

Lab 3 Solution

Harsh Agrawalla

Problem 1 Use the Acceptance–Rejection method to generate samples from $\text{Binomial}(n, p)$ setting Geometric as the proposal distribution.

- (1) For $(n, p) = (10, 0.3)$ use $\text{Geometric}(p)$ as the proposal.
- (2) Estimate the value of c .
- (3) For $(n, p) = (100, 0.3)$ use $\text{Geometric}(p^*)$ as the proposal with appropriate choice of p^* .
- (4) Estimate the value of c and compare it with the value of c when the proposal is set as $\text{Geometric}(p)$.

Solution:

```
## Function to generate binomial distribution from geometric distribution
## using Acceptance-rejection Method

generate_binomial_geometric <- function(n, p, c, p_star){

  # Count to store the number of times required for one acceptance
  count <- 0

  # Checking for acceptance
  while(TRUE){

    # Generating a random sample from Geometric(p*)
    geom_sample <- rgeom(1, p_star)

    # Generating a random sample from Uniform(0,1)
    u <- runif(1)

    # Incrementing the number of times the loop ran
    count <- count + 1

    # Checking for condition
```

```

    if (u <= dbinom(geom_sample, n, p)/(c * dgeom(geom_sample, p_star))){
      # Returning the corresponding sample and no. of times the loop ran
      return(c(geom_sample, count))
    }
  }
}

```

The function defined in the above code takes four parameters namely n required for generating the binomial sample, p the probability of success in case of the Binomial distribution, c the expected number of times the values are generated before it is accepted and p^* (p_star) the probability of success of the proposed Geometric distribution. In this case the target distribution is Binomial (n, p) and the proposed distribution is Geometric (p^*).

```

# Function to estimate the value of c theoretically
estimate_c <- function(n, p){
  k_vals <- 0:n

  binomial_pmf <- dbinom(k_vals, n, p)
  geometric_pmf <- dgeom(k_vals, p)

  # Calculating the theoretical value of c required in the ARM method
  c <- max(binomial_pmf/geometric_pmf)

  # Returning the theoretically obtained value
  return(c)
}

```

The function defined in the above code takes two parameters namely n and p and returns the theoretically expected value of c for the given parameters n and p .

```

## Part (1) Geometric(10,0.3)
n1 <- 10
p1 <- 0.3

# Theoretically estimating value of c for n = 10 and p = 0.3
c1 <- estimate_c(n1,p1)

# Theoretically estimated value of c for n = 10 and p = 0.3
c1

```

```
[1] 2.7783
```

```

# Array to store the generated samples
binom_samples <- numeric(1e4)

# Array to store the c values obtained
estimate_c_array <- numeric(1e4)

# Loop to call the function iteratively to generate samples
for (i in 1:1e4){
  ele <- generate_binomial_geometric(n1, p1, c1, p1)
  binom_samples[i] <- ele[1]
  estimate_c_array[i] <- ele[2]
}

mean(binom_samples)

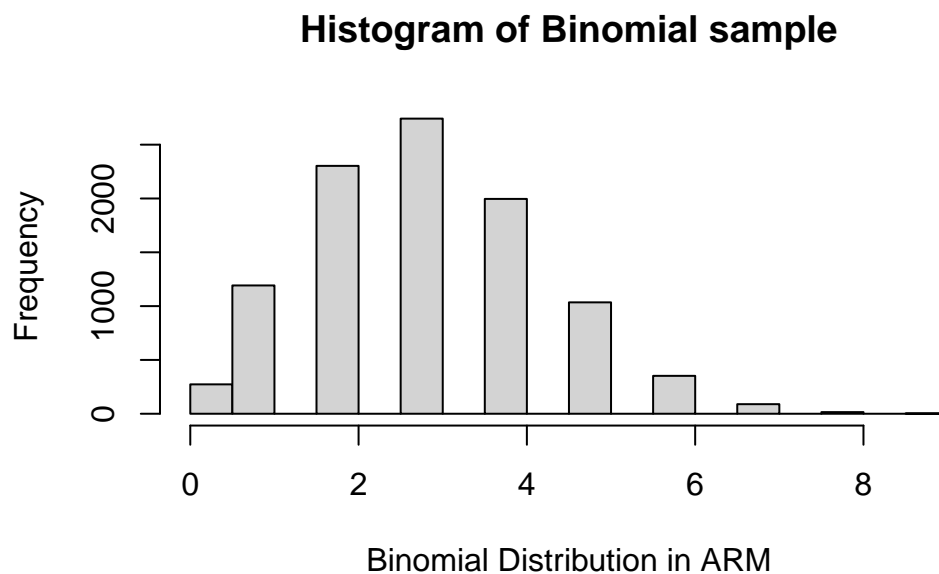
```

```
[1] 3.0065
```

```

hist(binom_samples,xlab = "Binomial Distribution in ARM",
      main = "Histogram of Binomial sample")

```



The above code calls the required functions to get the theoretically expected value of c and generated 10^4 samples from $\text{Binomial}(10, 0.3)$ with $\text{Geometric}(0.3)$ as the proposed the distribu-

tion. On checking the mean of the generated sample, we get the mean of the generated sample close to the theoretically expected mean of Binomial(10,0.3) which is $n * p = 10 * 0.3 = 3$.

```
## Part (2) Estimating the value of c
```

```
# Finding the mean of the array  
mean(estimate_c_array)
```

```
[1] 2.751
```

Since, the estimate_c_array stored the experimentally obtained values of c , taking the mean of the array gives us the experimentally estimated value of c , which is quite close to its theoretically obtained value.

```
## Part (3) Geometric(100,0.3) from Geometric(p*)
```

```
n2 = 100  
p2 = 0.3
```

```
# Value of p* = 1/(1 + np)  
p_star = 1/(n2*p2 + 1)
```

```
# Theoretically estimating value of c for n = 100 and p = p*  
c_star <- estimate_c(n2, p_star)
```

```
# Theoretically estimated value of c for n = 100 and p = p*  
c_star
```

```
[1] 7.715878
```

```
# Array to store the generated samples  
binomial_samples <- numeric(1e4)
```

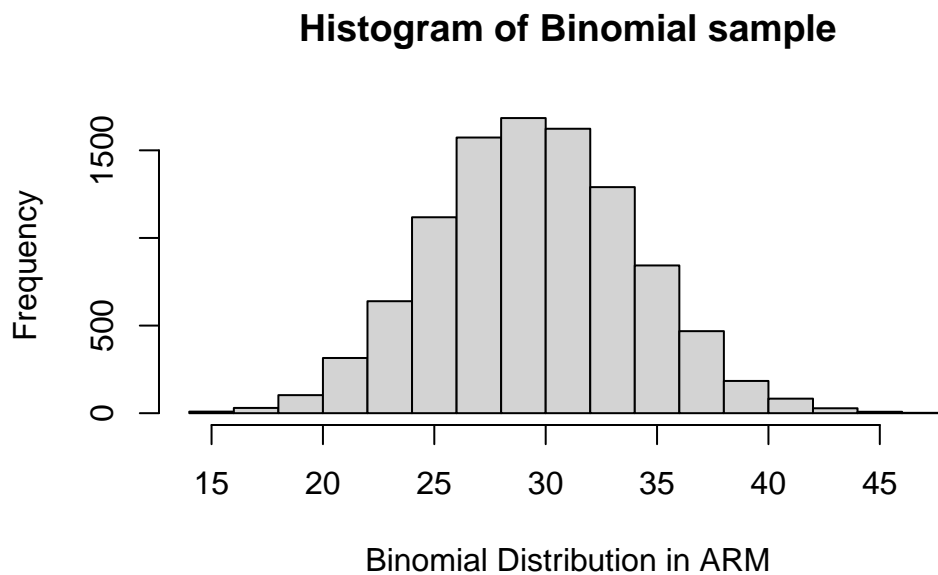
```
# Array to store the c values obtained  
estimate_c_star_array <- numeric(1e4)
```

```
# Loop to call the function iteratively to generate samples  
for (i in 1:1e4){  
  ele <- generate_binomial_geometric(n2, p2, c_star, p_star)  
  binomial_samples[i] <- ele[1]  
  estimate_c_star_array[i] <- ele[2]  
}
```

```
}
mean(binomial_samples)
```

```
[1] 30.0082
```

```
hist(binomial_samples,xlab = "Binomial Distribution in ARM",
      main = "Histogram of Binomial sample")
```



The above code calls the required functions to get the theoretically expected value of c and generated 10^4 samples from $\text{Binomial}(100, 0.3)$ with $\text{Geometric}(p^*)$ as the proposed the distribution. On checking the mean of the generated sample, we get the mean of the generated sample close to the theoretically expected mean of $\text{Binomial}(100, 0.3)$ which is $n \cdot p = 100 \cdot 0.3 = 30$.

```
## Part (4) Estimating the value of c* thus obtained and
## comparing it when samples are generated using Geometric(p)

# Finding the mean of the array
mean(estimate_c_star_array)
```

```
[1] 7.7725
```

```
# Theoretically estimating the value of c when  
# success probability of Geometric distribution. is p = 0.3  
c2 <- estimate_c(n2, p2)  
c2
```

```
[1] 48913.21
```

From the values obtained above we can see that if the success probability of Geometric (p) is significant, then it takes a large amount of tries for the acceptance of even a single value and hence scaling the success probability from p to p^* which helps reduce the computations significantly.