Lab2

Problem 1. Use the inverse transformation method to generate 10^4 samples from

(1) Bernoulli(1/2), Poisson(4), Binomial(10,1/3).

(2) For each distribution given above, calculate the sample mean and sample variance and compare with the population mean and variance.

Solution :

```
#### Problem 1
### Part 1
## Bernoulli (1/2)

# Success Probability pb and Failure Probability qb
pb <- 1/2
qb <- 1/2

# Array to store the random sample
bernoulli_array <- numeric(1e4)

# Sample Generation
for (i in 1:1e4){
  U <- runif(1, min = 0, max = 1)
  # Assigning values to the sample observations
  if (U < qb){
    bernoulli_array[i] <- 0
  }
  else{
    bernoulli_array[i] <- 1
  }
}
```

```r
# To count the number of 1's in the generated sample
sum(bernoulli_array)
```

```
[1] 5042
```

Here, I have used Uniform[0,1] distribution to generate the Bernoulli(1/2) distribution.

```r
### Part 2

## For Bernoulli(1/2)
sample_mean <- mean(bernoulli_array)            # Mean of the generated sample
sample_var <- var(bernoulli_array)              # Variance of the generated sample
population_mean <- pb                           # Population mean (pb)
population_var <- pb*qb                         # Population Variance (pb * qb)

sample_mean
```

```
[1] 0.5042
```

```r
population_mean
```

```
[1] 0.5
```

```r
sample_var
```

```
[1] 0.2500074
```

```r
population_var
```

```
[1] 0.25
```

From the values obtained by calculating the sample mean and sample variance for the generated sample, we can see that the value of sample mean is approximately equal to that of the population mean ($pb = 0.5$) in this case and the value of sample variance is approximately equal to the population variance ($pb * qb = 0.25$) in this case.

```r
### Part 1
## Poisson(4)

# Parameter for Poisson distribution
lambda <- 4

# Array to store the random sample
poisson_array <- numeric(1e4)

# Sample Generation
for (i in 1:1e4){
  j <- 0
  # Initializing base case for the recurrence relation
  p <- exp(-lambda)
  f <- 0
  U <- runif(1, min = 0, max = 1)
  # Generating thresholds of intervals to assign sample values
  while(TRUE){
    f <- f + p
    # Checking to assign values
    if(U <= f){
      poisson_array[i] <- j
      break
    }
    j <- j + 1
    # Updating the recurrence relation
    p <- lambda*p/j
  }
}
```

Here, I have used Uniform[0,1] distribution to generate the Poisson(4) distribution.

```r
### Part 2
## For Poisson(4)
samp_mean <- mean(poisson_array)            # Mean of the generated sample
samp_var <- var(poisson_array)              # Variance of the generated sample
pop_mean <- lambda                          # Population Mean (lambda)
pop_var <- lambda                           # Population Variance (lambda)

samp_mean
```

```
[1] 4.0018
```

```
pop_mean
```

```
[1] 4
```

```
samp_var
```

```
[1] 4.050602
```

```
pop_var
```

```
[1] 4
```

From the values obtained by calculating the sample mean and sample variance for the generated sample, we can see that the value of sample mean is approximately equal to that of the population mean ($\lambda = 4$) in this case and the value of sample variance is approximately equal to the population variance ($\lambda = 4$) in this case. Because for Poisson(λ) the population mean and population variance are both equal to λ.

```
### Part 1
## Binomial(10,1/3)

# Success probability(pbin) and failure probability(qbin)
pbin <- 1/3
qbin <- 2/3
fl <- 0

# Array to store the random sample
binomial_array <- numeric(1e4)
# Array to store the upper limits of the intervals
bin_lst <- numeric(11)

# Generating the upper limits of the intervals
for (i in 1:11){
  ele <- choose(10,(i-1))*(pbin^(i-1))*(qbin^(10-i+1))
  fl <- fl + ele
  bin_lst[i] <- fl
}

# Sample Generation
for (j in 1:1e4){
```

```
  U <- runif(1, min = 0, max = 1)
  # Checking the interval the sample point is a part of
  for (i in 1:11){
    if (U <= bin_lst[i]){
      binomial_array[j] <- (i - 1)      # Assigning the proper value for conversion
      break
    }
  }
}
```

Here, I have used Uniform[0,1] distribution to generate Binomial(10, $\frac{1}{3}$) distribution.

```
### Part 2
## For Binomial(10,1/3)
s_mean <- mean(binomial_array)                # Mean of the generated sample
s_var <- var(binomial_array)                  # Variance of the generated sample
p_mean <- 10*pbin                             # Population Mean
p_var <- 10*pbin*qbin                         # Population Variance

s_mean
```

```
[1] 3.3324
```

```
p_mean
```

```
[1] 3.333333
```

```
s_var
```

```
[1] 2.245135
```

```
p_var
```

```
[1] 2.222222
```

From the values obtained by calculating the sample mean and sample variance for the generated sample, we can see that the value of sample mean is approximately equal to that of the population mean ($n * pbin = 10 * \frac{1}{3} = 3.33$) in this case and the value of sample variance is

approximately equal to the population variance ($n * pbin * qbin = 10 * \frac{1}{3} * \frac{2}{3} = 2.22$) in this case.

Problem 2. Define N = $\lfloor \frac{log(U)}{log(1-p)} \rfloor$ where U Uniform(0,1) and 0 < p < 1. Generate 10^4 samples of N and calculated sample mean and sample variance.

```
# Taking a specific value of parameter p
p <- 1/3

# Array to store the random sample
N <- numeric(1e4)

# Sample Generation
for (i in 1:1e4){
  U <- runif(1, min = 0, max = 1)
  N[i] <- floor(log(U)/log(1-p))
}


sampl_mean <- mean(N)              # Mean of the generated sample
sampl_var <- var(N)                # Variance of the generated sample

sampl_mean
```

```
[1] 1.9832
```

```
sampl_var
```

```
[1] 5.72329
```

Here, I have used Uniform$(0, 1)$ distribution to generate the required sample.