Homework 1- PCS

Elena Volpi

10/3/2021

#Problem 2 Prompt: Write a computer program that, when given a probability mass function $\{p_j, j = 1, ..., n\}$ as an input, gives as an output the value of a random variable having this mass function.

```
#Given a vector p_j = P(X = j) of length n,
#check if u < p_0 or p_j[1]  -> x=1
#else if u < sum(p_j) ->
sim <- function(p_j) {
  n <- length(p_j)   #grab the length of the vector
  u <- runif(1)     #Generate a random number u~ unif(0,1)
  if(u < p_j[1]) {
    x <- 1
    return(x)
  }
  for(i in 2:n) {
    if(u < sum(p_j[1:i])) {
      x <- i
      return(x)
    }
  }
}

p_j <- c(0.2,0.15,0.25,0.4) #Example pj from Example 4a
sim(p_j)
```

```
## [1] 2
```

```
#We could also sort in order of greatest probability.
```

#Problem 4 Prompt: A deck of 100 cards - numbered 1,2,..,100 - is shuffled and then turned over one card at a time. Say that a "hit" occurs whenever card i is the ith card to be turned over, $i = 1, ..., 100$. Write a simulation program to estimate the expectation and variance of the total number of hits. Run the program. Find the exact answer and compare them with your estimates.

```
hit_num <- function(X){
  base <- seq(1,length(X))
  return(sum(X==base))
}

sim_hits <- function(n, nsim) {
  #outcomes <-map(1:nsim, ~sample(n))
```

```
  outcomes <- matrix(nrow=nsim,ncol=n)
  hits <- rep(0,nsim)
  for(i in 1:nsim) {
   outcomes[i,] <- sample(n)
    hits[i] <- hit_num(outcomes[i,])
  }
  print("The estimated mean is:")
  print(mean(hits))
  print("The estimated variance is", )
  print(var(hits))
}

sim_hits(50000,100)
```

```
## [1] "The estimated mean is:"
## [1] 1.12
## [1] "The estimated variance is"
## [1] 1.03596
```

Assume we have n cards, $X = (X_i, ..., X_n)$ that we are sampling without replacement. For $i \epsilon (1, ..., n)$, let H_i denote the position in the deck where H_i =1 indicates a "hit", or card i is the ith card to be turned over and H_i =0 indicates a "miss". Thus, $H_i \sim$ Bernoulli(p). Since i is fixed, we have $(n-1)!$ possibilities for the remaining i-1 cards and $n!$ outcomes for card i. Then the probability of a hit is as follows

$$P(H_i = 1) = \frac{(n-1)!}{n!} = \frac{1}{n}$$

Let Y denote the numbers of hits in a deck. It follows

$$Y \epsilon (0, 1, ..., n)$$

and Y $= H_1 + H_2 + ... + H_n$. Then,

$$E(Y) = E(\sum_{i=1}^{n} H_i) = \sum_{i=1}^{n} E(H_i) = \sum_{i=1}^{n} (\frac{1}{n}) = n(\frac{1}{n}) = 1$$

Furthermore,

$$E(Y^2) = E(\sum_{i=1}^{n}\sum_{j=1}^{n} H_i H_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} E(H_i H_j)$$

. We first consider the case of when $i = j$, $H_i H_j = H_i^2 = H_i,\ H_i\ \epsilon (0, 1$. When $i \neq j$,

$$H_i H_j = \begin{cases} 0, \text{ if i, j or both are misses} \\ 1, \text{ if i and j are both hits} \end{cases}$$

.

Thus, $H_i H_j \sim$ Bernoulli(p) and

$$E(H_i H_j) = \begin{cases} \frac{1}{n} \text{ if i=j} \\ \frac{1}{n(n-1)} \text{ if i} \neq \text{j} \end{cases}$$

.

Then,

$$E(Y^2) = \sum_{i=1}^{n}\sum_{j=1}^{n} (\frac{1}{n}) + \sum_{i=1}^{n}\sum_{j=1}^{n} (\frac{1}{n(1-n)}) = n(\frac{1}{n}) + 2(\frac{n(n-1)}{2})(\frac{1}{n(n-1)}) = 1 + 1 = 2$$

Moreover, $Var(Y) = E(Y^2) - E(Y)^2 = 2 - 1 = 1$.

Our simulation gave us estimated mean of 1 and estimated variance of 1.03. Other trials of our simulation produced similar results. We can say that our simulation can reasonably estimate our exact solution.

#Problem 7 Prompt: A pair of fair dice are to be continually rolled until all the possible outcomes 2,3,. . . ,12 have occurred at least once. Develop a simulation study to estimate the expected number of dice rolls that are needed.

```
dice_sim <- function() {
  current <- rep(0,11) #create a vector of zeros.
  counter_loop <- 0
  while(any(current ==0 )) {  #our vector is the size of all outcomes, we loop until the vector is non-
    dice <- sample(1:6,2, replace = TRUE) #had every possible outcome.

    dice_sum <- sum(dice)

    if((dice_sum %in% current) == FALSE) {  #If we haven't collected this sum we place it in our at ind
       current[dice_sum -1] <- dice_sum     #At index 1, we want 2 since we cant have a sum of 1
    }
    counter_loop <- counter_loop +1
  }
 counter_loop -1
}

Rolls_expected <- function(nsim) {
  rolls <- vector(length = nsim)
  for(i in 1:nsim) {
    rolls[i] <- dice_sim()
  }
  mean(rolls)
}
Rolls_expected(10000)
```

```
## [1] 60.4524
```

#Problem 13 Prompt: Give two methods for generating a random variable X such that:

$$P(X = i) = \frac{e^{-\lambda}\lambda^i/i!}{\sum_{j=0}^{k} e^{-\lambda}\lambda^j/j!}, i = 0, ..., k$$

First, the acception rejection method:

```
gen_x_poisson<- function(lambda) {
  #Generate u ~Unif(0,1)
  u <- runif(1)
  p <- exp(-lambda)      #initial value as defined in lecture
  f <- p #cdf value
  not_sure <- 1000
  for(i in 0:not_sure ) {    #Not sure what the bound of this loop should be?
    if(u < f) {
      x <- i
      return(x)
   } else {
```

```r
      p <- (lambda/(i+1))*p
      f <- f+ p
    }
  }
  x
}
```

We will have

$$p_i = \frac{e^{-\lambda}\lambda^i/i!}{\sum_{j=0}^{k} e^{-\lambda}\lambda^j/j!}$$

for

$$i \ \epsilon \ [0, k]$$

and

$$q_i = e^{-\lambda}\lambda^i/i!$$

. Also, let $c = \frac{1}{\sum_{j=0}^{k} e^{-\lambda}\lambda^j/j!}$. Then,

$$\frac{p_i}{cq_i} = \begin{cases} 1 & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases}$$

So,

```r
ajm_sim <- function(k,lambda){
  X <- gen_x_poisson(lambda)
  #print(X)
  while(X >k) {
    X <- gen_x_poisson(lambda)
    #print(X)
  }
  return(X)
}

ajm_sim(10,3)  #Doesn't seem to be working right
```

```
## [1] 2
```

Inverse transform Method: