du}{\int_0^\infty u^n e^{-ut} f_\Lambda(u) du}.$$

$$(b) f_{\Lambda|N(t)}(\lambda|n) = F'_{\Lambda|N(t)}(\lambda|n) = \frac{\lambda^n e^{-\lambda t} f_\Lambda(\lambda)}{\int_0^\infty \lambda^n e^{-\lambda t} f_\Lambda(\lambda) d\lambda}.$$

$$(c) E[\Lambda | N(t) = n] = \int_0^\infty \lambda f_{\Lambda|N(t)}(\lambda|n) d\lambda = \frac{\int_0^\infty \lambda^{n+1} e^{-\lambda t} f_\Lambda(\lambda) d\lambda}{\int_0^\infty \lambda^n e^{-\lambda t} f_\Lambda(\lambda) d\lambda}.$$

EXERCISE 6.3. (a) Denote by $\{N(t), t \geq 0\}$ the process of visitor arrival. We know that $N(t) \sim \text{Poisson}(\Lambda t)$ where $P(\Lambda = 4) = 0.46, P(\Lambda = 2) = 0.24$, and $P(\Lambda = 3) = 0.30$. The mean and variance of $N(t)$ are $E(N(t)) = tE(\Lambda) = t((4)(0.46) + (2)(0.24) + (3)(0.30)) = 3.22t$, and $Var(N(t)) = t^2Var(\Lambda) + tE(\Lambda) = t^2((4)^2(0.46) + (2)^2(0.24) + (3)^2(0.30) - (3.22)^2) + 3.22t = 0.6516t^2 + 3.22t$.

(b) The code below simulates 5 trajectories of the process with 200 visitors each.

```
#specifying parameters
p<- c(0.46, 0.24, 0.30)
lambda<- c(4, 2, 3)
nvisitors<- 200
time<- data.frame()
N<- data.frame()

#specifying seed
set.seed(109088)

#creating loop to simulate trajectories
for(j in 1:5) {

#selecting rate
Lambda<- lambda[sample(1:3, 1, prob=p)]
```

```

#setting initial values
time[1,j]<- 0
N[1,j]<- 0

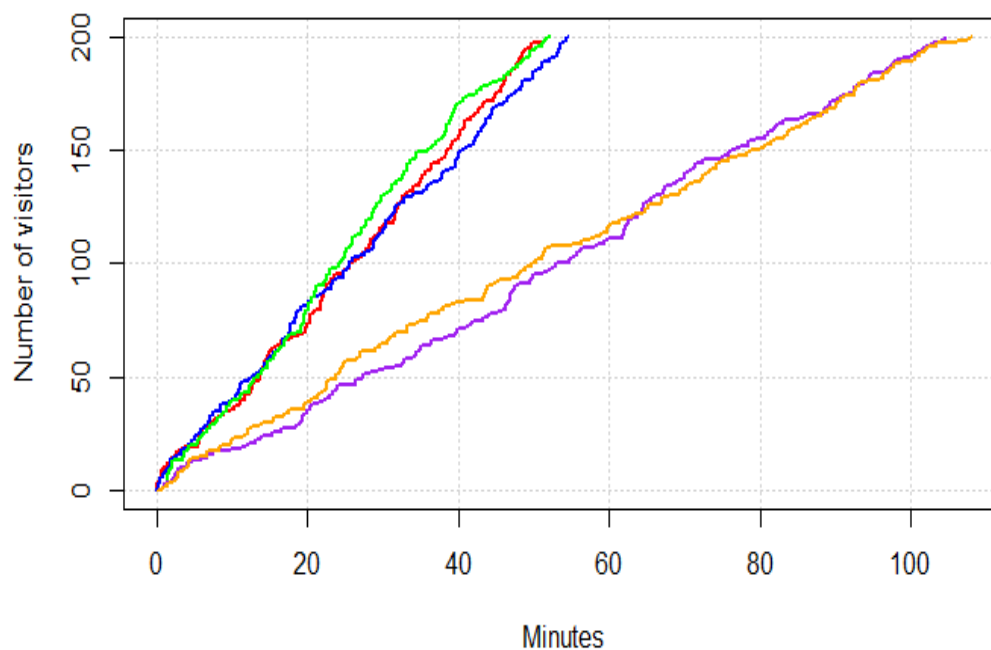
#simulating trajectory
i<- 2

repeat {
time[i,j]<- time[i-1,j]+round((-1/Lambda)*log(1-runif(1)),3)-0.001
  N[i,j]<- N[i-1,j]

if(i==2*nvisitors+2) break
else {
  time[i+1,j]<- time[i,j]+0.001
  N[i+1,j]<- N[i,j]+1
  i<- i+2
}
}

#plotting trajectories
matplot(time, N, type="l", lty=1, lwd=2, col=c("red", "blue", "green",
"purple", "orange"), xlab="Minutes", ylab="Number of visitors",
panel.first=grid())

```



(c) The code below simulates 5 trajectories of the process that depict arrivals within one hour.

```

#specifying parameters
t<- 60
p<- c(0.46, 0.24, 0.30)
lambda<- c(4, 2, 3)
time<- data.frame()
N<- data.frame()

#specifying seed

```

```

set.seed(5055562)

#creating loop to simulate trajectories
for(j in 1:5) {

#selecting rate
Lambda<- lambda[sample(1:3, 1, prob=p)]

#setting initial values
time[1,j]<- 0
N[1,j]<- 0

#generating N(t)
N.total<- rpois(1,Lambda*t)

#generating N(t) standard uniforms
u<- 1:N.total
for(i in 1:N.total)
  u[i]<- runif(1)

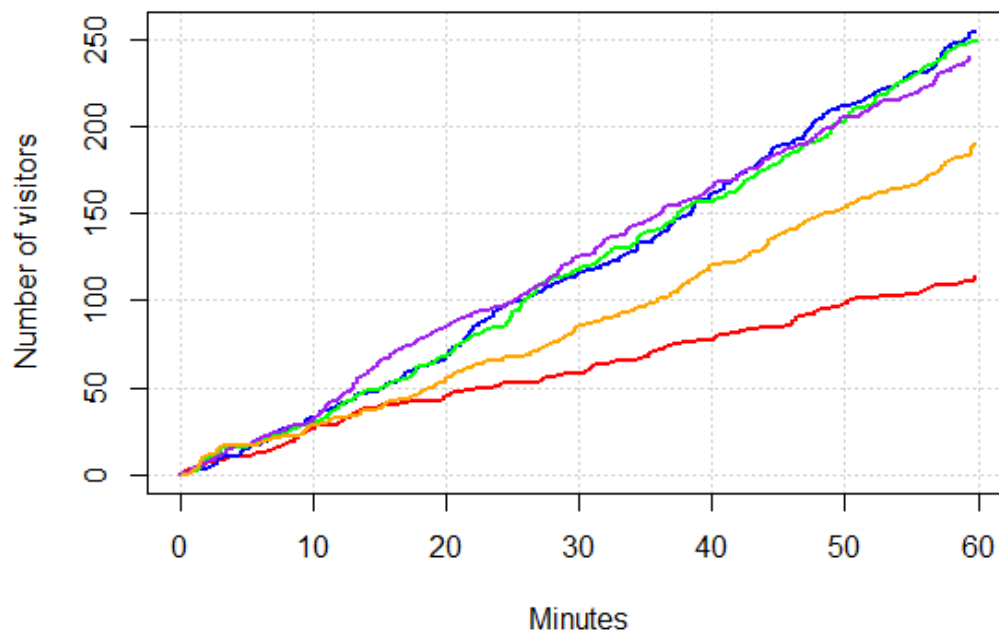
#sorting standard uniforms
u.sorted<- sort(u)

#computing N(t) event times
s<- t*u.sorted

#generating jumps
for (i in seq(2, 2*N.total, 2)) {
  time[i,j]<- s[i/2]-0.001
  time[i+1,j]<- s[i/2]
  N[i,j]<- N[i-1,j]
  N[i+1,j]<- N[i-1,j]+1
}
}

#plotting simulated trajectories
matplot(time, N, type="l", lty=1, lwd=2, col=c("red", "blue", "green",
"purple", "orange"), xlab="Minutes", ylab="Number of visitors",
panel.first=grid())

```



EXERCISE 6.4. (a) Let $N(t)$ denote the number of defaults by time t . It is given that $N(t) \sim \text{Poisson}(\Lambda t)$ where $\Lambda \sim \text{Uniform}(0, 2)$. The average number of defaults within 5 years is $E(N(5)) = tE(\Lambda) = (5)(1) = 5$.

(b) The variance of the number of defaults within 5 years is $\text{Var}(N(5)) = t^2 \text{Var}(\Lambda) + tE(\Lambda) = (5)^2 \left(\frac{2^2}{12}\right) + (5)(1) = 13.3333$.

(c) As shown in Exercise 6.1(a), $\text{Cov}(N(s), N(t) - N(s)) = s(t - s)\text{Var}(\Lambda)$. For $s = 3$, and $t = 5$, $\text{Cov}(N(3), N(5) - N(3)) = (3)(5 - 3)(1/3) = 2$.

(d) As shown in Exercise 1(b), $\text{Cov}(N(s), N(t)) = st\text{Var}(\Lambda) + sE(\Lambda)$. For $s = 3$, and $t = 5$, $\text{Cov}(N(3), N(5)) = (3)(5)(1/3) + (3)(1) = 8$.

(e) Using the result proven in Exercise 6.2, we get $P(\Lambda < 0.5 \mid N(5) = 2) = \frac{\int_0^{0.5} u^2 e^{-5u} f_\Lambda(u) du}{\int_0^\infty u^2 e^{-5u} f_\Lambda(u) du} = \frac{\int_0^{0.5} u^2 e^{-5u} du}{\int_0^\infty u^2 e^{-5u} du}$. Now, $\int u^2 e^{-5u} du = -\frac{1}{5}u^2 e^{-5u} - \frac{2}{25}u e^{-5u} - \frac{2}{125}e^{-5u}$. Therefore,

$$P(\Lambda < 0.5 \mid N(5) = 2) = \frac{\int_0^{0.5} u^2 e^{-5u} du}{\int_0^\infty u^2 e^{-5u} du} = \frac{-\frac{1}{5}u^2 e^{-5u} - \frac{2}{25}u e^{-5u} - \frac{2}{125}e^{-5u} \Big|_0^{0.5}}{-\frac{1}{5}u^2 e^{-5u} - \frac{2}{25}u e^{-5u} - \frac{2}{125}e^{-5u} \Big|_0^\infty} = \frac{-e^{-2.5} \left(\frac{1}{20} + \frac{1}{25} + \frac{2}{125} \right) + \frac{2}{125}}{\frac{2}{125}} = 0.456187.$$

EXERCISE 6.5. (a) Let $N(t)$ denote the amount of SWE accumulated within time t . It is given that $N(t) \sim \text{Poisson}(\Lambda t)$ where $\Lambda \sim \text{Poisson}(\lambda = 24.3)$. The average and standard deviation of SWE for one year are $E(N(1)) = tE(\Lambda) = \lambda t = (24.3)(1) = 24.3$ inches, and $\sqrt{\text{Var}(N(1))} = \sqrt{t^2 \text{Var}(\Lambda) + tE(\Lambda)} = \sqrt{t^2 \lambda + t\lambda} = \sqrt{(1^2 + 1)(24.3)} = \sqrt{48.6} = 6.97$ inches. For five years, $E(N(5)) = tE(\Lambda) = \lambda t = (24.3)(5) = 121.5$ inches and $\sqrt{\text{Var}(N(5))} = \sqrt{(5^2 + 5)(24.3)} = \sqrt{729} = 27$ inches.

(b) The code below simulates 5 trajectories that reach 140 inches of SWE each.

```
#specifying parameters
lambda<- 24.3
SWE.inches<- 140
time<- data.frame()
N<- data.frame()

#specifying seed
set.seed(9000004)

#creating loop to simulate trajectories
for(j in 1:5) {

#selecting rate
Lambda<- rpois(1,lambda)
```

```

#setting initial values
time[1,j]<- 0
N[1,j]<- 0

#simulating trajectory
i<- 2

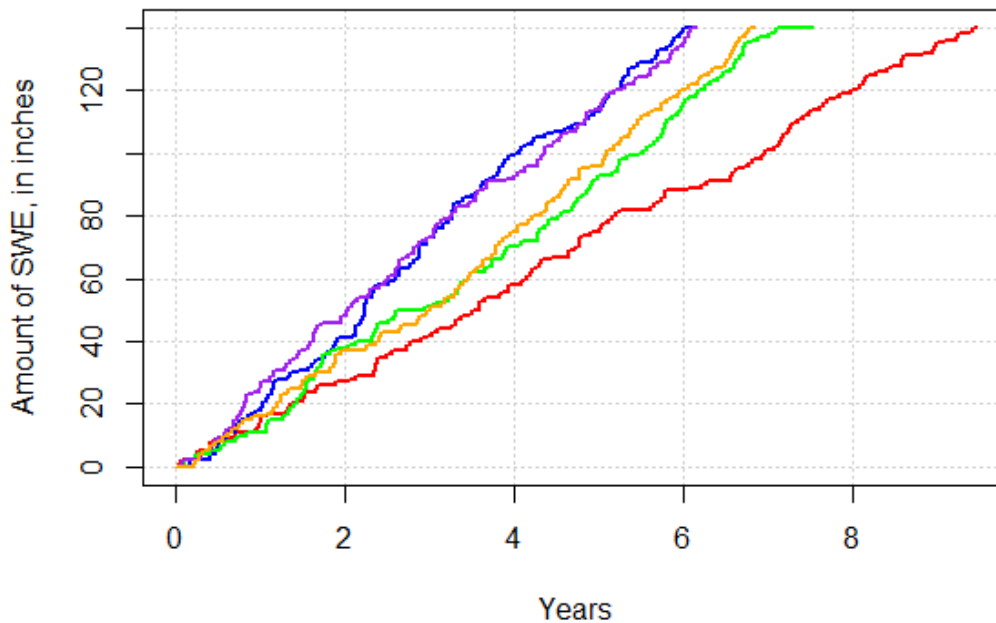
repeat {
time[i,j]<- time[i-1,j]+round((-1/Lambda)*log(1-runif(1)),3)-0.001
  N[i,j]<- N[i-1,j]

if(i==2*SWE.inches+2) break
else {
  time[i+1,j]<- time[i,j]+0.001
  N[i+1,j]<- N[i,j]+1
  i<- i+2
}
}

}

#plotting trajectories
matplot(time, N, type="l", lty=1, lwd=2, col=c("red", "blue", "green",
"purple", "orange"), xlab="Years", ylab="Amount of SWE, in inches",
panel.first=grid())

```



(c) The code below simulates 5 trajectories spanning over 7 years.

```

#specifying parameters
t<- 7
lambda<- 24.3
time<- data.frame()
N<- data.frame()

#specifying seed
set.seed(1001117)

```



```

#creating loop to simulate trajectories
for(j in 1:5) {

#selecting rate
Lambda<- rpois(1,lambda)

#setting initial values
time[1,j]<- 0
N[1,j]<- 0

#generating N(t)
N.total<- rpois(1,Lambda*t)

#generating N(t) standard uniforms
u<- 1:N.total
for(i in 1:N.total)
  u[i]<- runif(1)

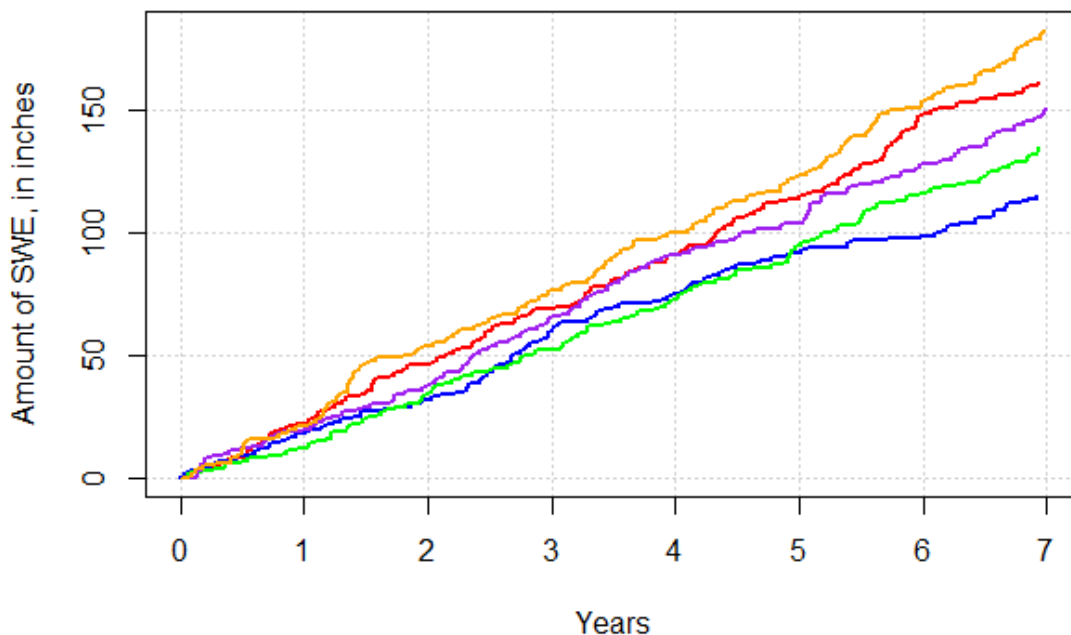
#sorting standard uniforms
u.sorted<- sort(u)

#computing N(t) event times
s<- t*u.sorted

#generating jumps
for (i in seq(2, 2*N.total, 2)) {
  time[i,j]<- s[i/2]-0.001
  time[i+1,j]<- s[i/2]
  N[i,j]<- N[i-1,j]
  N[i+1,j]<- N[i-1,j]+1
}
}

#plotting simulated trajectories
matplot(time, N, type="l", lty=1, lwd=2, col=c("red", "blue", "green",
"purple", "orange"), xlab="Years", ylab="Amount of SWE, in inches",
panel.first=grid())

```



EXERCISE 6.6. (a) Let $N(\ell)$ be the number of defects in an ℓ -yard roll of fabric. We know that $N(\ell) \sim \text{Poisson}(\Lambda\ell)$ where $\Lambda \sim \text{Gamma}(\alpha, \beta)$ with $E(\Lambda) = \alpha/\beta = 0.07$ and $\sqrt{\text{Var}(\Lambda)} = \sqrt{\alpha/\beta^2} = 0.01$. The mean and standard deviation of $N(40)$ are $E(N(40)) = \ell E(\Lambda) = (40)(0.07) = 2.8$ and $\sqrt{\text{Var}(N(40))} = \sqrt{\ell^2 \text{Var}(\Lambda) + \ell E(\Lambda)} = \sqrt{(40)^2(0.01)^2 + (40)(0.07)} = 1.72$.

(b) As derived in Application 6.2(b), for a given $N(t) = n$, the conditional distribution of Λ is gamma with parameters $n + \alpha$ and $\ell + \beta$. The parameters α and β solve $\begin{cases} \alpha/\beta = 0.07 \\ \alpha/\beta^2 = 0.0001 \end{cases}$. From here, $\alpha = 49$ and $\beta = 700$. We are also given that $\ell = 40$ and $n = 4$.

The line of code given below computes the probability to be above 0.08 for a gamma distribution with parameters $n + \alpha = 4 + 49 = 53$ and $\ell + \beta = 40 + 700 = 740$.

```
pgamma(0.08, 53, 740, lower.tail=FALSE)
```

```
0.1932722
```

CHAPTER 7

EXERCISE 7.1. (a) We plug $\lambda_n = n\lambda$ and $\mu_n = 0$ into (7.1), and note that n starts with 1 and not 0. The Kolmogorov forward equations become $P'_1(t) = -\lambda P_1(t)$ and $P'_n(t) = (n-1)\lambda P_{n-1}(t) - n\lambda P_n(t)$, $n = 2, 3, \dots$, with the initial condition $P_1(0) = 1$.

(b) To show that $P_n(t) = e^{-\lambda t}(1 - e^{-\lambda t})^{n-1}$, $n = 1, 2, \dots$, solve the Kolmogorov equations, we write $P_1(t) = e^{-\lambda t}$, so $P'_1(t) = -\lambda e^{-\lambda t} = -\lambda P_1(t)$. Also,

$$P'_n(t) = -\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-1} + e^{-\lambda t}(n-1)\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-2} = -\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-1} + e^{-\lambda t}(n-1)\lambda(e^{-\lambda t} - 1 + 1)(1 - e^{-\lambda t})^{n-2} = -\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-1} - (n-1)\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-1} + (n-1)\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-2} = (n-1)\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-2} - n\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-1} = (n-1)\lambda P_{n-1}(t) - n\lambda P_n(t).$$

(c) The distribution of $X(t)$ is geometric that models the number of trials until the first success where the probability of success is $p = e^{-\lambda t}$. Therefore, $E(X(t)) = \frac{1}{p} = e^{\lambda t}$, and $Var(X(t)) = \frac{1-p}{p^2} = \frac{1-e^{-\lambda t}}{e^{-2\lambda t}} = e^{\lambda t}(e^{\lambda t} - 1)$.

(d) If $\lambda = 4$, the probability that there will be between 3 and 5 particles at week 1 is $P_3(1) + P_4(1) + P_5(1) = e^{-4}(1 - e^{-4})^{3-1} + e^{-4}(1 - e^{-4})^{4-1} + e^{-4}(1 - e^{-4})^{5-1} = 0.051989$. The mean at week 1 is $E(X(1)) = e^4 = 54.59815$, and the standard deviation is $\sqrt{Var(X(1))} = \sqrt{e^4(e^4 - 1)} = 54.09584$.

EXERCISE 7.2. (a) We plug $\lambda_n = n\lambda$ and $\mu_n = 0$ into (7.1) and note that n starts with m and not 0. The Kolmogorov forward equations become $P'_m(t) = -m\lambda P_m(t)$ and $P'_n(t) = (n-1)\lambda P_{n-1}(t) - n\lambda P_n(t)$, $n = 2, 3, \dots$, with the initial condition $P_m(0) = 1$.

(b) To verify that $P_n(t) = \binom{n-1}{n-m} e^{-m\lambda t}(1 - e^{-\lambda t})^{n-m}$, $n = m, m+1, \dots$, solve the Kolmogorov equations, we write $P_m(t) = e^{-m\lambda t}$, so $P'_m(t) = -m\lambda e^{-m\lambda t} = -m\lambda P_m(t)$. Further, $P'_n(t) = -m\lambda \binom{n-1}{n-m} e^{-m\lambda t}(1 - e^{-\lambda t})^{n-m} + \binom{n-1}{n-m} e^{-m\lambda t}(n-m)\lambda e^{-\lambda t}(1 - e^{-\lambda t})^{n-m-1} = -m\lambda P_n(t) + \binom{n-1}{n-m} e^{-m\lambda t}(n-m)\lambda(e^{-\lambda t} - 1 + 1)(1 - e^{-\lambda t})^{n-m-1} = -m\lambda P_n(t) - (n-m)\lambda P_n(t) + (n-m)\binom{n-1}{n-m} \lambda e^{-m\lambda t}(1 - e^{-\lambda t})^{n-m-1} = -n\lambda P_n(t) + (n-1)\lambda \binom{n-2}{n-m-1} e^{-m\lambda t}(1 - e^{-\lambda t})^{n-m-1} = (n-1)\lambda P_{n-1}(t) - n\lambda P_n(t)$.

(c) The distribution of $X(t)$ is a negative binomial that models the number of trials until the m th success, where the probability of success is $p = e^{-\lambda t}$. Therefore, the mean and the variance are $E(X(t)) = \frac{m}{p} = m e^{\lambda t}$, and $Var(X(t)) = \frac{m(1-p)}{p^2} = m e^{\lambda t}(e^{\lambda t} - 1)$.

(d) $P_{12}(2) = \binom{12-1}{12-5} e^{-(5)(0.2)(2)}(1 - e^{-(0.2)(2)})^{12-5} = 0.0189$. The mean and standard deviations are $E(X(2)) = (5)e^{(0.2)(2)} = 7.459123$, and $\sqrt{Var(X(2))} = \sqrt{(5)e^{0.4}(e^{0.4} - 1)} = 1.915354$.

EXERCISE 7.3. (a) In the Kolmogorov forward equations (7.1), we use $\lambda_n = 0$, and $\mu_n = n\mu$, and the fact that the initial population size is N . We write $P'_N(t) = -N\mu P_N(t)$ and $P'_n(t) = (n+1)\mu P_{n+1}(t) - n\mu P_n(t)$, $n = 0, 1, \dots, N-1$, with the initial condition $P_N(0) = 1$.

(b) The probabilities $P_n(t) = \binom{N}{n} e^{-n\mu t} (1 - e^{-\mu t})^{N-n}$, $n = 0, \dots, N$, solve the Kolmogorov forward equations since $P_N(t) = e^{-N\mu t}$ and so, $P'_N(t) = -N\mu e^{-N\mu t} = -N\mu P_N(t)$. Also,

$$\begin{aligned} P'_n(t) &= -n\mu \binom{N}{n} e^{-n\mu t} (1 - e^{-\mu t})^{N-n} + \binom{N}{n} e^{-n\mu t} (N-n)\mu e^{-\mu t} (1 - e^{-\mu t})^{N-n-1} \\ &= -n\mu P_n(t) + (n+1)\mu \binom{N}{n+1} e^{-(n+1)\mu t} (1 - e^{-\mu t})^{N-(n+1)} = (n+1)\mu P_{n+1}(t) - n\mu P_n(t). \end{aligned}$$

(c) The distribution of $X(t)$ is binomial with parameters N and $p = e^{-\mu t}$. Therefore, $E(X(t)) = Np = Ne^{-\mu t}$, and $Var(X(t)) = Np(1-p) = Ne^{-\mu t}(1 - e^{-\mu t})$.

(d) $P_{12}(3) = \binom{15}{12} e^{-(12)(0.02)(3)} (1 - e^{-(0.02)(3)})^{15-12} = 0.0437$. The mean and standard deviation are $E(X(3)) = 15 e^{-(0.02)(3)} = 14.12647$, and $\sqrt{Var(X(3))} = \sqrt{15 e^{-(0.02)(3)} (1 - e^{-(0.02)(3)})} = 0.907007$.

EXERCISE 7.4. (a) We are given that $\lambda = 1.3$ and $\mu = 0.2$. We need to compute

$$P_4(2) = (1 - P_0) \left(1 - \frac{\lambda}{\mu} P_0\right) \left(\frac{\lambda}{\mu} P_0\right)^{n-1} = (1 - P_0) \left(1 - \frac{1.3}{0.2} P_0\right) \left(\frac{1.3}{0.2} P_0\right)^{4-1}$$

where

$$P_0 = \frac{\mu e^{(\lambda-\mu)t} - \mu}{\lambda e^{(\lambda-\mu)t} - \mu} = \frac{0.2 e^{(1.3-0.2)(2)} - 0.2}{1.3 e^{(1.3-0.2)(2)} - 0.2} = 0.139172.$$

Thus, $P_4(2) = 0.060783$. The mean and variance are $E(X(2)) = e^{(\lambda-\mu)t} = e^{(1.3-0.2)(2)} = 9.025013$ and $Var(X(2)) = \frac{\lambda+\mu}{\lambda-\mu} e^{(\lambda-\mu)t} (e^{(\lambda-\mu)t} - 1) = \frac{1.3+0.2}{1.3-0.2} e^{(1.3-0.2)(2)} (e^{(1.3-0.2)(2)} - 1) = 98.76253$.

(b) Below we simulate a 50-step trajectory of the process that starts in state 1 and has parameters $\lambda = 1.3$ and $\mu = 0.2$.

```
#specifying parameters
lambda<- 1.3
mu<- 0.2
njumps<- 50

#defining state and time as vectors
N<- c()
time<- c()

#setting initial values
N[1]<- 1
time[1]<- 0

#specifying seed
set.seed(353332)
```

```

#simulating trajectory
i<- 2

repeat {
  time.birth<- (-1/(N[i-1]*lambda))*log(1-runif(1))
  time.death<- (-1/(N[i-1]*mu))*log(1-runif(1))

  if(time.birth < time.death | N[i-1]==0) {
    time[i]<- time[i-1] + time.birth - 0.001
    N[i]<- N[i-1]

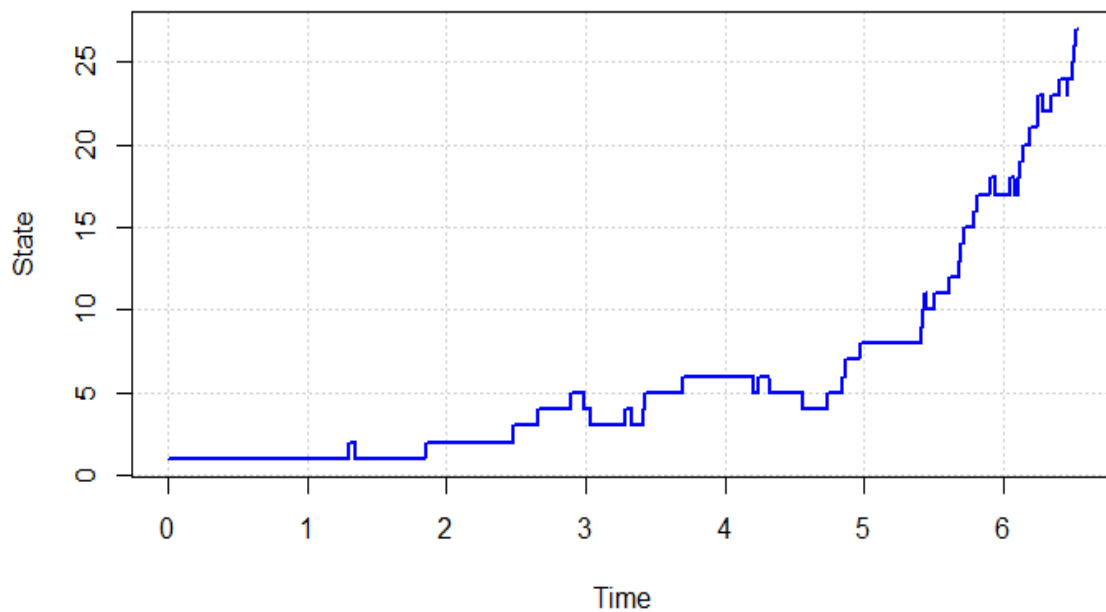
    if(i==2*njumps+2) break
  } else {
    time[i+1]<- time[i] + 0.001
    N[i+1]<- N[i] + 1
    i<- i+2
  }

  if(time.death < time.birth & N[i-1]!=0) {
    time[i]<- time[i-1] + time.death - 0.001
    N[i]<- N[i-1]

    if(i==2*njumps+2) break
  } else {
    time[i+1]<- time[i] + 0.001
    N[i+1]<- N[i] - 1
    i<- i+2
  }
}

#plotting trajectory
plot(time, N, type="l", lty=1, lwd=2, col="blue", xlab="Time", ylab="State",
panel.first=grid())

```



EXERCISE 7.5. (a) If $\lambda > \mu$, the queue will accumulate faster than customers go through the server, and so we expect an infinite number of customers in the system in the long run.

(b) For $\lambda = 3$ and $\mu = 5$, the long-run probability that there will be more than 2 customers in the system is $P(\# \text{ of customers} > 2) = 1 - P_0 - P_1 - P_2 = 1 - \left(1 - \frac{\lambda}{\mu}\right) \mu \left[1 + \frac{\lambda}{\mu} + \left(\frac{\lambda}{\mu}\right)^2\right] = 1 -$

$$\left(1 - \frac{\lambda}{\mu}\right) \frac{1 - \left(\frac{\lambda}{\mu}\right)^3}{1 - \frac{\lambda}{\mu}} = \left(\frac{\lambda}{\mu}\right)^3 = \left(\frac{3}{5}\right)^3 = 0.216.$$

(c) In the long run, the average number of customers in the system is

$$\lim_{t \rightarrow \infty} E(X(t)) = \frac{\lambda}{\mu - \lambda} = \frac{3}{5 - 3} = 1.5.$$

(d) In the long run, the proportion of customers in the system who have to wait more than 1 minute is $P(T > 1) = e^{-(\mu - \lambda)t} = e^{-(5 - 3)(1)} = 0.135335$, or roughly 13.5%.

EXERCISE 7.6. (a) Below we simulate a trajectory of a birth-and-death process with immigration and emigration, with parameters $\lambda = 1, \mu = 0.2, \alpha = 0.3$, and $\beta = 0.1$. The trajectory starts in state 10 and ends in state 25.

```
#specifying parameters
lambda<- 1
mu<- 0.2
alpha<- 0.3
beta<- 0.1

#defining state and time as vectors
N<- c()
time<- c()

#setting initial values
N[1]<- 10
time[1]<- 0

#specifying seed
set.seed(93743765)

#simulating trajectory
i<- 2

repeat {

  time.birth<- (-1/(N[i-1]*lambda+alpha))*log(1-runif(1))
  time.death<- (-1/(N[i-1]*mu+beta))*log(1-runif(1))

  if(time.birth < time.death | N[i-1]==0) {
    time[i]<- time[i-1] + time.birth - 0.001
    N[i]<- N[i-1]

    if(N[i]==25) break
  } else {

    time[i+1]<- time[i] + 0.001
    N[i+1]<- N[i] + 1
  }
}
```

```

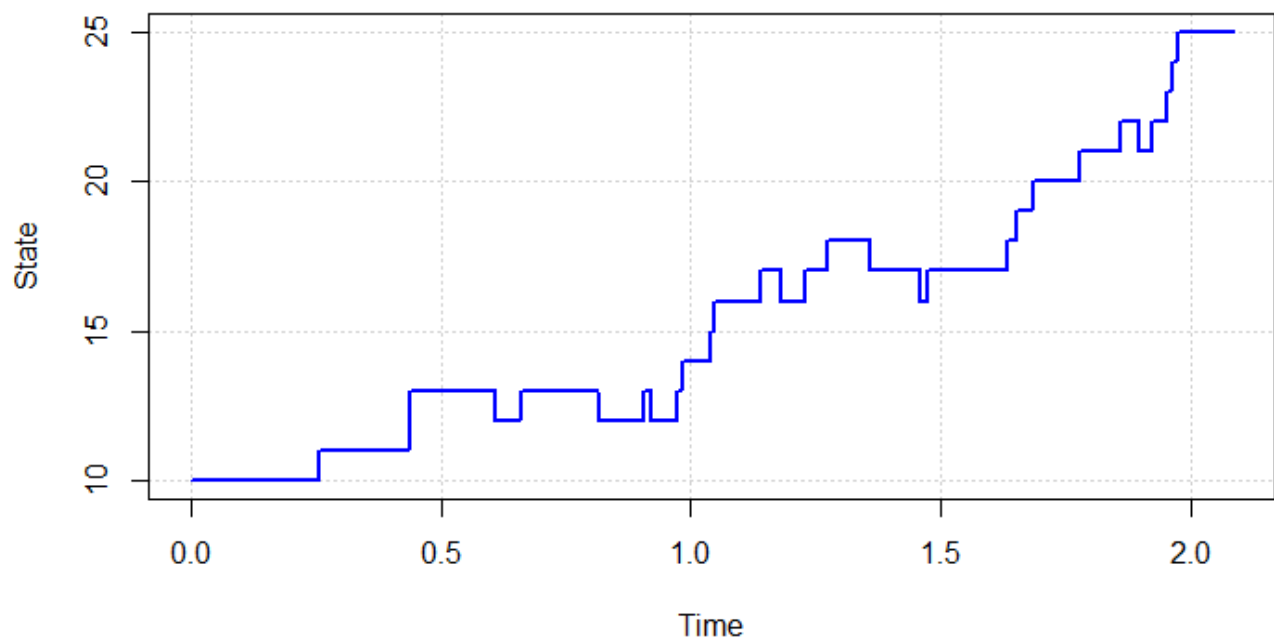
i<- i+2
}
}

if(time.death < time.birth & N[i-1]!=0) {
  time[i]<- time[i-1] + time.death - 0.001
  N[i]<- N[i-1]

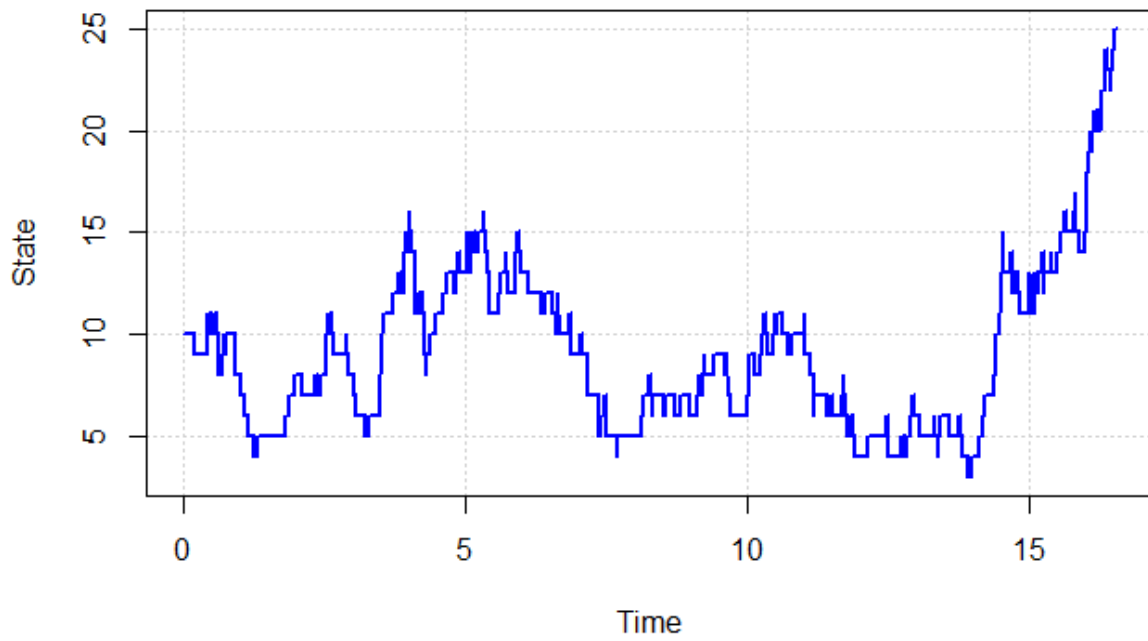
  if(N[i]==25) break
  else {

    time[i+1]<- time[i] + 0.001
    N[i+1]<- N[i] - 1
    i<- i+2
  }
}
}
#plotting trajectory
plot(time, N, type="l", lty=1, lwd=2, col="blue", xlab="Time", ylab="State",
panel.first=grid())

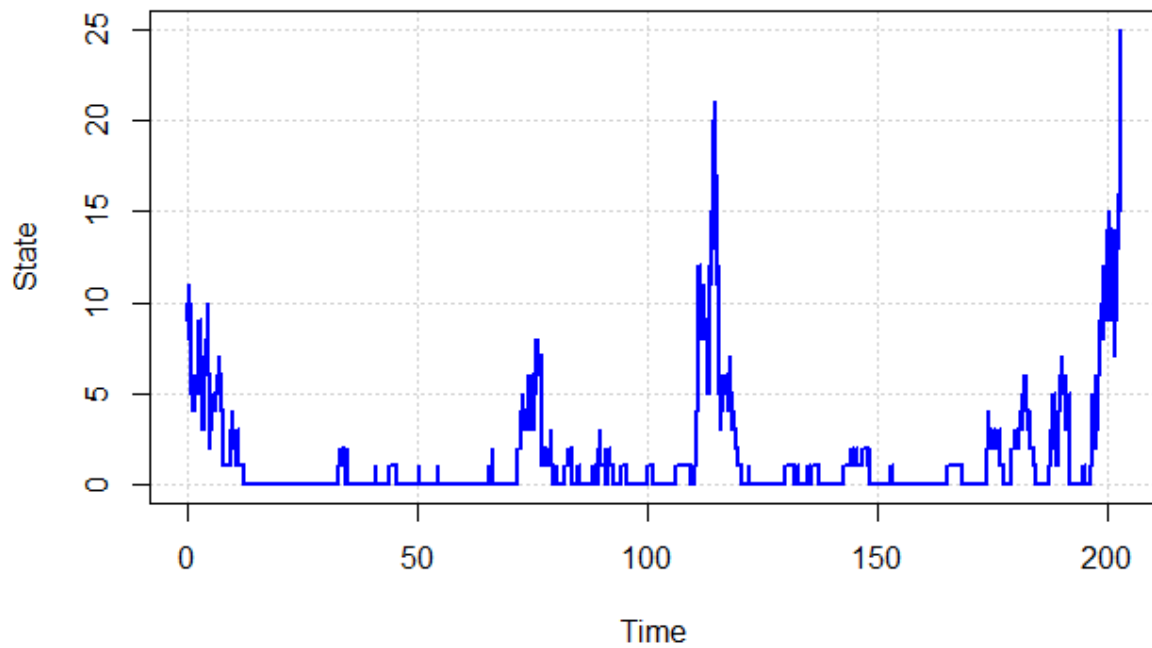
```



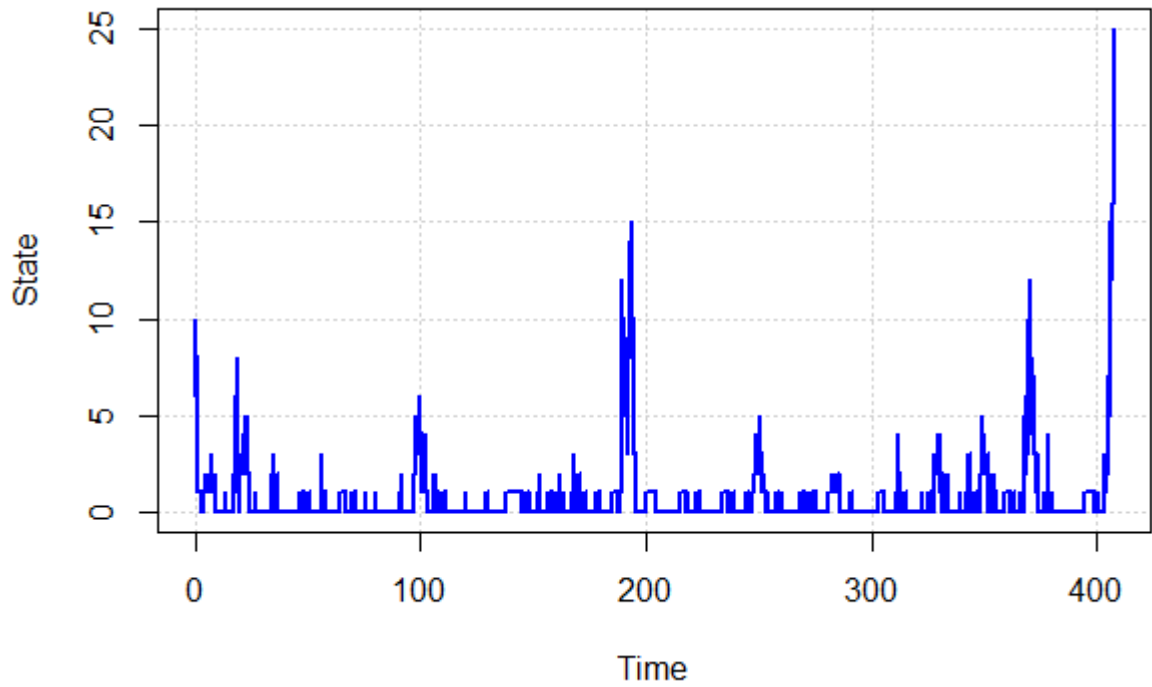
(b) To simulate a trajectory with $\mu = 0.8$, we run the same code but specify the parameter as:
`mu<- 0.8`. The plot is



(c) In the same code we specify `mu<- 1`. The simulated trajectory is



(d) Now we specify `mu<- 1.2`. The graph is



(e) In the plots above we see that as the death rate increases, the population relies on immigration more and more. When $\mu = 0.2$, the flock size grows from 10 to 25 birds within roughly 2 time units. When $\mu = 0.8$, it takes about 16 time units to grow. When $\mu = 1$, it takes about 200 time units to grow. When $\mu = 1.2$, it takes about 400 time units to grow. In the last two cases ($\mu = 1$ and $\mu = 1.2$), the flock keeps dying out and revives due to bird immigration.

CHAPTER 8

EXERCISE 8.1. (a) For each bacterium, the offspring size Z has mean $\mu = E(Z) = (0)(0.25) + (1)(0.15) + (2)(0.6) = 1.35 > 1$, thus the colony growth is a supercritical branching process. The variance of Z is $\sigma^2 = Var(Z) = (0)^2(0.25) + (1)^2(0.15) + (2)^2(0.6) - (1.35)^2 = 0.7275$.

The expected size of the n th generation is $100E(X_n) = 100\mu^n = (100)(1.35)^n$. The variance is $100 Var(X_n) = 100\sigma^2\mu^{n-1} \frac{1-\mu^n}{1-\mu} = (100)(0.7275)(1.35)^{n-1} \frac{(1.35)^n - 1}{1.35 - 1} = (207.8571)(1.35)^{n-1}((1.35)^n - 1)$.

(b) Let $\pi_0 < 1$ denote the extinction probability for descendants of one bacterium. It solves the equation $\pi_0 = \pi_0^0(0.25) + \pi_0^1(0.15) + \pi_0^2(0.6)$, or, equivalently, $0.6\pi_0^2 - 0.85\pi_0 + 0.25 = (\pi_0 - 1)(0.6\pi_0 - 0.25) = 0$. Thus, $\pi_0 = \frac{0.25}{0.6} = 0.4167$.

(c) $P(\text{extinction of descendants of at least one of ten bacteria}) = 1 - P(\text{no extinction}) = 1 - (1 - 0.4167)^{10} = 0.9954$.

EXERCISE 8.2. (a) Let Z denote the size of offspring. We are given that $Z \sim Poi(\lambda)$. The mean of Z is $\mu = E(Z) = \lambda$. If $\lambda > 1$, the process is supercritical; if $\lambda = 1$, the process is critical; if $\lambda < 1$, the process is subcritical.

(b) The variance of the offspring size is $\sigma^2 = Var(Z) = \lambda$. The mean of the size of the n th generation X_n is $E(X_n) = \mu^n = \lambda^n$. The variance is $Var(X_n) = \sigma^2\mu^{n-1} \frac{1-\mu^n}{1-\mu} = \lambda \lambda^{n-1} \frac{1-\lambda^n}{1-\lambda} = \lambda^n \frac{1-\lambda^n}{1-\lambda}$, if $\lambda \neq 1$; and $Var(X_n) = \sigma^2 n = \lambda n$, if $\lambda = 1$.

(c) Let $\pi_0 < 1$ denote the extinction probability. It is the smallest positive solution of the equation

$$\pi_0 = \sum_{n=0}^{\infty} \pi_0^n \frac{\lambda^n}{n!} e^{-\lambda} = e^{-\lambda} \sum_{n=0}^{\infty} \frac{(\pi_0 \lambda)^n}{n!} = e^{-\lambda} e^{\pi_0 \lambda} = e^{\lambda(\pi_0 - 1)}.$$

Below is the code and the graph of the numeric solution π_0 of this equation as a function of $\lambda > 1$.

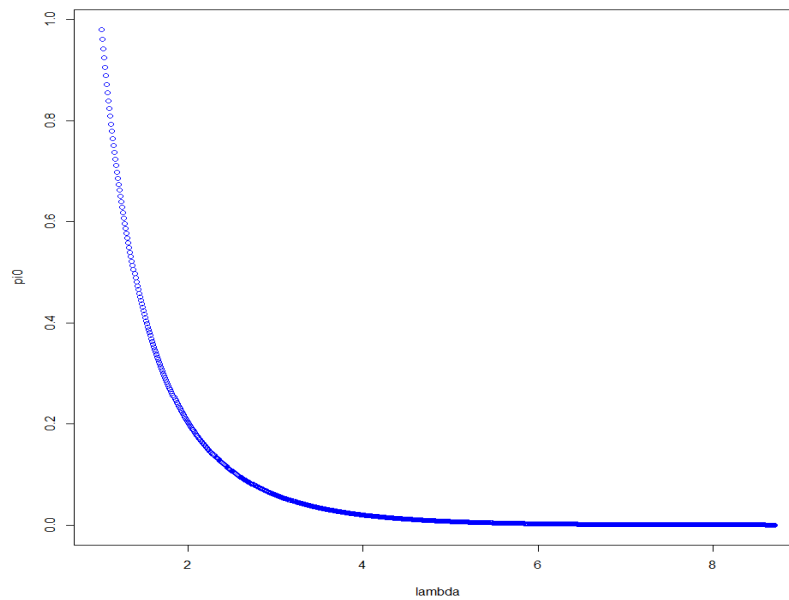
```
library(rootSolve)

lambda<- seq(1.01,8.7,0.01)
pi0<- 1:770

for (i in 1:770) {
equation<- function(x)
  x-exp(lambda[i]*(x-1))

pi0[i]<- uniroot.all(equation, c(0,0.99))
}

plot(lambda,pi0, col="blue")
```



```

> pi0
[1] 0.9801000000 0.9608919215
[3] 0.9422288852 0.9240007044
[5] 0.9061766745 0.8891012444
[7] 0.8721393433 0.8557623756
[9] 0.8396258028 0.8239578325
[11] 0.8086784805 0.7937183106
[13] 0.7791773528 0.7649519043
[15] 0.7510401355 0.7375289728
[17] 0.7242501181 0.7113162379
[19] 0.6987154931 0.6863624122
[21] 0.6742636161 0.6624665795
[23] 0.6509652371 0.6397054345
[25] 0.6286859791 0.6179043043
[27] 0.6073567259 0.5970386674
[745] 0.0003700480 0.0003700480
[747] 0.0003700480 0.0003700480
[749] 0.0003700480 0.0003700480
[751] 0.0003700480 0.0003700480
[753] 0.0003700480 0.0003700480
[755] 0.0003700480 0.0003700480
[757] 0.0003700480 0.0003700480
[759] 0.0003700480 0.0000000000
[761] 0.0000000000 0.0000000000
[763] 0.0000000000 0.0000000000
[765] 0.0000000000 0.0000000000
[767] 0.0000000000 0.0000000000
[769] 0.0000000000 0.0000000000

```

EXERCISE 8.3. (a) Let Z denote the size of male offspring. It is given that $P(Z = 0) = 0.4828$, and $P(Z = n) = (0.228292)(0.5586)^{n-1}$, $n = 1, 2, \dots$. The expected value and variance of Z are $\mu = E(Z) = (0)(0.4828) + (0.228292) \sum_{n=1}^{\infty} n(0.5586)^{n-1} = \frac{0.228292}{(1-0.5586)^2} = 1.171726$, and $\sigma^2 = \text{Var}(Z) = (0)^2(0.4828) + (0.228292) \sum_{n=1}^{\infty} n^2(0.5586)^{n-1} - (1.171726)^2 = (0.228292) \left[\frac{0.5586}{(1-0.5586)^2} + \frac{1}{(1-0.5586)^2} \right] - (1.171726)^2 = 0.45331$.

(b) The expected size and variance of the n th generation are $E(X_n) = \mu^n = (1.171726)^n$, and $\text{Var}(X_n) = \sigma^2 \mu^{n-1} \frac{1-\mu^n}{1-\mu} = (0.45331)(1.171726)^{n-1} \frac{1-(1.171726)^n}{1-1.171726} = (2.639728)(1.171726)^{n-1}((1.171726)^n - 1)$.

(c) Let π_0 denote the probability of extinction. It is found as the smallest positive solution of the equation

$$\pi_0 = \pi_0^0(0.4828) + \sum_{n=1}^{\infty} \pi_0^n (0.228292)(0.5586)^{n-1} = 0.4828 + \frac{0.228292}{0.5586} \sum_{n=1}^{\infty} (0.5586\pi_0)^n = 0.4828 + \frac{0.228292}{0.5586} \left(\frac{1}{1-0.5586\pi_0} - 1 \right) = 0.4828 + \frac{0.228292\pi_0}{1-0.5586\pi_0}, \text{ which is a quadratic equation}$$

$$0.5586\pi_0^2 - 1.0414\pi_0 + 0.4828 = 0. \text{ The solution is } \pi_0 = 0.864304.$$

EXERCISE 8.4.

(a) The mean winnings of a stake of \$1 is $\mu = (\$1)(0.3) + (\$15)(0.2) + (\$20)(0.1) + (\$0)(0.4) = \$5.3 > 1$. Since $5.3 > 1$, this the process is supercritical.

(b) The mean winning of n th bet is $\mu^n = (\$5.3)^n$, and therefore, the expected winning on the fifth bet is $\mu^5 = (\$5.3)^5 = \$4,181.96$.

(c) Let π_0 be the probability that the gambler's stake eventually turns into \$0. It is the smallest positive solution of the equation $\pi_0 = 0.4 + 0.3\pi_0 + 0.2\pi_0^{15} + 0.1\pi_0^{20}$. Solved numerically, $\pi_0 = 0.5714957$. The R script follows.

```
library(rootSolve)

equation<- function(x)
  0.4-0.7*x+0.2*x^15+0.1*x^20

Print(pi0<- uniroot.all(equation, c(0,0.99)))

0.5714957
```

EXERCISE 8.5. (a) The mean number of computers that are infected in one day is $\mu = \frac{1}{4}(0 + 1 + 2 + 3) = 1.5$. Since the mean is larger than one, it is a supercritical process.

(b) The average number and standard deviation of infected computers on day 10 are

$$E(X_{10}) = \mu^{10} = (1.5)^{10} = 57.66504, \text{ and}$$

$$\sqrt{Var(X_{10})} = \sqrt{\left(\frac{1}{4}(0^2 + 1^2 + 2^2 + 3^2) - (1.5)^2\right)(1.5)^9 \frac{1-(1.5)^{10}}{1-1.5}} = \sqrt{5,445.9862} = 73.7969.$$

(c) Since the process of virus spread is a supercritical branching process, the virus will not die out with probability one. The probability of its extinction π_0 is the smallest positive solution of the equation $\pi_0 = \frac{1}{4}(1 + \pi_0 + \pi_0^2 + \pi_0^3)$, or, equivalently,

$$1 - 3\pi_0 + \pi_0^2 + \pi_0^3 = (\pi_0 - 1)(\pi_0 - (-1 + \sqrt{2}))(\pi_0 - (-1 - \sqrt{2})) = 0.$$

Hence, $\pi_0 = -1 + \sqrt{2} = 0.4142$.

(d) Below we simulate the number of infected computers for 10 days. A total of 424 computers became infected in this simulation.

```
#specifying parameters
p<- c(0.25, 0.25, 0.25, 0.25)
```

```

N<- c()
N[1]<- 1

#specifying seed
set.seed(1088878)

#simulating offspring
for (i in 2:11) {
  Z<- 0
  for (j in 1:N[i-1]) {
    Z<- Z + sample(0:3, 1, prob=p)
  }

  N[i]<- Z

  if (N[i]==0) break
}

N

1  2  5  9 19 28 33 52 72 90 113

sum(N)

424

```

EXERCISE 8.6. (a) The code below generates the offspring of the first 20 generations of the branching process and calculates the total population size.

```

#specifying parameters
p<- c(0.1, 0.4, 0.5)
N<- c()
N[1]<- 1

#specifying seed
set.seed(377584410)

#simulating offspring
for (i in 2:20) {
  Z<- 0
  for (j in 1:N[i-1]) {
    Z<- Z + sample(0:2, 1, prob=p)
  }

  N[i]<- Z

  if (N[i]==0) break
}

N

1  1  2  4  6  8 11 15 21 30 39 56 79 101 129 181 251 345 468 667

```

In this simulation, the 20th generation size is 667 particles, which constitute the offspring of the 19th generation.

```
sum(N)
```

There are a total of 2,415 particles in the 20 generations.

- (a) The code below simulates a sample trajectory of this process and plots the branching process for 6 generations.

```

Library(tidyverse)

#specifying parameters
gen.max<- 6
p<- c(0.1, 0.4, 0.5)

#specifying seed
set.seed(332975)

#simulating trajectory
level.segment <- function(gen, y, branch.num) {

  branch<- data.frame(x=c(), y=c(), xend=c(), yend=c())
  gen.remaining<- gen.max-gen-1

  if (gen.remaining < 0) return(branch)

  if (branch.num > 0) {
    branch<- rbind(branch, data.frame(x=gen, y=y, xend=gen+1, yend=y),
      level.segment(gen=gen+1, y=y, branch.num=sample(0:2, 1, prob=p)))
  }

  if (branch.num > 1) {
    branch<- rbind(branch, data.frame(x=gen, y=y, xend=gen+1,
      yend=y+3^gen.remaining), level.segment(gen=gen+1, y=y+3^gen.remaining,
      branch.num=sample(0:2, 1, prob=p)))
  }

  if (branch.num > 2) {
    branch<- rbind(branch, data.frame(x=gen, y=y, xend=gen+1,
      yend=y-3^gen.remaining), level.segment(gen=gen+1, y=y-3^gen.remaining,
      branch.num=sample(0:2, 1, prob=p)))
  }

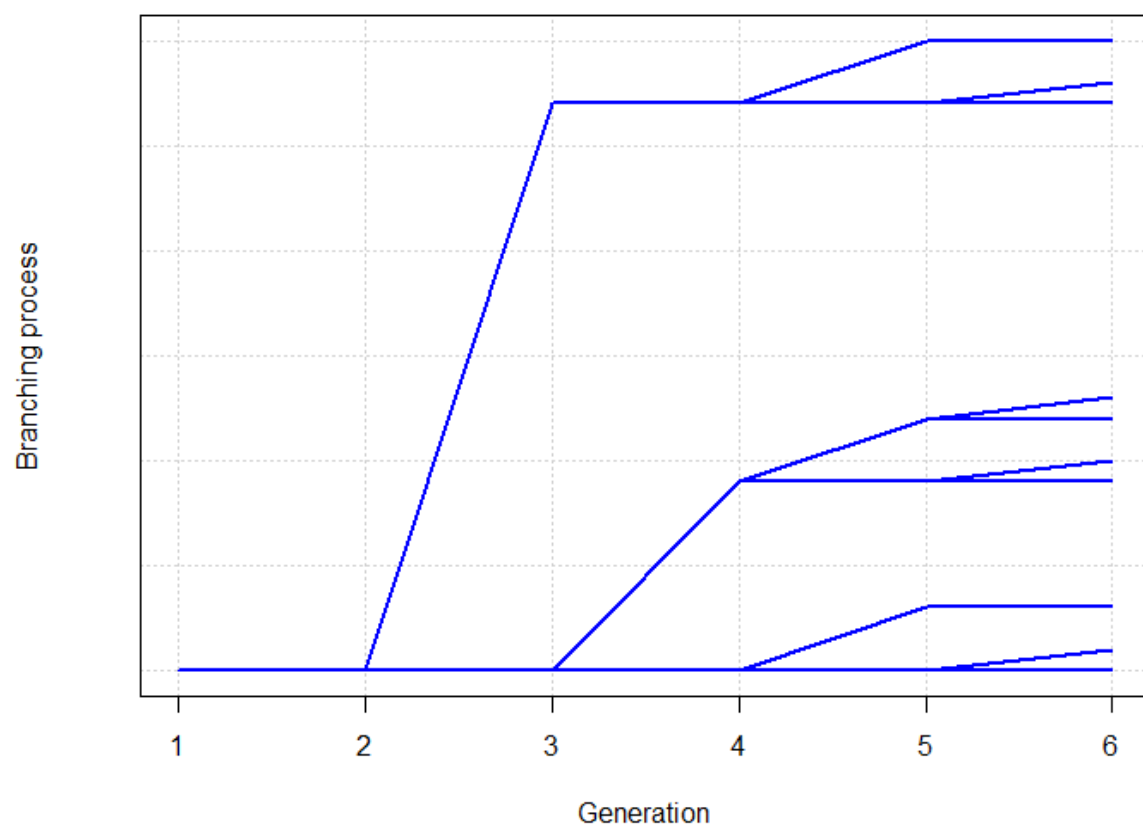
  branch
}

bp<- level.segment(1, 0, sample(0:2, 1, prob=p))

#plotting trajectory
plot(bp[,1], bp[,2], type="n", yaxt="n", xlim=c(1,6), ylim=c(range(bp)),
  xlab="Generation", ylab="Branching process", panel.first=grid())

segments(bp[,1], bp[,2], bp[,3], bp[,4], lwd=2, col="blue")

```



CHAPTER 9

EXERCISE 9.1.

$$\rho(B(s), B(t)) = \frac{\text{Cov}(B(s), B(t))}{\sqrt{\text{Var}(B(s))}\sqrt{\text{Var}(B(t))}} = \frac{\min(s, t)}{\sqrt{s}\sqrt{t}} = \begin{cases} \frac{s}{\sqrt{s}\sqrt{t}} = \sqrt{\frac{s}{t}}, & \text{if } s < t, \\ \frac{t}{\sqrt{s}\sqrt{t}} = \sqrt{\frac{t}{s}}, & \text{if } t < s \end{cases} = \sqrt{\frac{\min(s, t)}{\max(s, t)}}.$$

EXERCISE 9.2. (a) $X(t) = t B\left(\frac{1}{t}\right)$ has mean $E(X(t)) = tE\left(B\left(\frac{1}{t}\right)\right) = 0$ and variance

$\text{Var}(X(t)) = t^2 \left(\frac{1}{t}\right) = t$. Also, it has a normal distribution with independent and stationary increments since $B(t)$ does. Therefore, $X(t)$ is a standard Brownian motion.

(b) $Y(t) = \alpha B_1(t) + \sqrt{1 - \alpha^2} B_2(t)$ has a normal distribution as a linear combination of two normally distributed random variables. It also has independent and stationary increments, inherited from $B_1(t)$ and $B_2(t)$. The mean is $E(Y(t)) = \alpha E(B_1(t)) + \sqrt{1 - \alpha^2} E(B_2(t)) = 0$, and the variance is $\text{Var}(Y(t)) = \text{Var}(\alpha B_1(t) + \sqrt{1 - \alpha^2} B_2(t)) = \alpha^2 \text{Var}(B_1(t)) + (1 - \alpha^2) \text{Var}(B_2(t)) = \alpha^2 t + (1 - \alpha^2) t = t$, thus, $Y(t)$ is a standard Brownian motion.

EXERCISE 9.3.

(a) $P(0 < B(1) < 1, 1 < B(3) - B(1) < 3) = P(0 < B(1) < 1)P(1 < B(3) - B(1) < 3)$
(by independence of increments)

$= P(0 < B(1) < 1)P(1 < B(2) < 3)$ (by stationarity of increments)

$= P(0 < B(1) < 1)P(1 < \sqrt{2}B(1) < 3)$ (by rescaling of Brownian motion)

$= (\Phi(1) - \Phi(0)) \left(\Phi\left(\frac{3}{\sqrt{2}}\right) - \Phi\left(\frac{1}{\sqrt{2}}\right) \right) = 0.076053.$

(b) $P(0 < B(1) < 1, 1 < B(2) < 3) = P(0 < B(1) < 1, 1 - B(1) < B(2) - B(1) < 3 - B(1))$
 $= \int_0^1 P(1 - x < B(2) - B(1) < 3 - x \mid B(1) = x) f_{B(1)}(x) dx$

$= \int_0^1 P(1 - x < B(1) < 3 - x) f_{B(1)}(x) dx$ (by independence and stationarity of increments)

$$= \int_0^1 \int_{1-x}^{3-x} \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dy dx = 0.2198108.$$

```
func<- function(x) (pnorm (3-x,0,1)-pnorm(1-x,0,1))*pnorm(x,0,1)
```

```
integrate(func, 0, 1)
```

0.2198108 with absolute error < 2.4e-15

(c) $P(0 < B(1) < 1, 0 < B(2) < \infty) = P(0 < B(1) < 1, -B(1) < B(2) - B(1) < \infty) =$
 $\int_0^1 P(-x < B(2) - B(1) < \infty \mid B(1) = x) f_{B(1)}(x) dx = \int_0^1 P(-x < B(1) < \infty) f_{B(1)}(x) dx$
(by independence and stationarity of increments)

$= \int_0^1 P(-\infty < B(1) < x) f_{B(1)}(x) dx$ (by the symmetry of normal distribution)

$$= \int_0^1 \Phi(x) d\Phi(x) = \frac{1}{2} (\Phi^2(1) - \Phi^2(0)) = 0.228931.$$

EXERCISE 9.4. The distribution of $B(s) + B(t)$ is normal as the sum of two normally distributed random variables. The mean is $E(B(s) + B(t)) = E(B(s)) + E(B(t)) = 0$, and variance is $\text{Var}(B(s) + B(t)) = \text{Var}(2B(s) + B(t) - B(s)) = \text{Var}(2B(s)) + \text{Var}(B(t) - B(s))$ (by independence of increments) $= 4\text{Var}(B(s)) + \text{Var}(B(t - s))$ (by stationarity of increments) $= 4s + t - s = 3s + t$. Alternatively, $\text{Var}(B(s) + B(t)) = \text{Var}(B(s)) + 2\text{cov}(B(s), B(t)) + \text{Var}(B(t)) = s + 2\min(s, t) + t = 3s + t$.

EXERCISE 9.5. (a) For $x \geq 0$, $P(|B(t)| \leq x) = P(-x \leq B(t) \leq x) = \int_{-x}^x \frac{1}{\sqrt{2\pi t}} e^{-\frac{u^2}{2t}} du$
 $= 2 \int_0^{x/\sqrt{t}} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz = 2 \left(\Phi\left(\frac{x}{\sqrt{t}}\right) - \Phi(0) \right) = 2 \left(\Phi\left(\frac{x}{\sqrt{t}}\right) - \frac{1}{2} \right) = 2\Phi\left(\frac{x}{\sqrt{t}}\right) - 1 = F_{M(t)}(x).$
 (b) The density of $M(t)$ is $F'_{M(t)}(x) = 2\Phi'\left(\frac{x}{\sqrt{t}}\right) = 2 \frac{1}{\sqrt{2\pi}} e^{-\frac{(x/\sqrt{t})^2}{2}} = \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2t}}, x \geq 0$. Therefore, the mean is computed as

$$E(M(t)) = \int_0^\infty \sqrt{\frac{2}{\pi}} x e^{-\frac{x^2}{2t}} dx = \{z = x/\sqrt{t}\} = \sqrt{\frac{2t}{\pi}} \int_0^\infty e^{-\frac{z^2}{2}} d\frac{z^2}{2} = \sqrt{\frac{2t}{\pi}}.$$

(c) The theoretical mean is $E(M(5)) = \sqrt{\frac{(2)(5)}{\pi}} = 1.784124$. Below we simulate 1,000 trajectories of a standard Brownian motion on the interval $[0, 5]$ and calculate the sample mean of the maximum $\hat{E}(M(t))$.

```
#defining Brownian motion as matrix
BM<- matrix(NA, nrow=5000, ncol=1000)

#specifying seed
set.seed(8022022)

#simulating trajectories
for (j in 1:1000) {
  BM[1,j]<- 0

  for (i in 2:5000)
    BM[i,j]<- BM[i-1,j] + sqrt(0.001)*rnorm(1)
}

#computing maximum of each trajectory
max.BM<- c()
for (j in 1:1000)
  max.BM[j]<- max(BM[,j])

#computing mean of maxima
mean(max.BM)

1.780714
```

EXERCISE 9.6. (a) Consider the process $\{-B(t), t \geq 0\}$. It has a normal distribution with independent and stationary increments since $B(t)$ does. The mean is $E(-B(t)) = -E(B(t)) = 0$, and variance is $Var(-B(t)) = (-1)^2 Var(B(t)) = Var(B(t)) = t$. Therefore, it is a standard Brownian motion.

(b) To find the cumulative distribution function of $\min_{0 \leq s \leq t} B(s)$, we write for $x \leq 0$,

$$\begin{aligned} P\left(\min_{0 \leq s \leq t} B(s) \leq x\right) &= 1 - P\left(\min_{0 \leq s \leq t} B(s) > x\right) = 1 - P(B(s) > x, \forall s \in [0, t]) \\ &= 1 - P(-B(s) < -x, \forall s \in [0, t]) = 1 - P\left(\max_{0 \leq s \leq t} (-B(s)) < -x\right) \\ &= 1 - \left(2\Phi\left(\frac{-x}{\sqrt{t}}\right) - 1\right) = 2\left(1 - \Phi\left(\frac{-x}{\sqrt{t}}\right)\right) = 2\Phi\left(\frac{x}{\sqrt{t}}\right) \text{ (by symmetry).} \end{aligned}$$

(c) $P\left(\min_{0 \leq s \leq 5} B(s) < -3\right) = 2\Phi\left(\frac{-3}{\sqrt{5}}\right) = 0.179712.$

(d) The code below generates 1,000 trajectories of a standard Brownian motion and calculates the empirical probability that the minimum is less than -3 on the interval [0,5].

```
#specifying seed
set.seed(2541165)

#simulating trajectories

BM<- matrix(NA, nrow=500, ncol=1000)

for (j in 1:1000) {
  BM[1,j]<- 0

  for (i in 2:500)
    BM[i,j]<- BM[i-1,j] + sqrt(0.01)*rnorm(1)
}

#computing indicator of minimum < -3
ind<- c()

for(j in 1:1000)

  ind[j]<- ifelse(min(BM[,j])< -3, 1,0)

sum(ind)/1000

0.179
```

EXERCISE 9.7. (a) Consider the process

$$X(t) = \begin{cases} (1-t)B\left(\frac{t}{1-t}\right), & \text{if } 0 \leq t < 1, \\ 0, & \text{if } t = 1, \end{cases}$$

where $\{B(t), t \geq 0\}$ is a standard Brownian motion. Note that $X(0) = B(0) = 0 = X(1)$, so the process is tied at the endpoints of a unit interval. It has a normal distribution since $B(\cdot)$ is normally distributed. Its mean is $E(X(t)) = (1-t)E\left(B\left(\frac{t}{1-t}\right)\right) = 0$, and variance is $Var\left((1-t)B\left(\frac{t}{1-t}\right)\right) = (1-t)^2\left(\frac{t}{1-t}\right) = t(1-t)$. The covariance is

$$\begin{aligned} Cov(X(s), X(t)) &= (1-s)(1-t) Cov\left[B\left(\frac{s}{1-s}\right), B\left(\frac{t}{1-t}\right)\right] = (1-s)(1-t) \min\left(\frac{s}{1-s}, \frac{t}{1-t}\right) \\ &= \begin{cases} (1-s)(1-t) \frac{s}{1-s}, = s-st, & \text{if } s < t, \\ (1-s)(1-t) \frac{t}{1-t}, = t-st, & \text{if } t < s \end{cases} = \min(s, t) - st. \end{aligned}$$

Thus, $X(t)$ is a Brownian bridge.

(b) Consider the process $\{B(t) = (1+t)X\left(\frac{t}{1+t}\right), t \geq 0\}$, where $\{X(t), 0 \leq t \leq 1\}$ is a Brownian bridge. It is normally distributed since $X(t)$ is. The mean is $E(B(t)) = (1+t)E\left(X\left(\frac{t}{1+t}\right)\right) = 0$, and the variance is $Var(B(t)) = (1+t)^2 Var\left(X\left(\frac{t}{1+t}\right)\right) = (1+t)^2 \left(\frac{t}{1+t}\right) \left(1 - \frac{t}{1+t}\right) = t$. The covariance is $Cov(B(s), B(t)) = (1+s)(1+t) Cov\left(X\left(\frac{s}{1+s}\right), X\left(\frac{t}{1+t}\right)\right) = (1+s)(1+t) \left[\min\left(\frac{s}{1+s}, \frac{t}{1+t}\right) - \left(\frac{s}{1+s}\right) \left(\frac{t}{1+t}\right) \right] = (1+s)(1+t) \left[\frac{\min(s, t)}{1 + \min(s, t)} - \left(\frac{s}{1+s}\right) \left(\frac{t}{1+t}\right) \right] = \frac{(1+s)(1+t) \min(s, t)}{1 + \min(s, t)} - st = (1 + \max(s, t)) \min(s, t) - st = \min(s, t) + st - st = \min(s, t)$. The above properties indicate that $\{B(t), t \geq 0\}$ is a standard Brownian motion.

(c) Let $\{B(t) = X(t) + tZ, 0 \leq t \leq 1\}$ where $\{X(t), 0 \leq t \leq 1\}$ is a Brownian bridge and Z is an independent standard normal random variable. The distribution of $B(t)$ is normal being a linear combination of two normally distributed random variables. The mean is $E(B(t)) = E(X) + tE(Z) = 0$, and the variance is $Var(B(t)) = Var(X(t) + tZ) = Var(X(t)) + t^2 Var(Z) = t(1-t) + t^2 = t$. The covariance is $Cov(B(s), B(t)) = Cov(X(s) + sZ, X(t) + tZ) = Cov(X(s), X(t)) + st Var(Z) = \min(s, t) - st + st = \min(s, t)$. These indicate that the process $\{B(t), 0 \leq t \leq 1\}$ is a standard Brownian motion.

EXERCISE 9.8. (a) For $0 \leq s < t$, $P(B(s) \leq x \mid B(t) = w) = \int_{-\infty}^x f_{B(s) \mid B(t)}(u \mid w) du$

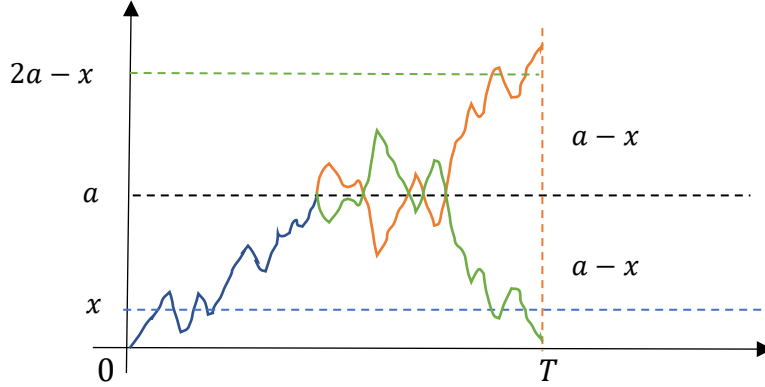
$$= \int_{-\infty}^x \frac{f_{B(s), B(t)-B(s)}(u, w-u)}{f_{B(t)}(w)} du = \int_{-\infty}^x \frac{f_{B(s)}(u) f_{B(t)-B(s)}(w-u)}{f_{B(t)}(w)} du \quad (\text{by independence of increments})$$

$$= \int_{-\infty}^x \frac{\frac{1}{\sqrt{2\pi s}} e^{-\frac{u^2}{2s}} \frac{1}{\sqrt{2\pi(t-s)}} e^{-\frac{(w-u)^2}{2(t-s)}}}{\frac{1}{\sqrt{2\pi t}} e^{-\frac{w^2}{2t}}} du \quad (\text{by stationarity of increments})$$

$$= \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} \sqrt{\frac{t}{s(t-s)}} e^{-\frac{(u - \frac{sw}{t})^2}{2s(t-s)/t}} du, \text{ which means that the conditional distribution is normal with mean } \frac{s}{t} B(t) \text{ and variance } \frac{s}{t} (t-s).$$

(b) Brownian motion, conditional on the value of the endpoint $B(t) = 0$ is indeed a Brownian bridge on the interval $[0, t]$. It is normally distributed with independent and stationary increments (inherited from the Brownian motion). Its mean is $\frac{s}{t}B(t) = 0$ and variance is $\frac{s}{t}(t - s) = s\left(1 - \frac{s}{t}\right)$.

EXERCISE 9.9. (a) As seen from the picture below, $P(M(T) \geq a, B(T) \leq x) = P(M(T) \geq a, B(T) \geq 2a - x)$. That is, by the reflection principle, once the Brownian motion hits level a , the probability to end up at time T at x or below is the same as the probability of ending up at or above $2a - x$.



And if the Brownian motion ends up at or above $2a - x$ at time T , its maximum is definitely above a , so the event $M(T) \geq a$ can be omitted. Hence, $P(M(T) \geq a, B(T) \leq x) = P(B(T) \geq 2a - x)$, where $a > 0$ and $0 \leq x \leq a$.

(b) The joint cdf of $M(T)$ and $B(T)$ can be obtained as follows:

$$P(M(T) \leq a, B(T) \leq x) = P(B(T) \leq x) - P(M(T) \geq a, B(T) \leq x) = P(B(T) \leq x) - P(B(T) \geq 2a - x) = \Phi\left(\frac{x}{\sqrt{T}}\right) - \left(1 - \Phi\left(\frac{2a-x}{\sqrt{T}}\right)\right), \quad a > 0 \text{ and } 0 \leq x \leq a.$$

The joint density of $M(T)$ and $B(T)$ is found by differentiating the joint cdf with respect to a and x :

$$\begin{aligned} f_{M(T), B(T)}(a, x) &= \frac{\partial^2}{\partial a \partial x} \left[\Phi\left(\frac{x}{\sqrt{T}}\right) - \left(1 - \Phi\left(\frac{2a-x}{\sqrt{T}}\right)\right) \right] = \frac{2}{\sqrt{T}} \frac{1}{\sqrt{2\pi}} \frac{2(2a-x)}{2T} e^{-\frac{(2a-x)^2}{2T}} \\ &= \frac{2(2a-x)}{T\sqrt{2\pi T}} e^{-\frac{(2a-x)^2}{2T}}, \quad a > 0, 0 \leq x \leq a. \end{aligned}$$

The conditional density of $M(T)$ given that $B(T) = x$ is calculated as follows:

$$f_{M(T)|B(T)}(a|x) = \frac{f_{M(T), B(T)}(a, x)}{f_{B(T)}(x)} = \frac{\frac{2(2a-x)}{T\sqrt{2\pi T}} e^{-\frac{(2a-x)^2}{2T}}}{\frac{1}{\sqrt{2\pi T}} e^{-\frac{x^2}{2T}}} = \frac{2(2a-x)}{T} e^{-\frac{4a^2-4ax}{2T}}$$

$$= \frac{2(2a - x)}{T} e^{-\frac{2a(a-x)}{T}}, \quad a > 0, 0 \leq x \leq a.$$

(c) Letting $x = 0$ in the above formula for conditional density, we obtain the density of $M_{BB}(T)$, the maximum of a Brownian bridge on the interval $[0, T]$.

$$f_{M_{BB}(T)}(a) = \frac{4a}{T} e^{-\frac{2a^2}{T}}, \quad a > 0.$$

(d) The expected value of $M_{BB}(T)$ is derived as

$$\begin{aligned} E(M_{BB}(T)) &= \int_0^\infty a f_{M_{BB}(T)}(a) da = \int_0^\infty a \frac{4a}{T} e^{-\frac{2a^2}{T}} da = \left\{ z = \frac{2a}{\sqrt{T}} \right\} \\ &= \frac{\sqrt{T}}{2} \int_0^\infty z^2 e^{-\frac{z^2}{2}} dz = \frac{\sqrt{T}}{2} \cdot \frac{1}{2} \sqrt{2\pi} \int_{-\infty}^\infty \frac{z^2}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz = \frac{1}{2} \sqrt{\frac{\pi T}{2}}. \end{aligned}$$

The integral is equal to one since it is the expression for the variance of a standard normal random variable.

EXERCISE 9.10. (a) The code below plots a simulated trajectory of a two-dimensional Brownian bridge, both coordinates of which are independent Brownian bridges on the time interval $[0, T]$ where $T = (24)(60) = 1440$ minutes.

```
#defining processes as vectors
BMX<- c()
BMY<- c()
BBX<- c()
BBY<- c()

#specifying seed
set.seed(6151009)

#simulating two trajectories of Brownian motion

BMX[1]<- 0
BMY[1]<- 0

for (i in 2:1440) {
  BMX[i]<- BMX[i-1] + rnorm(1)
  BMY[i]<- BMY[i-1] + rnorm(1)
}

#computing two trajectories of Brownian bridge

for (i in 1:1440) {
  BBX[i]<- BMX[i]-i/1440*BMX[1440]
  BBY[i]<- BMY[i]-i/1440*BMY[1440]
}

#plotting two-dimensional Brownian bridge
```

```

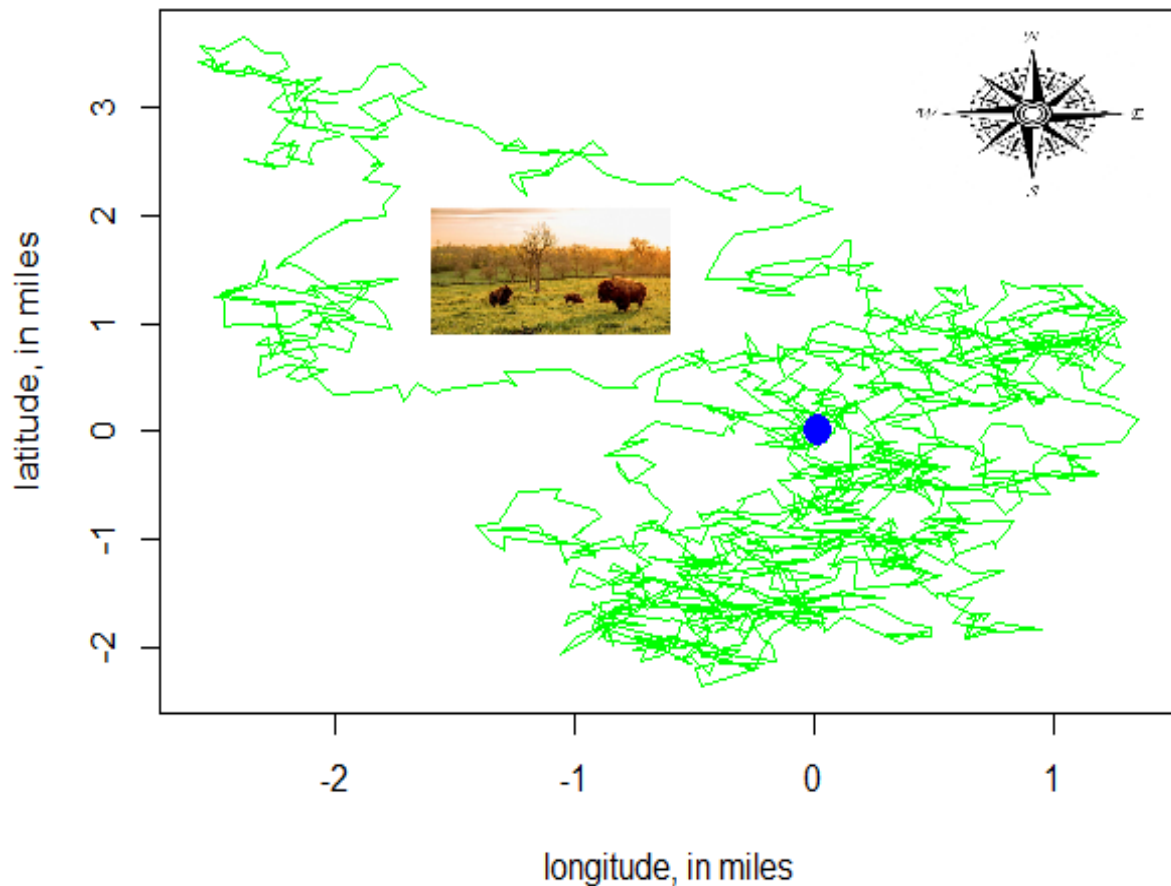
plot(BBX/10, BBY/10, type="l", col="green", xlab="longitude, in miles",
ylab="latitude, in miles")

#plotting water source in blue
points(BBX[1], BBY[1], col="blue", pch=16, cex=2)

install.packages("magick")
library(magick)
compass<- image_read("./compass.png")
bison<- image_read("./bison.png")

rasterImage(compass, 0.4, 2.1, 1.4, 3.8)
rasterImage(bison, -1.6, 0.9, -0.6, 2.1)

```



(b) From Application 9.1, we know that the expected diameter of a one-dimensional home range is $\sqrt{\frac{\pi T}{2}}$ tenths of a mile. Thus, the expected area of a two-dimensional home range is $\sqrt{\frac{\pi T}{2}} \cdot \sqrt{\frac{\pi T}{2}} = \frac{\pi T}{2} = \frac{\pi(1440)}{2} = 2261.945$ squared tenths of a mile or 22.62 squared miles.

(c) The following code simulates 1,000 trajectories and computes the sample value of the area.

```

#defining trajectories as matrices
BMX<- matrix(NA, nrow=1440, ncol=1000)
BMY<- matrix(NA, nrow=1440, ncol=1000)
BBX<- matrix(NA, nrow=1440, ncol=1000)

```

```

BBY<- matrix(NA, nrow=1440, ncol=1000)

#specifying seed
set.seed(822815)

#simulating trajectories of Brownian motion

for (j in 1:1000) {
  BMX[1,j]<- 0
  BMY[1,j]<- 0

  for (i in 2:1440) {
    BMX[i,j]<- BMX[i-1,j] + rnorm(1)
    BMY[i,j]<- BMY[i-1,j] + rnorm(1)
  }
}

#computing trajectories of Brownian bridge

for (j in 1:1000) {
  for (i in 1:1440) {
    BBX[i,j]<- BMX[i,j]-i/1440*BMX[1440,j]
    BBY[i,j]<- BMY[i,j]-i/1440*BMY[1440,j]
  }
}

#computing sample ranges
xrange<- c()
yrange<- c()

for (j in 1:1000) {
  xrange[j]<- max(BBX[,j])-min(BBX[,j])
  yrange[j]<- max(BBY[,j])-min(BBY[,j])
}

print(mean.area<- mean(xrange*yrange)/100)

```

21.91089

For this simulation, the sample value of the area is 21.91 squared miles.

EXERCISE 9.11. The following code estimates parameters and simulates a Brownian motion with drift and volatility and plots actual and simulated bird population size against time (in months).

(a) We plot the data first.

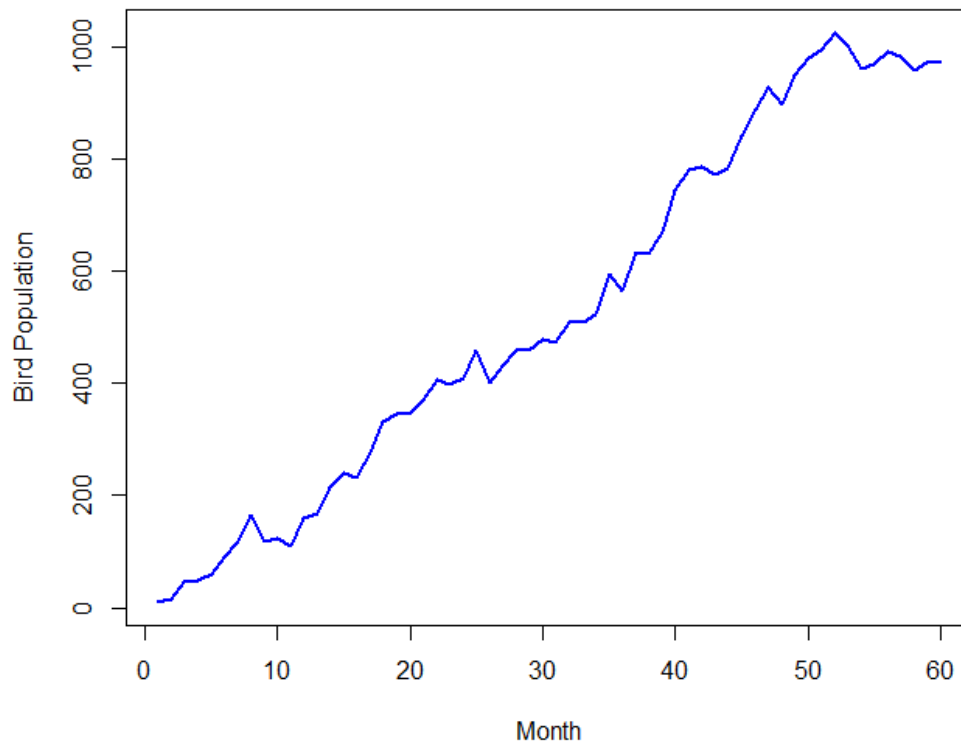
```

birds.data<- read.csv(file="./BirdPopulation.csv", header=TRUE, sep=",")

month<- birds.data$month
popl<- birds.data$population

plot(month, popl, type="l", lwd=2, cex=0, col="blue", xlab="Month",
ylab="Bird Population")

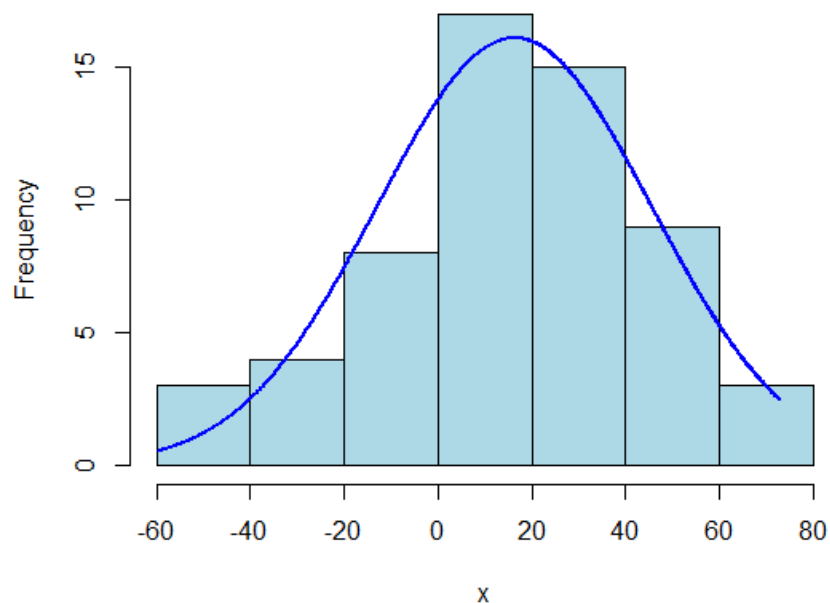
```



(b) We calculate the increments and present the histogram. The data do look normally distributed as evidenced by the bell-shaped histogram.

```
#computing increments of population size
delta.pop1<- pop1-c(0, head(pop1, -1))
delta.pop1<- delta.pop1[-1]

library(rcompanion)
plotNormalHistogram(delta.pop1, col="light blue")
```



(c) We estimated the drift and volatility by the method of moments estimators (which are the same as the maximum-likelihood estimators).


```
#estimating parameters
print(mu.hat<- mean(delta.popl))
```

16.33898

```
print(sigma.hat<- sd(delta.popl))
```

29.28611

(d) We simulate Brownian motion with drift and volatility and plot the actual and simulated trajectories on the same graph.

```
#simulating Brownian motion with drift and volatility
```

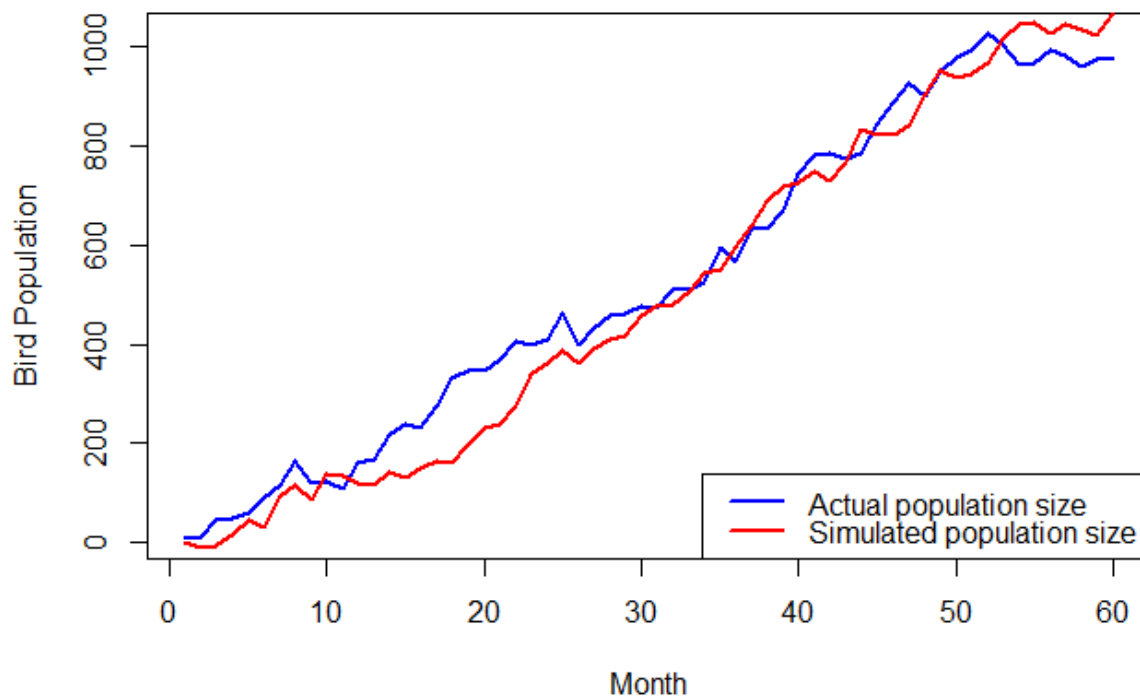
```
#specifying Brownian motion as vector
BM<- c()
```

```
#specifying initial value
BM[1]<- 0
```

```
#specifying seed
set.seed(2217626)
```

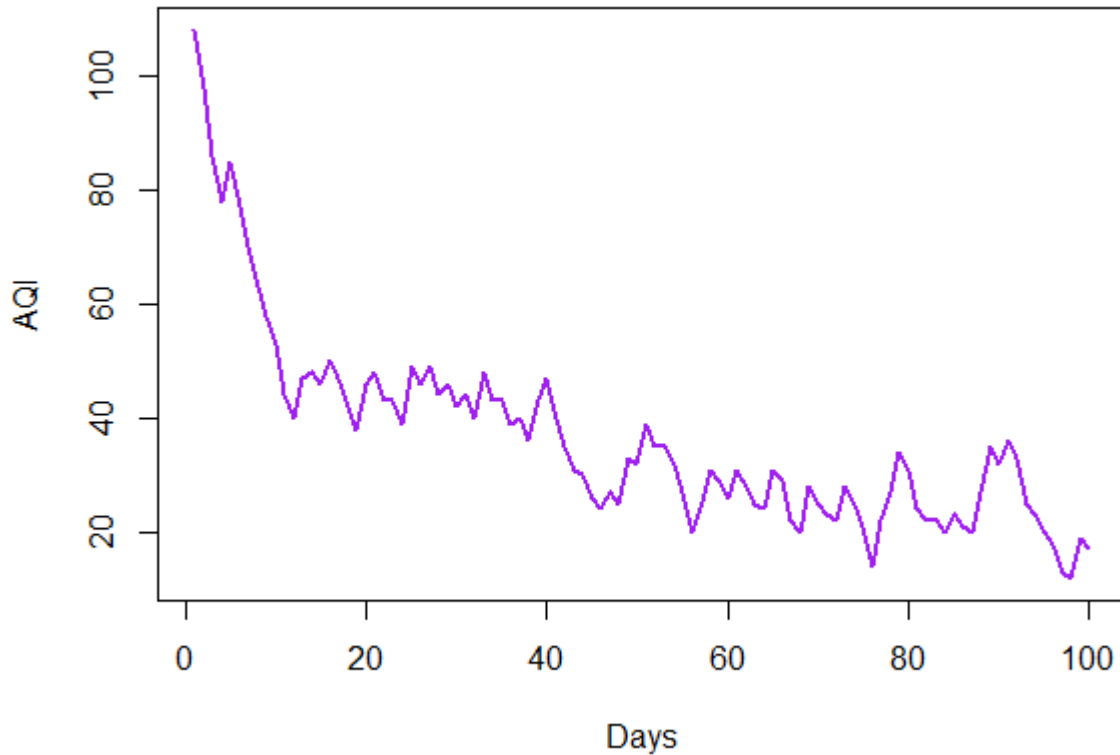
```
#simulating trajectory
for (i in 2:60)
  BM[i]<- mu.hat + BM[i-1] + sigma.hat*rnorm(1)
```

```
#plotting actual and simulated trajectories
plot(month, popl, type="l", lwd=2, cex=0, col="blue", xlab="Month",
ylab="Bird Population")
lines(month, BM, lwd=2, col="red")
legend("bottomright", c("Actual population size", "Simulated population size"),
lty=1, lwd=2, col=c("blue", "red"))
```



EXERCISE 9.12. (a) Below is the script and the plot of the data.

```
AQI.data<- read.csv(file="./AQI.csv", header=TRUE, sep=",")  
  
day<- AQI.data$day  
AQI<- AQI.data$AQI  
  
plot(day, AQI, type="l", lwd=2, cex=0, col="purple", xlab="Days", ylab="AQI")
```



We see from the graph that the values of AQI start high and then quickly decrease, suggesting that possibly a geometric Brownian motion might have a good fit.

(b) The script lines below estimate the drift and volatility coefficients of the geometric Brownian motion model.

```
#computing increments of log-AQI  
log.inc<- c()  
  
AQI1<- AQI[-1]  
AQI1.lag<- head(AQI, -1)  
log.inc<- log(AQI1/AQI1.lag)  
  
#estimating parameters  
print(mu.hat<- mean(log.inc))  
  
-0.01867594  
  
print(sigma.hat<- sd(log.inc))  
  
0.1617922
```

(c) Below we plot the actual and simulated values.

```
#simulating geometric Brownian motion

#specifying geometric Brownian motion as vector
GBM<- c()

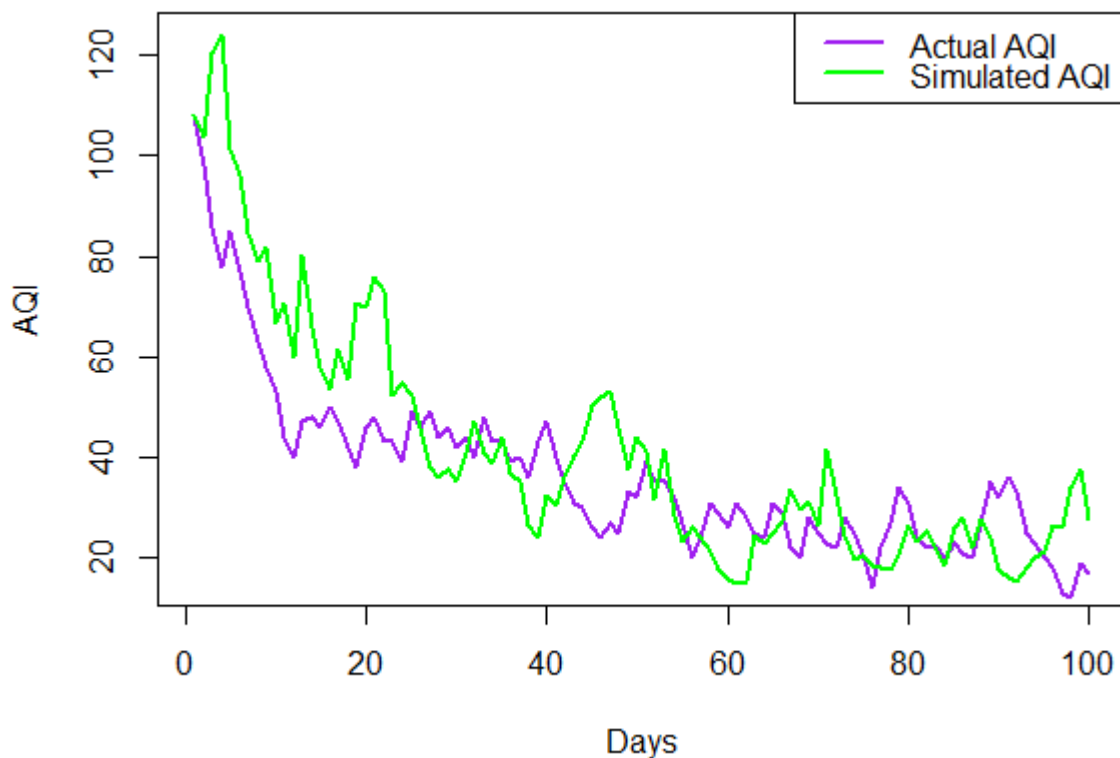
#specifying initial value
GBM[1]<- AQI[1]

#specifying seed
set.seed(34597)

#simulating trajectory
for (i in 2:100)
  GBM[i]<-GBM[i-1]*exp(mu.hat+sigma.hat*rnorm(1))

#plotting actual and simulated trajectories
plot(day, AQI, type="l", lwd=2, cex=0, col="purple", ylim=range(GBM), xlab="Days",
ylab="AQI")

lines(day, GBM, lwd=2, col="green")
legend("topright", c("Actual AQI", "Simulated AQI"), lty=1, lwd=2, col=c("purple",
"green"))
```



EXERCISE 9.13. We are given that $\mu = -0.4$, $\sigma^2 = 0.76$, $X(0) = 150$, $K = 120$, and $t = 7$.

We compute $r = \mu + \frac{\sigma^2}{2} = -0.4 + \frac{0.76}{2} = -0.02$, $A = \frac{\mu t - \ln\left(\frac{K}{X(0)}\right)}{\sigma\sqrt{t}} = \frac{(-0.4)(7) - \ln\left(\frac{120}{150}\right)}{\sqrt{(0.76)(7)}} = -1.11721$ and

$C = X(0)\Phi(A + \sigma\sqrt{t}) - e^{-rt}K\Phi(A) = (150)\Phi\left(-1.11721 + \sqrt{(0.76)(7)}\right) - e^{-(-0.02)(7)}(120)\Phi(-1.11721) = \114.21 .

EXERCISE 9.14. (a) Foreign currency exchange rates can be modeled well with an Ornstein-Uhlenbeck (OU) process because if the rates are very high, demand decreases, or, similarly, if the rates are very low, then demand increases. In both cases, the rates will revert to some long-term mean. In addition, it is reasonable to assume that the variance of exchange rates stays within certain bounds because the variance cannot grow forever in this setting.

(b) The code below fits an OU process to euro/US\$ daily exchange rate. Plots of actual and simulated trajectories are also presented.

```
exchrte.data<- read.csv(file="./Foreign_Exchange_Rates.csv", header=TRUE,
sep=",")

#estimating parameters

inc<- exchrte.data$EURO[-1]-head(exchrte.data$EURO,-1)
fit<- glm(inc ~ head(exchrte.data$EURO,-1))

theta.hat<- -fit$coefficients[2]
mu.hat<- fit$coefficients[1]/theta.hat
sigma.hat<- sigma(fit)

#simulating trajectory of OU process

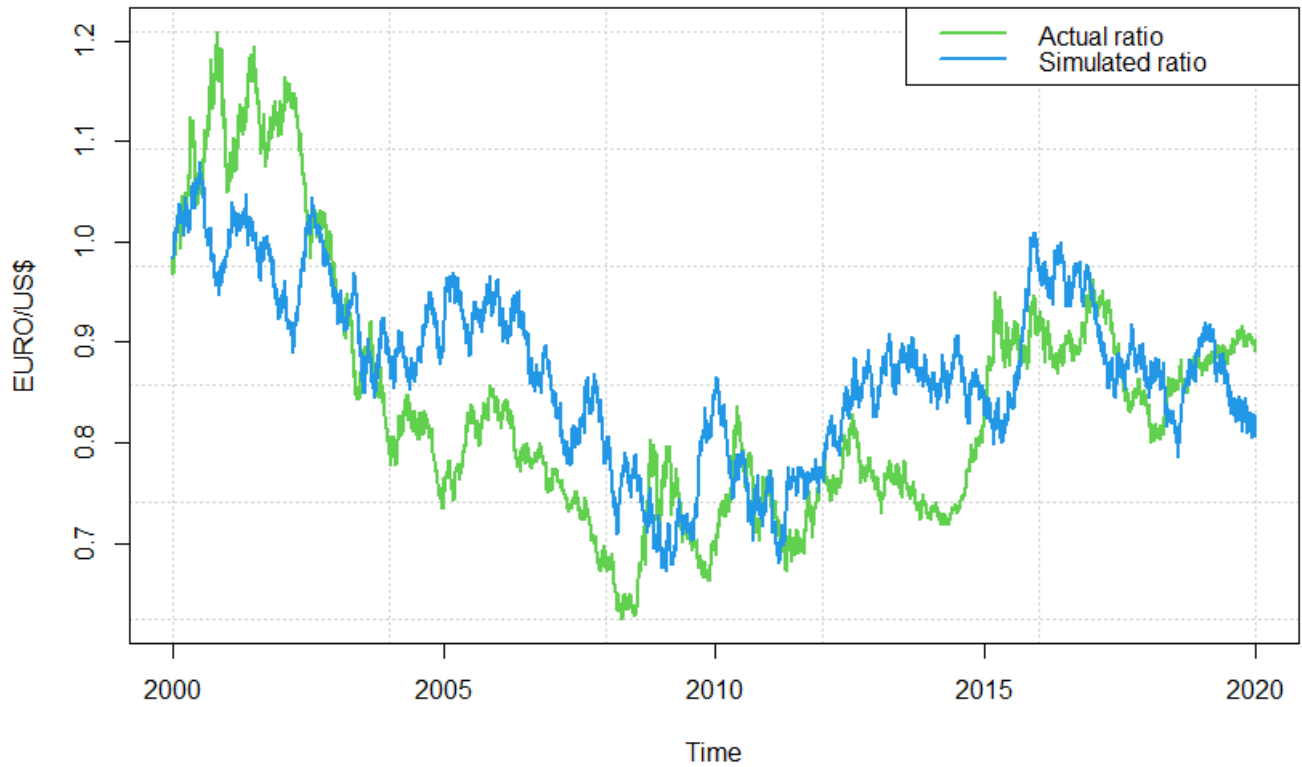
#specifying seed
set.seed(5536667)

#defining OU trajectory as vector
OU<- c()

#specifying initial value
OU[1]<- exchrte.data$EURO[1]

for (i in 2:length(exchrte.data$DATE))
  OU[i]<- OU[i-1]+theta.hat*(mu.hat-OU[i-1])+sigma.hat*rnorm(1)

#plotting trajectories
plot(as.Date(exchrte.data$DATE), exchrte.data$EURO, type="l", lty=1, lwd=2,
col=3, xlab="Time", ylab="EURO/US$", first.panel=grid())
lines(as.Date(exchrte.data$DATE), OU, lwd=2, col=4)
legend("topright", c("Actual ratio", "Simulated ratio"), lty=1, lwd=2, col=3:4)
```



EXERCISE 9.15. The process $\{B_3(t), t \geq 0\}$ has a normal distribution with independent and stationary increments since $\{B_1(t), t \geq 0\}$ and $\{B_2(t), t \geq 0\}$ have these properties. The expected value and variance of $B_3(t)$ are $E(B_3(t)) = E(\rho B_1(t) + \sqrt{1 - \rho^2} B_2(t)) = 0$, and $Var(B_3(t)) = Var(\rho B_1(t) + \sqrt{1 - \rho^2} B_2(t)) = \rho^2 Var(B_1(t)) + (1 - \rho^2) Var(B_2(t)) = \rho^2 t + (1 - \rho^2)t = t$. Thus, $\{B_3(t), t \geq 0\}$ is a standard Brownian motion.

(b) The covariance between $B_1(t)$ and $B_3(t)$ is $Cov(B_1(t), B_3(t)) = Cov(B_1(t), \rho B_1(t) + \sqrt{1 - \rho^2} B_2(t)) = \rho Var(B_1(t)) = \rho t$. The correlation coefficient is

$$\rho_{B_1(t), B_3(t)} = \frac{Cov(B_1(t), B_3(t))}{\sqrt{Var(B_1(t))} \sqrt{Var(B_3(t))}} = \frac{\rho t}{\sqrt{t} \sqrt{t}} = \rho.$$

(c) The increments $B_1(t) - B_1(s)$ and $B_2(t) - B_2(s)$ have $N(0, t - s)$ distribution, and, hence, by part (a), $B_3(t) - B_3(s)$ has the same distribution. By part (b), the increments $B_1(t) - B_1(s)$ and $B_3(t) - B_3(s)$ are correlated with the correlation coefficient ρ .

(d) We modeled the Exxon and British Petroleum (BP) historical stock prices between 4/1/2020 and 3/30/2021. The R code below plots the two processes and estimates the correlation coefficient.

```
data<- read.csv(file="./Exxon_BP_stock_prices.csv", header=TRUE, sep=",")
time<- as.Date(data$Date)
Exxon<- data$Exxon
BP<- data$BP
```

```

plot(time, Exxon, type="l", lty=1, lwd=2, col="blue", ylim=c(0,70), xlab="Time",
ylab="Stock prices", first.panel=grid())
lines(time, BP, lwd=2, col="green")
legend("bottomright", c("Exxon", "BP"), lty=1, lwd=2, col=c("blue", "green"))

```



```

Exxon.diff<-Exxon[-1]-head(Exxon,-1)
BP.diff<- BP[-1]-head(BP,-1)
cor(Exxon.diff, BP.diff)

```

0.8331595

We conclude that Exxon and BP stock prices are highly correlated with the estimated correlation coefficient of about 0.83.