# Statistics Software Lab Report - 2

Name of the Student: Shatansh Patnaik
Roll No: 20MA20067

IIT Kharagpur
Statistics Software Lab

## 1 Generation of Random Variate under Discrete Distributions:

At its core, Inverse Direct Transform method relies on the cumulative distribution function and its inverse to transform uniformly distributed random variables into variables that adhere to the desired probability distribution.

Consider a random variable $X$ with a cumulative distribution function $F(x)$. The Direct Inverse Transform method enables the generation of random samples from $X$ by employing the inverse of its cumulative distribution function, denoted as $F^{-1}(u)$. Here, $u$ is a random variable uniformly distributed in the range $[0, 1)$. The algorithm involves two key steps: first, generating a random variable $u$ from a uniform distribution, and second, computing $X$ by applying the inverse of the cumulative distribution function to $u$.

### 1.1 Bernoulli Distribution

Let $X_i$ be a random variable following a Bernoulli distribution with parameter $p$. The probability mass function (PMF) of a Bernoulli distribution is given by:

$$P(X_i = x) = \begin{cases} p, & \text{if } x = 1, \\ 1 - p, & \text{if } x = 0. \end{cases}$$

Below is the algorithm that we shall be using for the generation of the random variables (Later we shall observe that this algorithm will be a direct result of the Direct Inverse Transform Method)

**Algorithm 1** Generate Binary Random Variable $X$

```
1: Generate a uniform random variable $U$ on the interval $[0, 1)$.
2: if $U \leq p$ then
3:     $X \leftarrow 1$
4: else
5:     $X \leftarrow 0$
6: end if
```

Same code can be illustrated for 1000 random samples generated via Uniform Distribution, we will be using the builtin function *runif()* to obtain the Uniform Distribution, following which we will use the above algorithm to generate the random sample following Bernoulli Distribution, the code in R is provided below for the same:

```r
BIG_NUM <- 1000
p =0.67

generate_bernoulli_rv <- function(p) {
  U <- runif(BIG_NUM)
  n <- length(U)
  X <- numeric(0)

  for (i in 1:n) {
    if (U[i] < p)
      X <- append(X, 1)
    else
```

```
      X <- append(X, 0)
  }
  return(X)
}

# Generating the random sample of Bernoulli Distribution
X <- generate_bernoulli_rv(p)
hist(X, main = "Histogram of Bernoulli Distribution", xlab = "Value", ylab =
    "Frequency", col = "lightgreen", border = "black")
```

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

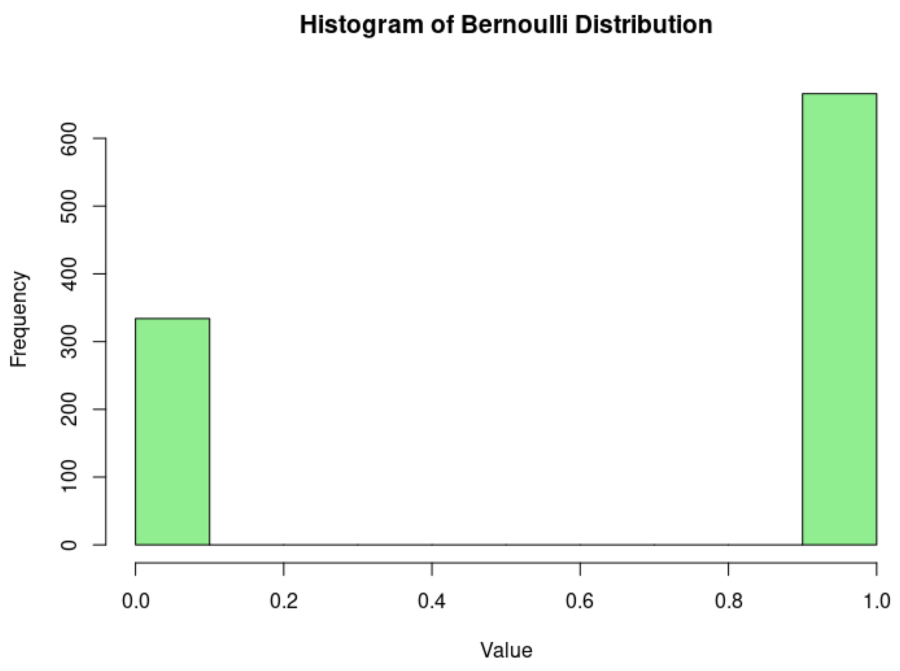

Figure 1: The given figure is a histogram plot of Bernoulli Distribution

Now we shall implement the Chi Square Goodness of Fit test to check if the required distribution is indeed Bernoulli. The following code illustrates the usage of Chi Square Goodness of Fit:

```
# Applying the Chi Square Goodness of fit test for the generated sample
get_frequency_bernoulli <- function (g) {
  n <- length(g)
  freq<-c(0,0)

  for (i in 1:n){
    if(g[i]==1){
      freq[1] <- freq[1] + 1
    } else{
```

```r
      freq[2] <- freq[2] + 1
    }
  }
  return (freq)
}

E <- c(BIG_NUM*(p), BIG_NUM*(1- p))
O <- get_frequency_bernoulli(X)

W <- sum(((O-E)^2)/E)
criticial_value <- qchisq(0.95, length(E)-1)

if(W > criticial_value){
  print("The given distribution doesnt follow Bernoulli Distribution")
} else {
  print("The given distribution follows Bernoulli Distribution")
}
```

## 1.2 Discrete Uniform Distribution

The density function for a discrete uniform distribution on the integers $i, i+1, \dots, j$ is given by:

$$p(x) = \begin{cases} \frac{1}{j-i+1}, & \text{if } x = i, i+1, \dots, j, \\ 0, & \text{otherwise.} \end{cases}$$

**Algorithm:** Generate $U \sim \text{Uniform}[0,1]$. Return $X = i + \lfloor (j-i+1)U \rfloor$.

**Algorithm 2** Generate Random Sample from Discrete Uniform Distribution

```
1: Generate a uniform random variable $U$ on the interval $[0, 1)$.
2: $X \leftarrow i + \lfloor (j-i+1)U \rfloor$.
3: Return $X$.
```

Same code can be illustrated for 1000 random samples generated via Uniform Discrete Distribution, we will be using the builtin function *runif()* to obtain the Uniform Distribution, following which we will use the above algorithm to generate the random sample following Discrete Uniform Distribution from a to b, the code in R is provided below for the same:

```r
i = 10
j = 700

get_discrete_uniform_dist <- function () {
  U <- runif(BIG_NUM, 0, 1)
  return (i + floor((j-i+1)*U))
}

X<-get_discrete_uniform_dist()
hist(X, main = "Histogram of Uniform Distribution", xlab = "Value", ylab = "
    Frequency", col = "red", border = "blue")
```

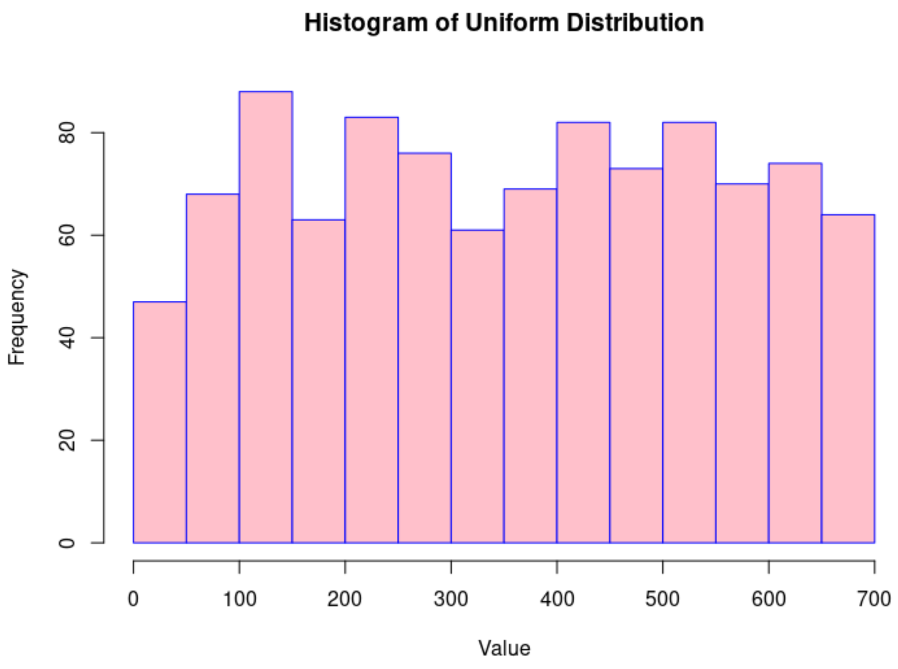

Figure 2: The given figure is a histogram plot of Discrete Uniform Distribution

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

Now we shall implement the Chi Square Goodness of Fit test to check if the required distribution is indeed Discrete Uniform. The following code illustrates the usage of Chi Square Goodness of Fit:

```
for(y in unique(X)){
  O <- append(O, length(X[X == y]))
  E <- append(E, 1000/(j-i))
}

W <- sum(((O-E)^2)/E)

criticial_value <- qchisq(0.95, length(E)-1)
if(W > criticial_value){
  print("The given distribution doesnt follow Discrete Uniform Distribution"
     )
} else {
  print("The given distribution follows Discrete Uniform Distribution")
}
```

## 1.3 Binomial $(n, p)$ Distribution

The probability mass function for a Binomial distribution with parameters $n$ (number of trials) and $p$ (probability of success) is given by:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad \text{for } k = 0, 1, \ldots, n.$$

**Algorithm:** Generate $Y_1, Y_2, \ldots, Y_n$ as independent Bernoulli $(p)$ random variables. Return $X = Y_1 + Y_2 + \ldots + Y_n$.

Below is the algorithm that we shall be using for the generation of the random variables (Later we shall observe that this algorithm will be a direct result of the Direct Inverse Transform Method)

**Algorithm 3** Generate Random Sample from Binomial $(n, p)$ Distribution

```
1: for $i = 1$ to $n$ do
2:     Generate a Bernoulli $(p)$ random variable $Y_i$.
3: end for
4: $X \leftarrow Y_1 + Y_2 + \ldots + Y_n$.
5: Return $X$.
```

Same code can be illustrated for 1000 random samples generated via Uniform Distribution, we will be using the builtin function *runif()* to obtain the Uniform Distribution, following which we will use the above algorithm to generate the random sample following Binomial Distribution, the code in R is provided below for the same:

```r
generate_binomial_distribution <- function (p) {
  U <- runif(BIG_NUM)
  X <- rep(0, BIG_NUM)
  for (l in 1:BIG_NUM){
    H <- generate_bernoulli_rv(p)
    X <- X + H
  }
  return (X)
}

p=0.5
X<-generate_binomial_distribution(p)
hist(X, main = "Histogram of Binomial Distribution", xlab = "Value", ylab =
    "Frequency", col = "orange", border = "blue")
```

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

Now we shall implement the Chi Square Goodness of Fit test to check if the required distribution is indeed Binomial. The following code illustrates the usage of Chi Square Goodness of Fit:

```r
# Now we shall calculate the frequency of each element in X
O <- numeric(0)
E <- numeric(0)

# O and E calculations:
```

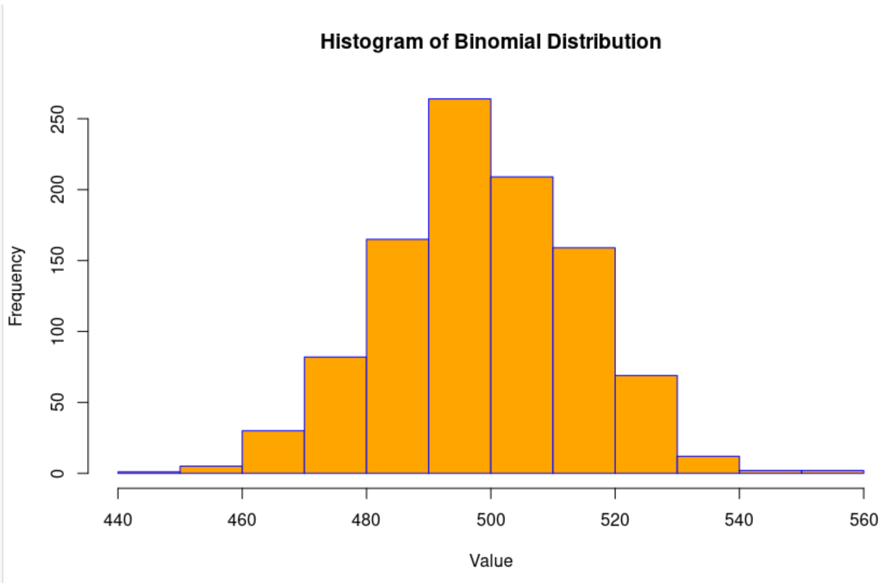

Figure 3: The given figure is a histogram plot of Binomial Distribution

```r
for(y in unique(X)){
  O <- append(O, length(X[X == y]))
  E <- append(E, (1000 * (choose(1000,y)*(p^y)*(1-p)^(1000-y))))
}

# Caluclations of values
W <- sum(((O-E)^2)/E)
criticial_value <- qchisq(0.95, length(E)-1)

if(W > criticial_value){
  print("The given distribution doesnt follow Binomial Distribution")
} else {
  print("The given distribution follows Binomial Distribution")
}
```

## 1.4 Geometric Distribution

The probability mass function for a Geometric distribution with parameter $p$ (probability of success on each trial) is given by:

$$P(X = k) = (1 - p)^{k-1}p, \quad \text{for } k = 1, 2, \ldots.$$

Same code can be illustrated for 1000 random samples generated via Uniform Distribution, we will be using the builtin function *runif()* to obtain the Uniform Distribution, following which we will use the above algorithm to generate the random sample following Geometric Distribution, the code in R is provided below for the same:

**Algorithm 4** Generate Random Sample from Geometric $(p)$ Distribution

```
1: Generate a uniform random variable $U$ on the interval $[0, 1)$.
2: $X \leftarrow \lfloor \frac{\log U}{\log(1-p)} \rfloor + 1$.
3: Return $X$.
```

```r

generate_geometric_distribution <- function (p) {
  U <- runif(BIG_NUM)
  H <- floor(log(U)/log(1-p))
  return (H)
}

X <- generate_geometric_distribution(0.3)
hist(X, main = "Histogram of Geometric Distribution", xlab = "Value", ylab =
     "Frequency", col = "lightyellow", border = "blue")
```

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

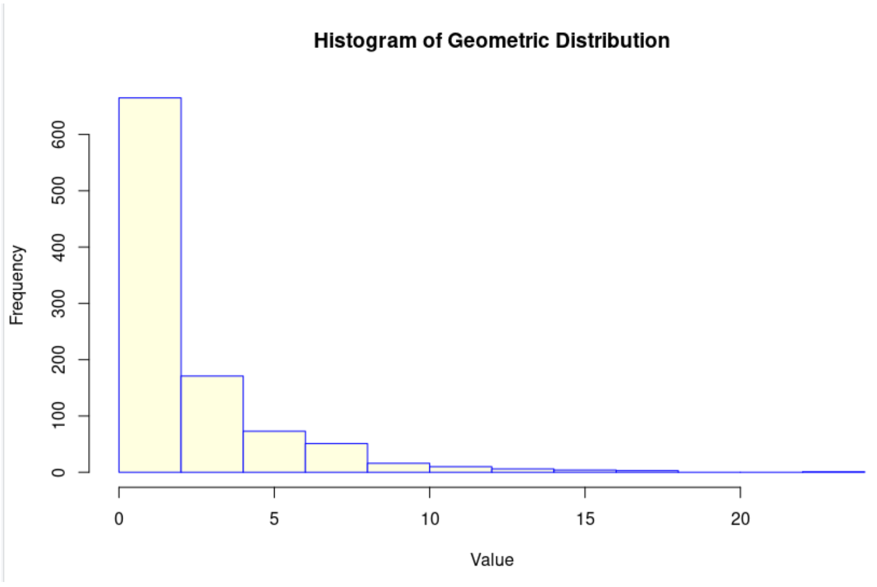

Figure 4: The given figure is a histogram plot of Geometric Distribution

```r
O <- numeric(0)
E <- numeric(0)

# O and E calculations:

for(y in unique(X)){
  O <- append(O, length(X[X == y]))
  E <- append(E, 1000*((1-p)^y)*p)
}


# Calculations of values
W <- sum(((O-E)^2)/E)
criticial_value <- qchisq(0.95, length(E)-1)

if(W > criticial_value){
  print("The given distribution doesnt follow Geometric Distribution")
} else {
  print("The given distribution follows Geometric Distribution")
}
```

## 1.5 Negative Binomial $(r, p)$ Distribution

The probability mass function for a Negative Binomial distribution with parameters $r$ (number of failures until the $r$-th success) and $p$ (probability of success on each trial) is given by:

$$P(X = k) = \binom{k + r - 1}{r - 1} p^r (1 - p)^k, \quad \text{for } k = 0, 1, 2, \ldots.$$

Below is the algorithm that we shall be using for the generation of the random variables (Later we shall observe that this algorithm will be a direct result of the Direct Inverse Transform Method)

**Algorithm 5** Generate Random Sample from Negative Binomial $(r, p)$ Distribution

```
1: for $i = 1$ to $r$ do
2:     Generate a Geometric ($p$) random variable $Y_i$.
3: end for
4: $X \leftarrow Y_1 + Y_2 + \ldots + Y_r$.
5: Return $X$.
```

Same code can be illustrated for 1000 random samples generated via Uniform Distribution, we will be using the builtin function *runif()* to obtain the Negative Binomial $(r, p)$ Distribution, following which we will use the above algorithm to generate the random sample following Negative Binomial Distribution, the code in R is provided below for the same:

```r
r <- 10
p <- 0.45

generate_negative_binomial_distribution_rv <- function (r, p) {
  X <- rep(0, BIG_NUM)
  for (i in 1: r){
    H <- generate_geometric_distribution(p)
    X <- X + H
  }
  return(X)
}

```

```
X <- generate_negative_binomial_distribution_rv(r, p)
hist(X, main = "Histogram of Negative Binomial Distribution", xlab = "Value"
    , ylab = "Frequency", col = "lightgreen", border = "blue")
```

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

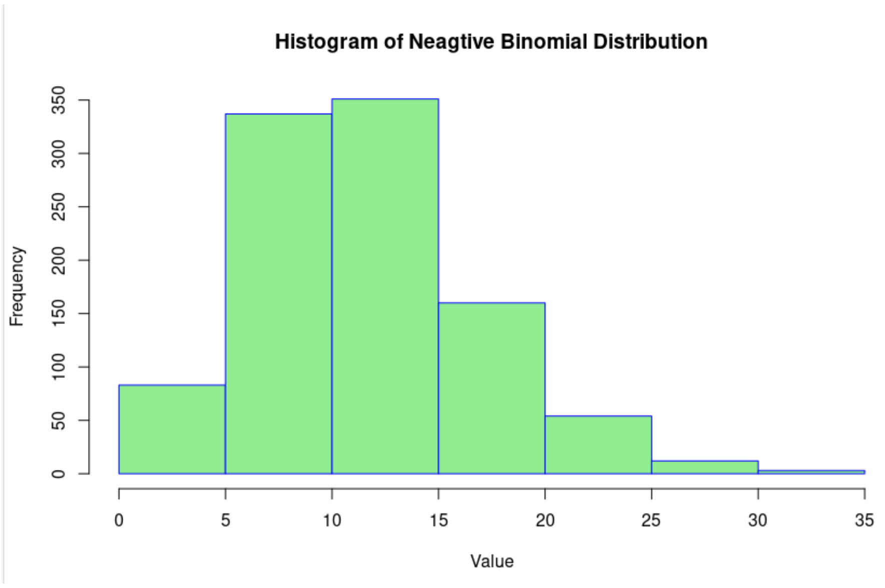

Figure 5: The given figure is a histogram plot of Negative Binomial Distribution

Now we shall implement the Chi Square Goodness of Fit test to check if the required distribution is indeed Negative Binomial. The following code illustrates the usage of Chi Square Goodness of Fit:

```

# O and E calculations:
O <- numeric(0)
E <- numeric(0)
for(y in unique(X)){
  O <- append(O, length(X[X == y]))
  E <- append(E, 1000*(choose(y+r-1, y)*((1-p)^y)*p^r))
}

# Calculations of values
W <- sum(((O-E)^2)/E)
criticial_value <- qchisq(0.95, length(E)-1)

if(W > criticial_value){
  print("The given distribution doesnt follow Negative Binomial Distribution
      ")
} else {
```

```r
  print("The given distribution follows Negative Binomial Distribution")
}
```

## 1.6 Poisson Distribution

For generating Poisson Distribution, we can follow the following steps:

1. Let $a = e^{-\lambda}$, $b = 1$, $i = 0$.
2. Generate $U_{i+1} \sim$ Uniform$[0, 1]$ and replace $b$ by $bU_{i+1}$. If $b < a$, return $X = i$, otherwise go to Step 3.
3. Replace $i$ by $i + 1$ and go back to Step 2.

Below is the algorithm that we shall be using for the generation of the random variables (Later we shall observe that this algorithm will be a direct result of the Direct Inverse Transform Method)

**Algorithm 6** Generate Random Sample from Poisson ($\lambda$) Distribution

```
1: Let $a = e^{-\lambda}$, $b = 1$, $i = 0$.
2: while true do
3:     Generate $U_{i+1} \sim$ Uniform$[0, 1]$ and replace $b$ by $bU_{i+1}$.
4:     if $b < a$ then
5:         Return $X = i$.
6:     else
7:         $i \leftarrow i + 1$.
8:     end if
9: end while
```

Same code can be illustrated for 1000 random samples generated via Uniform Distribution, we will be using the builtin function *runif()* to obtain the Poisson Distribution, following which we will use the above algorithm to generate the random sample following Poisson Distribution, the code in R is provided below for the same:

```r
n <- 1000
lambda <- 13

generate_each_entry_of_Poisson <- function (lambda) {
  a <- exp(-lambda)
  b <- 1
  i <- 0
  X <- numeric(0)

  while(1) {
    u <- runif(1, 0, 1)
    b <- b*u

    if(b < a){
      return(i)
    }
    i <- i+1
    }

  return (X)
}

generate_Poisson_Distribution <- function (n, lambda) {
  X <- numeric(0)
  for (i in 1:n){
    X <- append(X, generate_each_entry_of_Poisson(lambda))
  }
  return(X)
}

X <- generate_Poisson_Distribution(n, lambda)

hist(X, main = "Histogram of Poisson Distribution", xlab = "Value", ylab = "
    Frequency", col = "lightcyan", border = "blue")
```

We can illustrate the following in a histogram with X-axis representing the Values in the Distribution that are generated and y-axis represents the Frequencies of those values in the distribution.

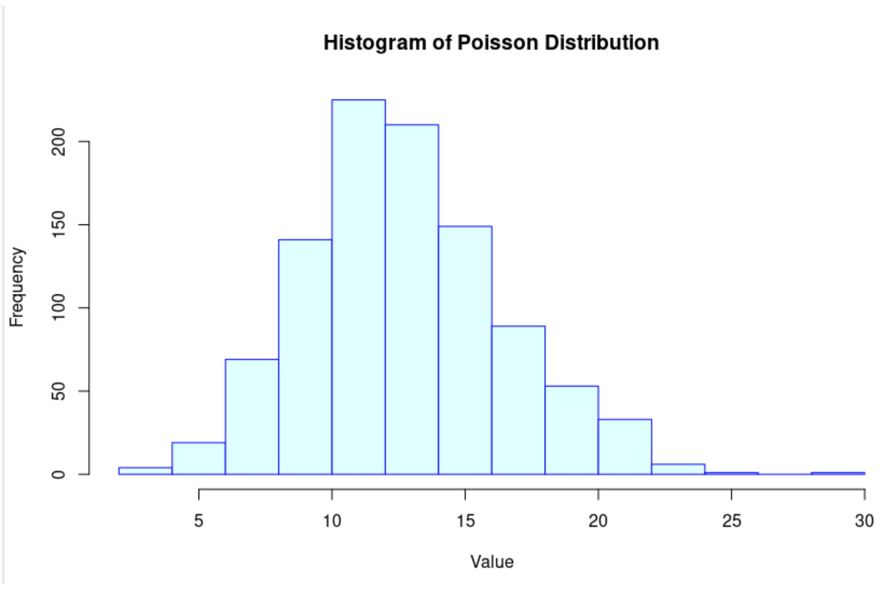

Figure 6: The given figure is a histogram plot of Poisson Distribution

Now we shall implement the Chi Square Goodness of Fit test to check if the required distribution is indeed Poisson Distribution. The following code illustrates the usage of Chi Square Goodness of Fit:

```
O <- numeric(0)
E <- numeric(0)

```

```r
# O and E calculations:
for(y in unique(X)){
  O <- append(O, length(X[X == y]))
  E <- append(E, 1000*((lambda^y)*exp(-lambda)/factorial(y)))
}

# Calculations of values
W <- sum(((O-E)^2)/E)
criticial_value <- qchisq(0.95, length(E)-1)

if(W > criticial_value){
  print("The given distribution doesnt follow Poisson Distribution")
} else {
  print("The given distribution follows Poisson Distribution")
}
```

## Inferences from the comparision of the Algorithms and Direct Inverse Transform Method

Mathematically, it can easily be shown that the algorithms that we've used previously for the generation of random samples under different distributions are direct consequences of the Direct Inverse Transform Method. But since we are using R for the calculations of values, there might be floating point and other estimation related errors. So we shall end up getting different results, some being more accurate and some being less accurate in both these methods.

For instance, in the generation of Bernoulli, Discrete Uniform Distribution, Binomial Distribution, Geometric Distribution, the Chi-Square value is more in case of the algorithms compared to the Direct Inverse Transform method. But in case of Negative Binomial and Poisson Distribution, the opposite effect is observed. We can also observe that in all the above cases the margin of difference of the values between the two methods is not very high.

Given below are the functions that has been used in the Generation of Random Variables via Direct Inverse Transform method

```r
generate_each_entry_of_dist <- function(pmf){
  i <- 0
  U <- runif(1,0,1)
  value <- 0

  while(1){
    value <- value + pmf(i)
    if(value>U){
      return(i)
    }
    i <- i+1;
  }
}
generate_sample <- function(n, pmf){
  X <- numeric(0)
  for(i in 1:n){
    X <- append(X, generate_each_entry_of_dist(pmf))
  }
  return(X)
}
```