Lab 4 Solution

Utkarsh Kesharwani

Problem 1 :

To generate Gamma(4,5) from an appropriate proposal distribution using AR method.

```r
## Function to generate gamma distribution from exponential distribution
## using Acceptance-rejection Method

generate_gamma_exponential <- function(alpha, beta, lambda, c){

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

  # Checking for acceptance
  while(TRUE){

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

    # Using ITM to generate a random sample from Exponential(lambda)
    expo_sample <- -log(u_star)/lambda

    # 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 <= dgamma(expo_sample,alpha,beta)/(c*dexp(expo_sample,lambda))){
      # Returning the corresponding sample and no. of times the loop ran
      return(c(expo_sample, count))
    }
```

```
  }
}
```

The function defined in the above code takes four parameters namely α and β, the shape and scale parameters of the Gamma distribution, λ the rate parameter of the exponential distribution and c, denoting the expected number of times the loop ran before an acceptance.

Here, our target distribution is Gamma(4,5) and the proposal distribution is Exponential (λ), with suitable choice of λ.

```
# The parameters
n <- 1e4
alpha <- 4
beta <- 5

# To mininmize the number of loops, we choose the parameter of the proposal
# ... such that computations become easier
lambda <- beta/alpha

# The theoretical value of c
theoretical_c <- (alpha^alpha)*(exp(1 - alpha))/gamma(alpha)
```

We know that to minimize the computations, we choose c such that $c = max(\frac{\beta^\alpha x^{\alpha-1} e^{-x(\beta-\lambda)}}{\lambda \Gamma(\alpha)})$ for $x > 0$. On solving it we get that c takes its maximum value at $x = \frac{\alpha-1}{\beta-\lambda}$, which it turn takes the minimum value at $\lambda = \frac{\beta}{\alpha}$. On, substituting the required values, we get the theoretical value of $c = \frac{\alpha^\alpha e^{1-\alpha}}{\Gamma(\alpha)}$.

```
# Array to store the generated sample from gamma distribution
gamma_array <- numeric(n)

# Array to store the values of c
count_array <- numeric(n)

# Function call
for (i in 1:n){
  element <- generate_gamma_exponential(alpha, beta, lambda, theoretical_c)
  gamma_array[i] <- element[1]
  count_array[i] <- element[2]
}

# Sample mean
mean(gamma_array)
```

```
[1] 0.7940884
```

```
# Population mean
alpha/beta
```

```
[1] 0.8
```

```
# Sample Variance
var(gamma_array)
```

```
[1] 0.1540199
```

```
# Population Variance
alpha/(beta^2)
```

```
[1] 0.16
```
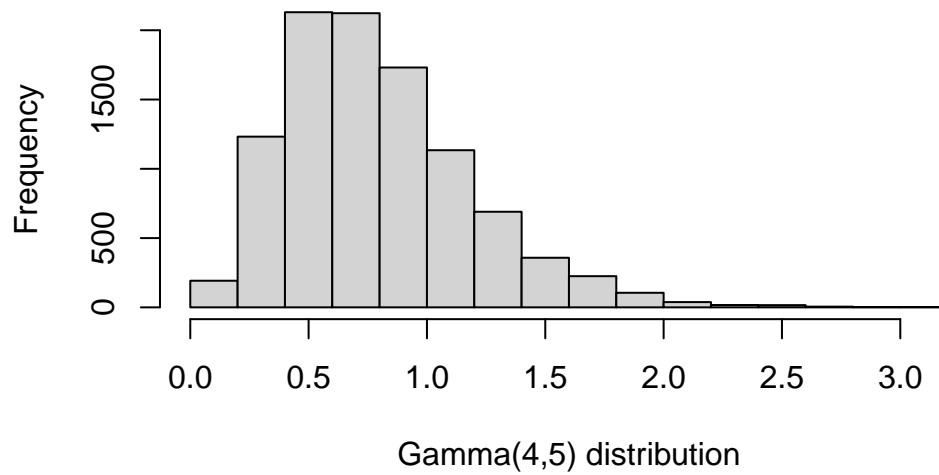
```
# Theoretical Value of c
theoretical_c
```

```
[1] 2.124248
```

```
# Estimated Value of c
mean(count_array)
```

```
[1] 2.1125
```

```
# Histogram of the generated gamma sample
hist(gamma_array, main = "Histogram of Gamma(4,5) distribution",
     xlab = "Gamma(4,5) distribution", ylab = "Frequency")
```

Histogram of Gamma(4,5) distribution

Frequency — Gamma(4,5) distribution

```
# Histogram of values of c
hist(count_array, main = "Histogram of distribution of c",
     xlab = "Distribution of c", ylab = "Frequency")
```

Histogram of distribution of c

Frequency — Distribution of c
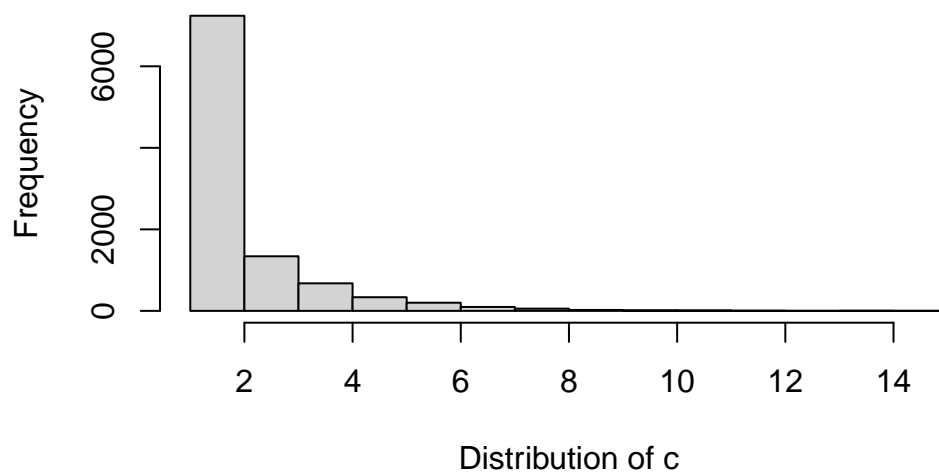
From the above, we can see that the sample mean of the Gamma(4,5) distribution is approximately the same as that of the population mean ($\frac{\alpha}{\beta} = \frac{4}{5} = 0.80$) and the sample variance is close to the population variance ($\frac{\alpha}{\beta^2}) = \frac{4}{5^2} = 0.16$. We can also see that the estimated value of c is quite close to the theoretical value of c.

We can see that distribution of c follows a Geometric distribution.

Problem 2

To generate 10^4 samples from a unit circle.

```r
## Generating 1e4 points inside a unit circle centered at origin

# Count of number of points generated
t_count <- 1

# Array to store the x-coordinates of the points generated
x_arr <- numeric(1e4)
# Array to store the y-coordinates of the points generated
y_arr <- numeric(1e4)

# Generating points inside the circle
while(t_count <= 1e4){

  # Drawing uniformly from square with side length 2
  x <- runif(1, min = -1, max = 1)
  y <- runif(1, min = -1, max = 1)

  # Checking if the point lies inside the circle
  if (x^2 + y^2 < 1){
    x_arr[t_count] <- x
    y_arr[t_count] <- y

    # Incrementing the count of points inside circle
    t_count <- t_count + 1
  }
}

## Scatterplot of the points generated
plot(x_arr, y_arr, pch = 16, col = 'turquoise3')
```
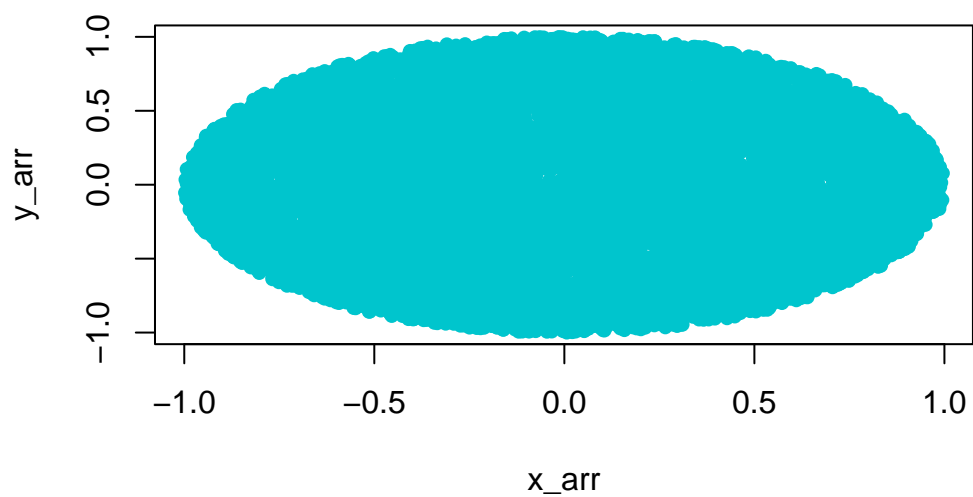
From the scatter-plot we can see that the points are generated inside the circle uniformly as there are no visible clustering of the generated point.