HW 2

Jeff Gould

9/10/2020

2) Let $Y \sim \mathcal{N}(0, \Sigma)$ with $Y \in \mathbb{R}^n$. Let M be an invertible $n \times n$ matrix. Show that $MY \sim \mathcal{N}(0, M\Sigma M^T)$. Don't assume that MY is normal. (I did this problem in class, but left the very end for you to do.)

$P(MY \in R - P(Y \in M^{-1}R) = \int \cdots \int_{M^{-1}R} dy_1 \dots dy_n \frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} e^{-Y^T \Sigma^{-1} Y/2}$

Let $z = h(y) \quad h : \mathbb{R}^n \to \mathbb{R}^n$, $y = h^{-1}(z) \Rightarrow$

$= \int \cdots \int_{h(M^{-1}R)} dz_1 \dots dz_n \frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} e^{-h^{-1}(z)\Sigma^{-1}h^{-1}(z)/2} \Rightarrow z = MY \leftrightarrow Y = M^{-1}z$

$P(MY \in R) = \int \cdots \int_R dz_1 \dots dz_n \frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} \det M^{-1} e^{-(M^{-1}z)^T \Sigma^{-1}(M^{-1}z)/2}$

$\det M^{-1} = \frac{1}{\det M} = \frac{1}{\det M^{1/2} \det M^{1/2}} = \frac{1}{\det M^{1/2} \det M^{T^{1/2}}} \Rightarrow$

$\frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} \det M^{-1} = \frac{1}{(2\pi)^{n/2}(\det M \det \Sigma \det M^T)^{1/2}} = \frac{1}{(2\pi)^{n/2}(\det(M\Sigma M^T)^{1/2}}$

$(M^{-1}z)^T \Sigma^{-1}(M^{-1}z) = (z^T M^T)^{-1}\Sigma^{-1}M^{-1}z = z^T(M^{T^{-1}}\Sigma^{-1}M^{-1})z = z^T(M^T \Sigma M)^{-1}z \Rightarrow e^{-(M^{-1}z)^T \Sigma^{-1}(M^{-1}z)/2} = e^{-z^T(M\Sigma M^T)^{-1}z/2}$

Therefore

$P(MY \in R) = \int \cdots \int_R dz_1 \dots dz_n \frac{1}{(2\pi)^{n/2}(\det \Sigma)^{1/2}} \det M^{-1} e^{-(M^{-1}z)^T \Sigma^{-1}(M^{-1}z)/2} = \int \cdots \int_R dz_1 \dots dz_n \frac{1}{(2\pi)^{n/2}(\det(M\Sigma M^T)^{1/2}} e^{-z^T}$

And this is the pdf for a multivariate normal with $\mu = 0$ and a variance matrix of $(M\Sigma M^T)$

$MY \sim \mathcal{N}(0, M\Sigma M^T)$

3) Let X be an exponential r.v. with rate 1. Using cdf inversion, write a function that generates n independent samples of X (we discussed this example in class, but you should do the cdf inversion yourself, not just quote our result). Compare the speed of your sampler for $n = 10^6$ with that of your language's exponential sampler (in R rexp).

for a general exponential r.v. Y, $f(y) = \lambda e^{-\lambda y} \to F(y) = 1 - e^{-\lambda y}$. Since we are given $\lambda = 1$ above, $f(x) = e^{-x} \to F(x) = 1 - e^{-x}$, for $x \geq 0$

$y = F(x) \to y = 1 - e^{-x} \to e^{-x} = 1 - y \to -x = \ln(1-y) \to x = -\ln(1-y)$

So $x = F^{-1}(y) = -\ln(1-y)$, and $y < 1$, as $\ln(1-y)$ is undefined if $1 - y \leq 0$, and $y > 0$ as $1 - e^{-x} < 1 \;\; \forall x \geq 0$

So let $Y \sim Unif(0,1)$, then $x = F_X^{-1}(y) = -\ln(1-y)$ has an exponential distribution

```
rand_exp <- function(n){

  Y <- runif(n)
  X <- -log(1-Y)
  return(X)
```

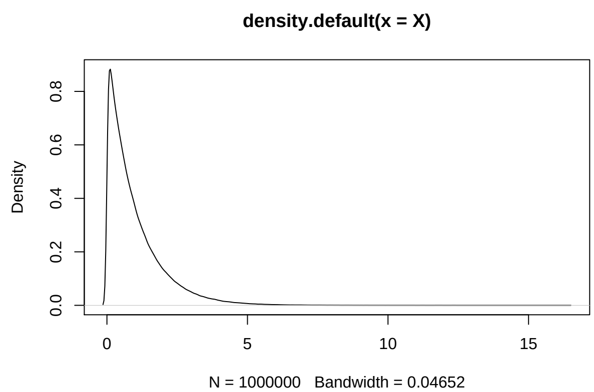
```
}
set.seed(123)
tictoc::tic()
X <- rand_exp(10^6)
tictoc::toc()
```

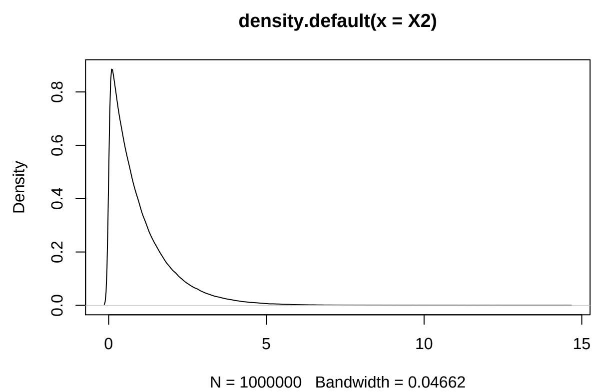
```
## 0.043 sec elapsed
```

```
set.seed(123)
tictoc::tic()
X2 <- rexp(10^6)
tictoc::toc()
```

```
## 0.044 sec elapsed
```

```
plot(density(X))
plot(density(X2))
```



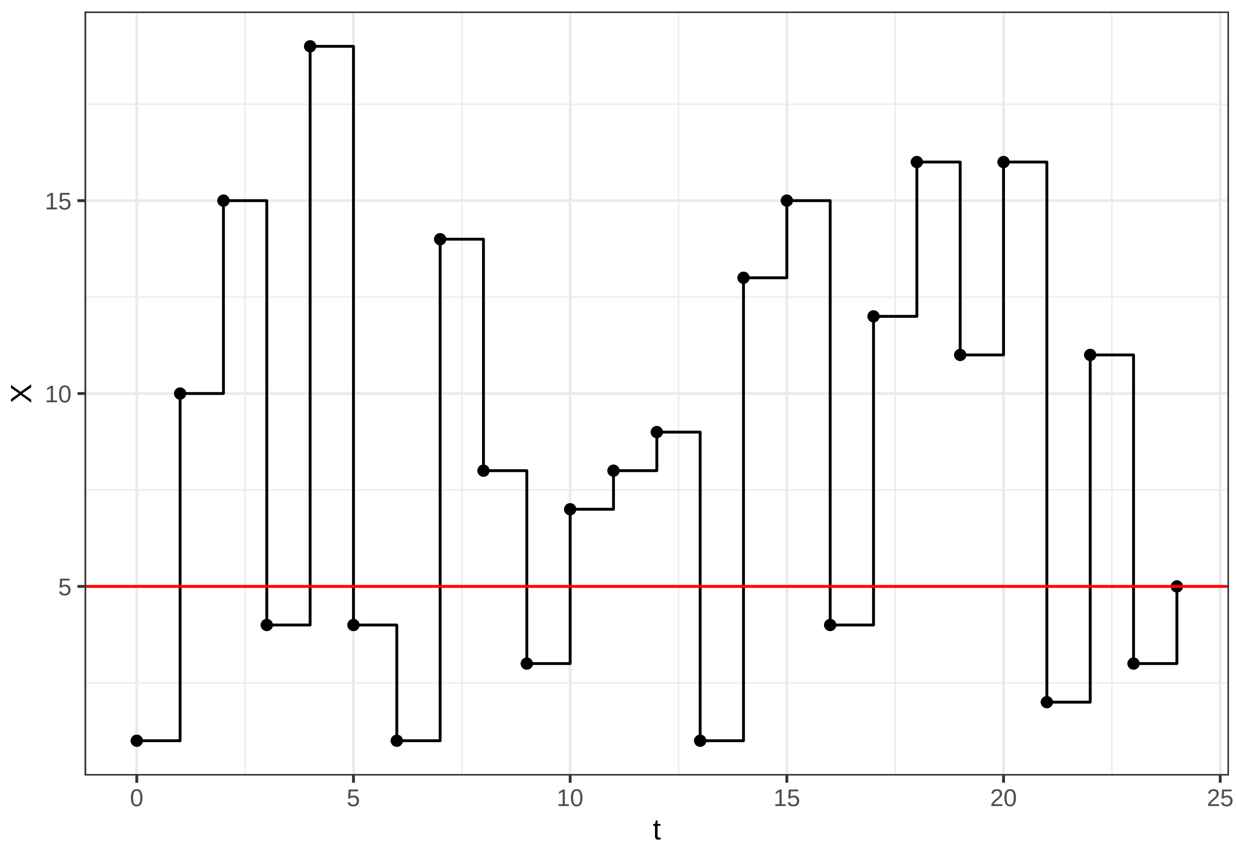


We find very similar runtimes for both the self-made function, drawing from the Unifrom distribution and transforming to an exponential r.v., vs using the system's built in `rexp` sampler. We also see what appears to be identical density plots

4) Write a function `MarkovChain(P, s_0, s)` that simulates a Markov chain $X(t)$ until the first time the chain is in state s, assuming $X(0) = s_0$. The function should return the path of the chain from $t = 0$ to when it "hits" state s. You may use your language's discrete sampler (in R sample) or write your own.

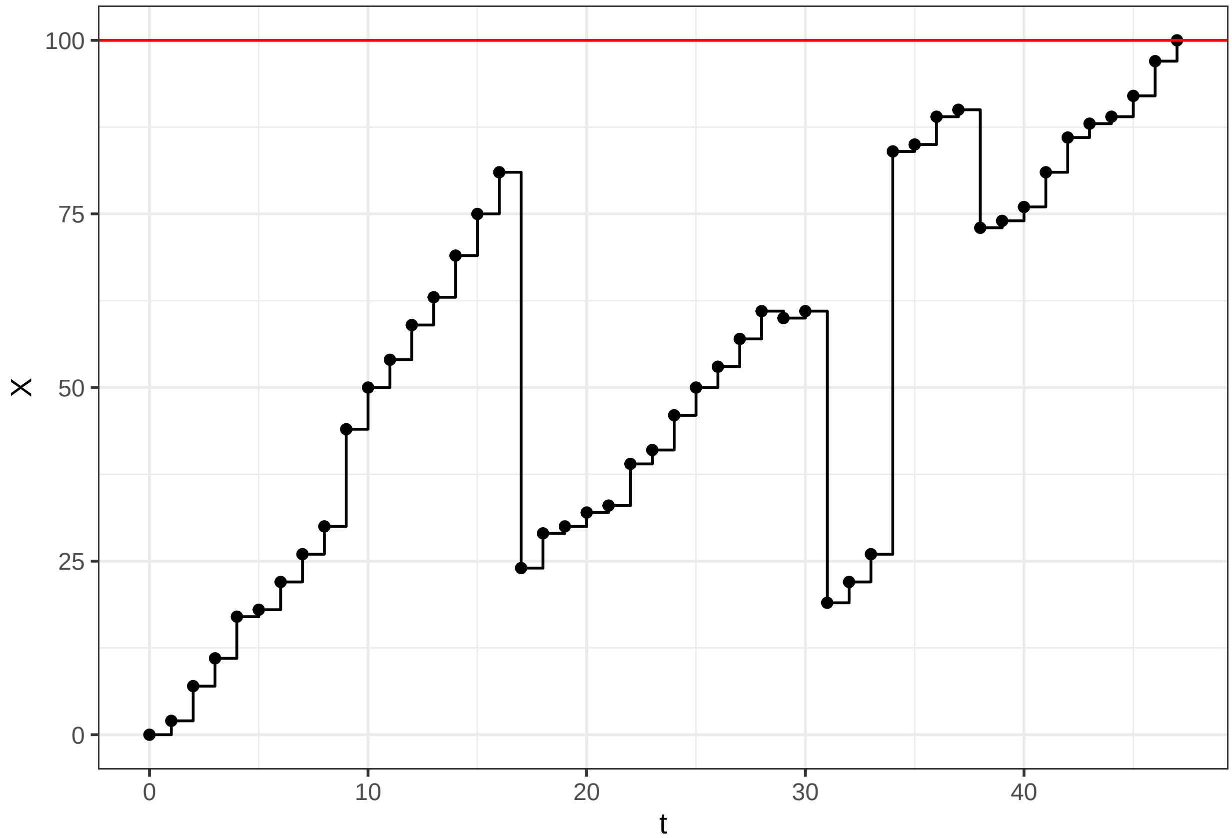
```
MarkovChain <- function(P, s_0, s){
  if(any(rowSums(P)< 0.999)  | any(rowSums(P)> 1.001) |nrow(P) != ncol(P))return(print("Please Give A Va

  state_spaces <- seq(1,nrow(P))
  X <- c(s_0)
  S <- s_0

  while(S != s){
    S <- sample(state_spaces, size = 1, prob = P[S, ])
```

```r
    X <- c(X,S)
  }

  MC <- data.frame(
    X = X,
    t = seq(0, length(X) - 1, 1)
  )
  return(MC)
}

n <- 20
P <- matrix(data <- rep(1/n, n^2), nrow = n)
set.seed(25)
X <- MarkovChain(P = P, 1, 5)

ggplot(X, aes(x = t, y = X)) +
  geom_step() +
  geom_point() +
  theme_bw() +
  geom_hline(color = "red", yintercept = 5)
```



5) Chutes & Ladders

```r
### First, make probability transition matrix. this will be 101 x 101, as we start at square 0, and the


P <- t(sapply(X = c(0:100), FUN = function(x){
  if(x == 0){
    return(c(0, rep(1/6, 6), rep(0, 94)))
  }else if(x <= 94){
    return(c(rep(0, x+1), rep(1/6, 6), rep(0, 100 - x - 6)))
  }else if(x < 100){
    return(c(rep(0,x+1), rep(1/6, 100-x-1), (6 - (100-x-1)) / 6))
  } else{
    return(c(rep(0,100), 1))
  }
}))

chutes_ladders <- read_csv(file = "~/Desktop/Math-611/HW 2/chutes_and_ladder_locations.csv")


for (i in 1:nrow(chutes_ladders)) {
  location <- chutes_ladders$start[i]
  new_end <- chutes_ladders$end[i]

  P_events <- which(P[,location + 1] != 0)
  probs <- P[P_events, location + 1]

  P[P_events, new_end + 1] <- probs + P[P_events, new_end + 1]
  P[P_events, location + 1] <- 0

}
set.seed(69)
ChutesAndLadders <- MarkovChain(P = P, 1, 101) %>%
  mutate(X = X - 1)

ggplot(ChutesAndLadders, aes(x = t, y = X)) +
  geom_step() +
  geom_point() +
  theme_bw() +
  geom_hline(color = "red", yintercept = 100)
```



Monte Carlo error:

$error = \left(\frac{1}{N}\sum_{i=1}^{N}\hat{L}^{(i)} - E[L]\right)$

central limit theorem: If X is a r.v. with variance equal to σ^2. then $\lim_{N\to\infty}\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\hat{X}^{(i)} - E[X]\right) = N(0,\sigma^2) \Rightarrow \sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\hat{X}^{(i)} - E[X]\right) \sim N(0,\sigma^2)$

From here, we can set a 95% confidence interval for $E[X]$, with bounds of 5 units above/below:

$P(\frac{1}{N}\sum X^{(i)} - 5 < E[X] < \frac{1}{N}\sum X^{(i)} + 5) = 0.95 \to P(-\frac{\sqrt{N}}{\sigma}5 < N(0,1) < \frac{\sqrt{N}}{\sigma}5) = 0.95$

This gives $\frac{\sqrt{N}}{\sigma}5 = 1.96 \to N = \left(\frac{1.96}{5}\right)^2\sigma^2 = 0.154\sigma^2$

However, we don't know σ^2 ahead of time.

We run the simulation $N = 1,000$ times in the chunk below. we find $\sigma^2 = 546.7$. So plugging that into our equation above, we would have 95% confidence that our mean was within 5 turns of the true average after $0.154(546.99) = 84.23646$ turns

```
set.seed(420)
Chutes <- lapply(rep(1,1000), MarkovChain, P = P, s = 101) %>%
  bind_rows()

last_turns <- Chutes %>% filter(X == 101) %>%
  mutate(avgTurns = cumsum(t) / row_number(), ### for some reason cummean is counting the first row twi
         simNumber = row_number())

sd(last_turns$t[1:1000])^2
```

```
## [1] 546.6999
```

```r
ggplot(last_turns, aes(x = simNumber, y = avgTurns)) +
  geom_line() +
  theme_bw() +
  geom_hline(yintercept = c(mean(last_turns$t - 5), mean(last_turns$t + 5)), color = "red")
```