# Statistics Software Lab Report - 3 (Outputs file)

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

IIT Kharagpur
Statistics Software Lab

## Output for Exercise-1

```
> #############################################################s###########
> # Exercise - 1: Generation of a random variable having beta density
> set.seed(20067)
>
> # Step-0: Simpson's Integration method
> simpsons_rule_integral <- function(gimme_func, lo, hi, n){
+   steps <- (hi - lo) / n
+   res <- gimme_func(lo) + gimme_func(hi)
+
+   for (i in 1:n-1) {
+     a_i <- lo + i * steps
+     res <- res + 2*gimme_func(a_i)*(1 + i %% 2)
+   }
+
+   res <- res*(steps/3)
+
+   return(res)
+ }
>
> # Step - 1: We generate the random variables via the specified algorithm:
> generate_beta_random_variables <- function() {
+   while(1){
+     u1 <- runif(1,0,1)
+     u2 <- runif(1,0,1)
+
+     if(u2 <= (256/27)*u1*((1-u1)^3)){
+       return(u1)
+     }
+   }
+ }
>
> generate_b_rv <- function (n) {
+   X <- numeric(0)
+   for (i in 1:n){
+     X <- append(X, generate_beta_random_variables())
+   }
+   return(X)
+ }
>
> # Step-2: We write the cdf for the upcoming calculations
> beta <- function(x) {
+   return (20*x*((1-x)^3))
+ }
>
> # Step-3: We perform the Chi Square Goodness of Fit Test
> do_chi_sq_test <- function(O, E, s){
+   W <- sum(((O-E)^2)/E)
+   critical_value <- qchisq(0.95, 9)
+
+   print(W)
```

```
+   print(critical_value)
+
+   if(W > critical_value){
+     sprintf("The given distribution doesnt follow %s Distribution", s)
+   } else {
+     sprintf("The given distribution follows %s Distribution", s)
+   }
+ }
>
> do_the_tests <- function(X, O, breaks, s) {
+   E <- numeric(0)
+   for(i in 1:10){
+     y <- simpsons_rule_integral(beta, breaks[i], breaks[i+1], 1000)
+     E <- append(E, 1000*y)
+   }
+   do_chi_sq_test(O, E, s)
+ }
>
> # We are assuming the number of buckets is 10
> X <- generate_b_rv(1000)
> breaks <- seq(0, 1, 0.1)
> O <- hist(X, breaks=breaks, main = "Histogram of Beta Distribution", xlab = "Value", ylab = "Frequency", col = "lightgreen", border = "blue")$count
> do_the_tests(X, O, breaks, "Beta")
[1] 7.916101
[1] 16.91898
[1] "The given distribution follows Beta Distribution"
> ####################################################s###########
```

## Output for Exercise-2

```
> ####################################################s###########
> # Exercise - 2: Generation of random standard variables
> generate_normal_rv <- function() {
+   while(1){
+     y1 <- rexp(1,1)
+     y2 <- rexp(1,1)
+
+     if(y2 > (0.5)*(y1-1)^2){
+       y <- y2 - (0.5)*(y1-1)^2
+       u <- runif(1,0,1)
+
+       if(u<=0.5){
+         return(y1)
+       }else{
+         return(-y1)
+       }
+     }
+   }
+ }
```

```
>
> generate_normal <- function(n) {
+   X <- numeric(0)
+   for(i in 1:n){
+     X <- append(X, generate_normal_rv())
+   }
+   return (X)
+ }
>
> normal_pdf <- function (x){
+   return((1/sqrt(2*pi))*exp(-x^2/2))
+ }
>
> # do_chi_sq_test_2 <- function (X, normal_pdf, n, s){
>
> # }
>
> t1 <- Sys.time()
> X <- generate_normal(1000)
> hist(X, main = "Histogram of Normal Distribution", xlab = "Value", ylab =
   "Frequency", col = "darkgoldenrod2", border = "brown")
> maxine <- max(X)
> minine <- min(X)
>
> n <- 10
> k <- (maxine-minine)/n
> ints <- numeric(0)
> E <- numeric(0)
> O <- numeric(0)
>
> for(i in 1:n){
+   ints <- append(ints, minine +(i-1)*k)
+ }
>
> ints[n+1] <- maxine
>
> for(i in 1:n){
+   cdfValue <- simpsons_rule_integral(normal_pdf,ints[i],ints[i+1],1000)
+   E <- append(E, cdfValue*1000)
+   O <- append(O, length(X[X<=ints[i+1]])-length(X[X<ints[i]]))
+ }
> W <- sum((O-E)^2/E)
> critical_value <- qchisq(0.95,n-1)
> print(W)
[1] 12.15747
> print(critical_value)
[1] 16.91898
>
> if(W > critical_value){
+   sprintf("The given distribution doesnt follow Standard Normal
   Distribution")
+ } else {
```

```
+   sprintf("The given distribution follows Standard Normal Distribution")
+ }
[1] "The given distribution follows Standard Normal Distribution"
> ###########################################################s###########
```

### Output for Exercise-3

```
> ###########################################################s###########
> # Question: 3 Generation of a random variable X that takes one of the
    values 1,2,...,10
> p <- c(0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10)
>
> generate_random_element <- function(){
+   while(TRUE){
+     u1 <- runif(1,0,1)
+     y <- floor(10*u1)+1
+     u2 <- runif(1,0,1)
+     if(u2<=(p[y]/(0.12))){
+       return(y)
+     }
+   }
+ }
>
> generate_sample <- function(n){
+   X <- numeric(0)
+   for(i in 1:n){
+     X <- append(X, generate_random_element())
+   }
+   return(X)
+ }
>
> X <- generate_sample(1000)
> hist(X, main = "Histogram of the given Distribution", xlab = "Value", ylab
   = "Frequency", col = "lightpink", border = "brown")
> # Now we shall calculate the frequency of each element in X
> 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 * p[y]))
+ }
>
> # Calculations of values
> W <- sum(((O-E)^2)/E)
> criticial_value <- qchisq(0.95, length(E)-1)
> print(W)
[1] 3.259419
> print(critical_value)
[1] 16.91898
```

```
>
> if(W > criticial_value){
+   print("The given distribution doesnt follow the given Distribution")
+ } else {
+   print("The given distribution follows the given Distribution")
+ }
[1] "The given distribution follows the given Distribution"
> ########################################################s###########
```