160A Final Project

Dan Costello

2022-11-27

Chapter 1 Problem 1.32 R: Simulate flipping three fair coins and counting the number of heads X.

(a)
- Use your simulation to estimate P(X = 1) and E(X).

```
set.seed(1)
Coin <- c(0,1)
s <- sample(Coin, 3, replace = TRUE)
mean(s)
```

```
## [1] 0.3333333
```

I simulated coin flips with the sample function, with 1 denoting heads and 0 denoting tails. This sample outputted [0, 1, 0], indicating one heads and two tails. The estimated P(X=1) is the mean of this sample, 1/3. Estimated E(X) = Sum (X * Estimated Pr(x)) = 0 * 2/3 + 1 * 1/3 Estimated E(X) = 1/3

(b) Modify the above to allow for a biased coin where P(Heads) = 3/4.

```
set.seed(1)
Coin <- c(0,1)
Pr <- c(.25,.75)
s <- sample(Coin, 3, replace = TRUE, prob = Pr)
mean(s)
```

```
## [1] 1
```

Similar to (a), I used the sample function. This sample outputted [1, 1, 1], indicating three heads and zero tails. The estimated P(X=1) is the mean of this sample, 1. Estimated E(X) = Sum (X * Estimated Pr(x)) = 1 * 1 = 1

Chapter 2 Problem 2.23 R : Simulate the first 20 letters (vowel/consonant) of the Pushkin poem Markov chain of Example 2.2.

```
set.seed(1)
p_2 <- matrix(c(.175, .526,
                .825, .474), nrow = 2)
states <- numeric(20)
states[1] <- 1
for(t in 2:20){
  p <- p_2[states[t-1], ]
  states[t] <-  which(rmultinom(1, 1, p) == 1)}
for(i in 1:length(states)){
  if(states[i] == 1){
    states[i] <- 'Vowel'}
  else{
    states[i] <- 'Consonant'}}
states
```

```
##  [1] "Vowel"     "Consonant" "Vowel"     "Consonant" "Consonant" "Vowel"
##  [7] "Vowel"     "Vowel"     "Consonant" "Consonant" "Vowel"     "Consonant"
## [13] "Vowel"     "Consonant" "Vowel"     "Consonant" "Vowel"     "Consonant"
## [19] "Consonant" "Vowel"
```

Consider this simulation of the Pushkin poem Markov Chain. In these 20 characters, we found 10 vowels and 10 consonants. Markov used this to show that the Law of Large Numbers also applies to dependent sequences.

Chapter 3 Question 3.63 R : Hourly wind speeds in a northwestern region of Turkey are modeled by a Markov chain in Sahin and Sen (2001). Seven wind speed levels are the states of the chain.

(a) How often does the highest wind speed occur? How often does the lowest speed occur?

```
P63 <- matrix(c(
  .756, .174, .141, .003, 0, 0, 0,
  .113, .821, .001, 0, 0, 0, 0,
  .129, .004, .776, .192, .002, 0, 0,
  .002, .001, .082, .753, .227, .007, 0,
  0, 0, 0, .052, .735, .367, .053,
  0, 0, 0, 0, .036, .604, .158,
  0, 0, 0, 0, 0, .022, .789), nrow = 7)

(P63 %^% 100000)[1,7]
```

```
## [1] 0.0002953386
```

```
(P63 %^% 100000)[1,1]
```

```
## [1] 0.3245862
```

The highest wind speed occurs with proportion 0.0002953386, about .03% of the time. The lowest wind speed occurs with proportion 0.3245862, about 32% of the time.

(b) Simulate the chain for 100,000 steps and estimate the proportion of times that the chain visits each state.

```
(P63 %^% 100000)[1,]
```

```
## [1] 0.3245861739 0.2066042918 0.3039305856 0.1318890289 0.0298620155
## [6] 0.0028325658 0.0002953386
```

Speed 1 occurs with proportion 0.3245861739. Speed 2 occurs with proportion 0.2066042918. Speed 3 occurs with proportion 0.3039305856. Speed 4 occurs with proportion 0.1318890289. Speed 5 occurs with proportion 0.0298620155. Speed 6 occurs with proportion 0.0028325658. Speed 7 occurs with proportion 0.0002953386.

Chapter 4 Problem

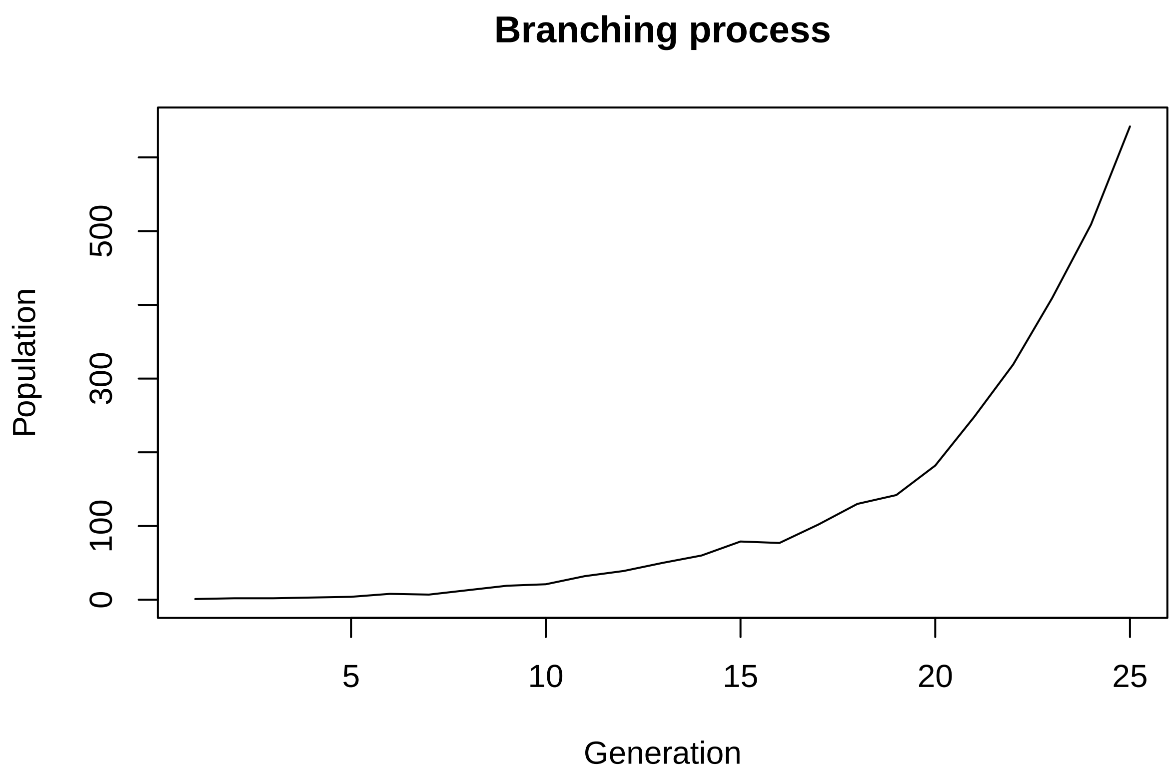
4.30 R: Simulate the branching process in Exercise 4.12. Use your simulation to estimate the extinction probability e.

Consider the following simulation and graph of the branching process with a = (1/4, 1/4, 1/2).

```
set.seed(333)
a = c(1/4, 1/4, 1/2)
mu30 = 0 * 1/4 + 1 * 1/4 + 2 * 1/2

generations <- 25
X <- matrix(0, generations, 1)
X[1] <- 1
for (n in 2:generations){
  ifelse(X[n - 1] > 0, #Tests for extinction
```

```
        X[n] <- sum(sample(0:(length(a) - 1),
                           X[n - 1], replace = TRUE, a)),
        X[n] <- 0)}
plot(X, type = "l", xlab = "Generation", ylab = "Population", main = "Branching process")
```




```
Y <- matrix(0, 100, 1)
for (i in 1:100){
  X <- matrix(0, generations, 1)
  X[1] <- 1
  for (n in 2:generations){
    ifelse(X[n - 1] > 0, #Tests for extinction
           X[n] <- sum(sample(0:(length(a) - 1),
                              X[n - 1], replace = TRUE, a)),
           X[n] <- 0)}
  if (X[25] > 0){Y[i] = 1}}
sum(Y)
```

```
## [1] 47
```

See the above graph for a simulation of this branching process. I ran this simulation for 25 generations 100 times and found 47 did not go extinct, implying extinction probability e = .53. Offspring Generating Function: G(s) = .25 + .25s + .5ss. *Solving s = G(s) = .25 + .25s + .5*ss yields roots s=.5 and s=1. Thus, extinction probability is .5, similar to .53 found by the simulation.

Chapter 5 Problems 5.18

```
set.seed(123)
```

```r
theta <- numeric(100000)
theta[1] <- 2.45
for (i in 2:100000){
  previous <- theta[i-1]
  prop <- runif(1,0,5)
  acc <- (exp(-(prop-1)^2/2) + exp(-(prop-4)^2/2)) /
    ((exp(-(previous-1)^2/2) + exp(-(previous-4)^2/2)))
  if (runif(1) < acc){
    theta[i] <- prop}
  else theta[i] <- previous
}
mean(theta)
```

```
## [1] 2.499279
```

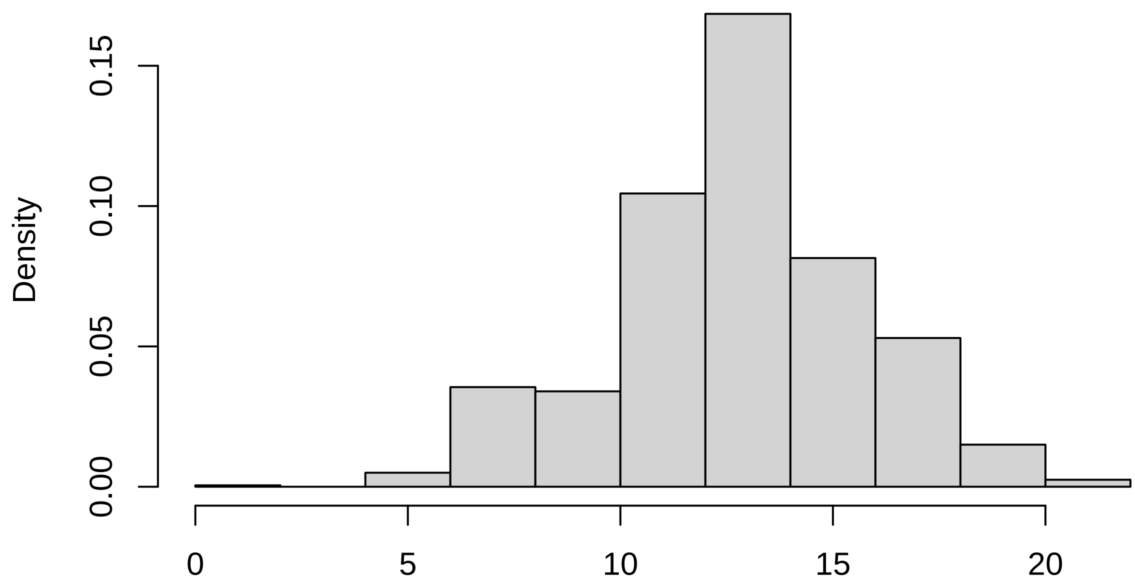
```r
var(theta)
```

```
## [1] 2.100073
```

In this question, we use a Metropolis-Hastings algorithm to sample from the given density. We utilize a proposal distribution that is uniform on 0 to 5. After simulating 100,000 units of time, we find a mean of 2.499279 and variance of 2.100073. Our approximations do a good job and come quite close to the true values of 2.5 and 2.0994.

5.19

```r
set.seed(123)
theta <- numeric(1000)
theta[1] <- 0
n <- 50
for (i in 2:1000){
  theta_star <- runif(1, min = 0, max = n)
  alpha = dbinom(as.integer(theta_star), 50, 1/4) / dbinom(as.integer(theta[i - 1]), 50, 1/4)
  if (runif(1) < alpha) theta[i] <- theta_star
  else theta[i] <- theta[i - 1]
}
hist(theta,xlab="",main="",prob=T)
```




```
res = 0
for (i in 1:1000) {
  if(10 <= theta[i] & theta[i] <= 15){res = res + 1}
}
res/1000
```

```
## [1] 0.64
```

```
sum(dbinom(10:15, 50, 1/4))
```

```
## [1] 0.6732328
```

In this question, we use a Metropolis-Hastings algorithm to sample from a binomial distribution with parameters n=50 and p=1/4. We utilize a proposal distribution that is uniform on 0 to n. After simulating 1,000 units of time, we find that our sample exists in [10, 15] 64% of the time. This estimates that P(X in [10,15]) is approximately 0.64. Our approximation of 0.64 is similar to the true value, 0.6732328.

5.20

```
set.seed(12)
p <- 1/3
theta <- numeric(1000)
for (i in 2:1000){
  state <- 0
  for (j in 1:50){
    y <- rgeom(1, p) + 1
    acc <- 3^y * factorial(state)/(3^state * factorial(y)) * (1-p)^(state-y)
    if (runif(1) < acc){state <- y}
  }
```

```r
  theta[i] <- state
}
mean(theta)
```

```
## [1] 3.073
```

```r
var(theta)
```

```
## [1] 2.578249
```

In this question, we implement a MCMC algorithm to simulate a Poisson distribution with parameter lambda = 3. We utilize a proposal distribution that is geometric with success probability of 1/3. After simulating 50 units of time, we consider the string to be near stationary. Simulating this 1,000 times yields a mean estimate of 3.073 and variance estimate of 2.578249. Our approximation of 3.073 and 2.578249 are similar to the true values, 3 and 3.