Assignment 2

Random Number Generation through CDF and acceptance-rejection sampling

Roman Grebnev

2022-10-25

Contents

Task 1 1
    Linear Congruential Random Number Generation Algorithm 1
Task 2 3
    How can you obtain then a random sample from the exponential distribution? 3
    Write down the mathematical expression for this. 3
    Implementation of the inversion method for exponential distribution 3
    Comparison of the exponential distribution obtained based on the inversion method vs. theoretical exponential distribution 3
    Experiments with parameter lambda 4
Task 3 5
    Write a function which uses an acceptance-rejection approach to sample from a beta distribution. 5
    What is a natural candidate for a proposal(instrumental) distribution? 6
    Finding a good constant to keep the rejection proportion small 7

Task 1

Linear Congruential Random Number Generation Algorithm

The generated sequence is defined based on the linear equation: $x_{n+1} = (ax_n + c)(modm)$

Linear Congruential Random Number Algorithm is defined as follows:

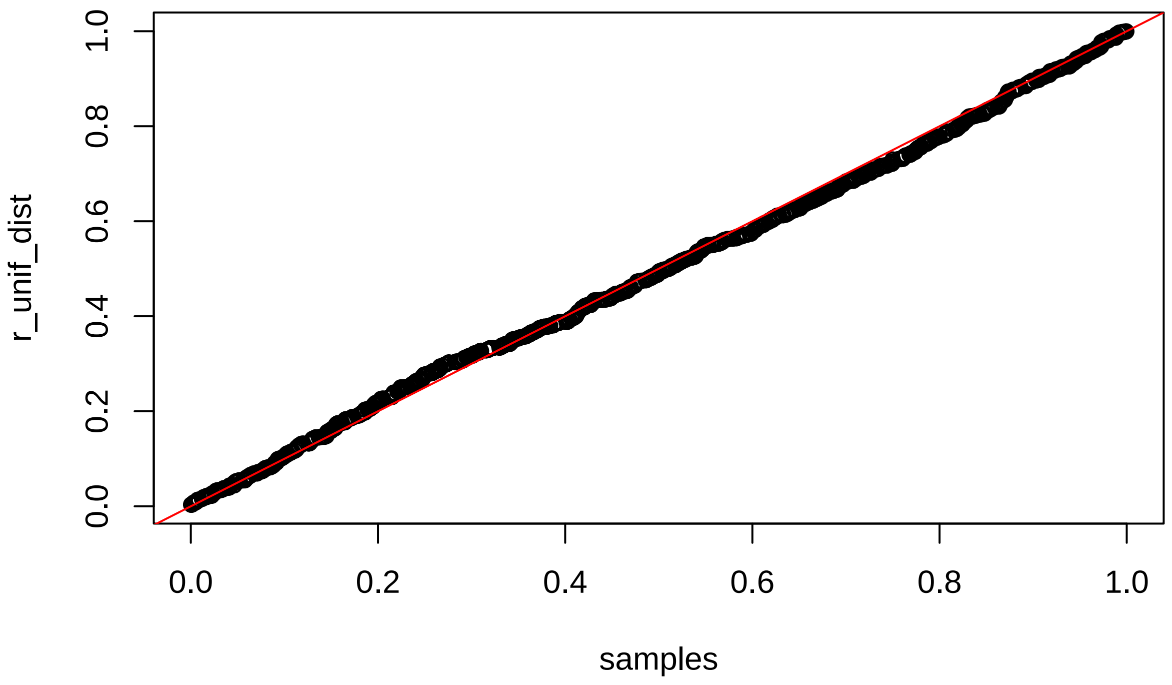
```r
mc.gen <- function(n,m,a,c=0,x0)
{
  us <- numeric(n)
  for (i in 1:n)
  {
    x0 <- (a*x0+c) %% m
    us[i] <- x0 / m
  }
  return(us)
}
```

Let's try the following parameters for Congruential RNG Algorithm:

```r
seed_x0 <- 346
n_samples <- 1000
m <- 50000000000
a <- round(sqrt(m))
c <- 5000
```

```
samples <- mc.gen(n_samples, m, a, c, seed_x0)
```

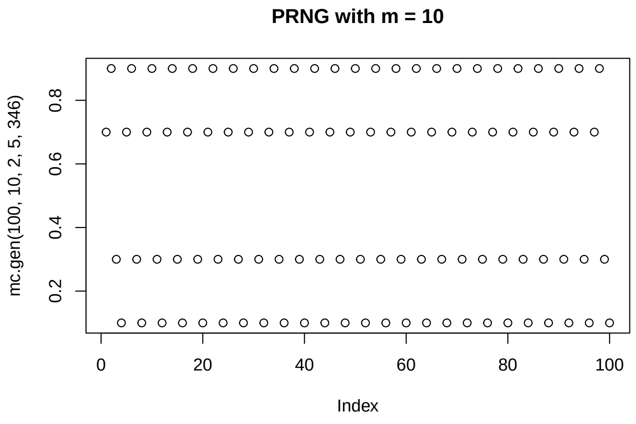
```
r_unif_dist <- runif(1000)
qqplot(samples, r_unif_dist)
abline(a=0,b=1, col = "red")
```

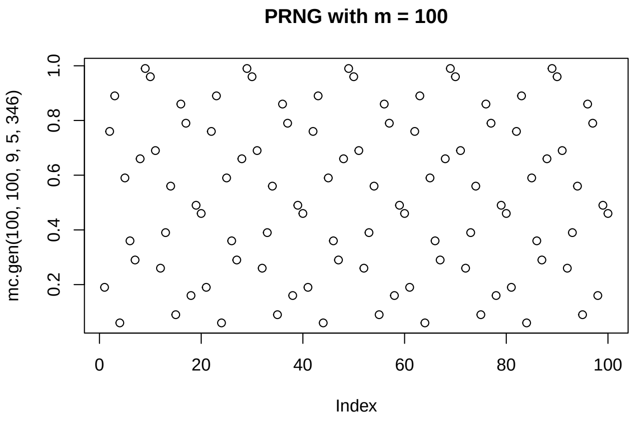


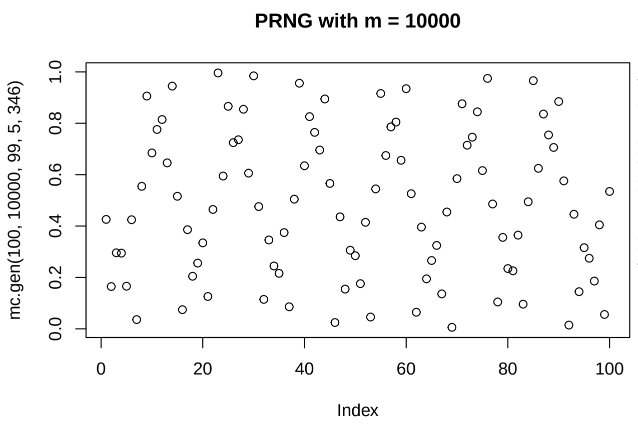
Obtained sample is comparable to the theoretical uniform distribution.

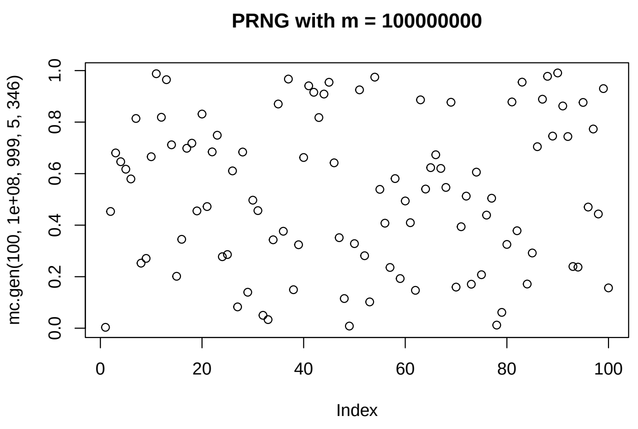
Let's compare the behavior of the Congruential Linear Algorithm with different values m

```
plot(mc.gen(100, 10, 2, 5, 346), main = "PRNG with m = 10")
plot(mc.gen(100, 100, 9, 5, 346),  main = "PRNG with m = 100")
plot(mc.gen(100, 10000, 99, 5, 346),  main = "PRNG with m = 10000")
plot(mc.gen(100, 100000000, 999, 5, 346),  main = "PRNG with m = 100000000")
```









Sequences, generated for relatively small m's are not random and it is possible to predict the next element of the squence given the previous ones. Approach works well with high m's.

Task 2

How can you obtain then a random sample from the exponential distribution?

Explanation of the approach:

- We can draw samples from the uniform distribution. Later on they can be used as the arguments for the quantile function of any desired distribution. Quantile function provides x for corresponding probability value.
- We can obtain the quantile function of the cumulative density function $F(x) = 1 - exp(-\lambda x), \lambda > 0$ by inverting it. Which looks as follows $F_x^{-1}(u) = \frac{-ln(1-p)}{\lambda}$.
- Based on the quantile function of the exponential distribution and the generated sample drawn from the uniform distribution, we can obtain the sample with the exponential distribution.

Write down the mathematical expression for this.

To obtain the random sample drawn from the exponential distribution we can use quantile function of the exponential distribution: $F_x^{-1}(u) = \frac{-ln(1-p)}{\lambda}$

Implementation of the inversion method for exponential distribution

```
exp_quant.gen <- function(n_samples_exp, lambda)
{
  unif_samle <- runif(n_samples_exp)
  exp_dist_sample <- -log(1-unif_samle)/lambda
  return(exp_dist_sample)
}
```

Comparison of the exponential distribution obtained based on the inversion method vs. theoretical exponential distribution

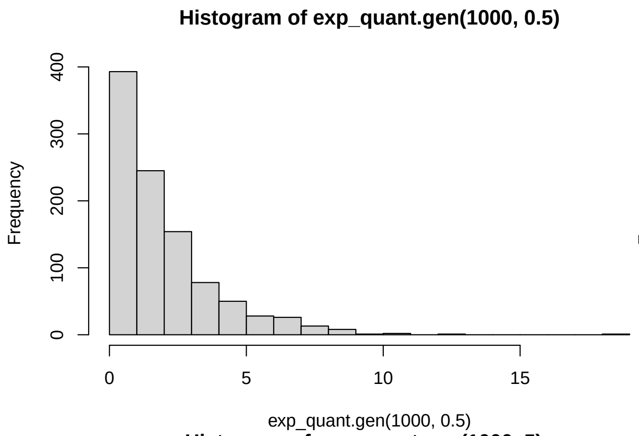
To obtain exponential distribution we can use values from the univariate distribution, using x

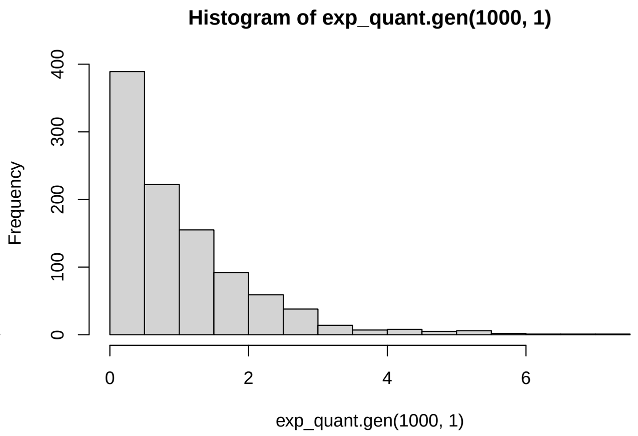
```
exp.gen <- function(n_samples_exp, lambda)
{
  unif_samle <- runif(n_samples_exp)
  exp_dist_sample <- 1 - exp(-lambda * unif_samle)
  return(exp_dist_sample)
}
```

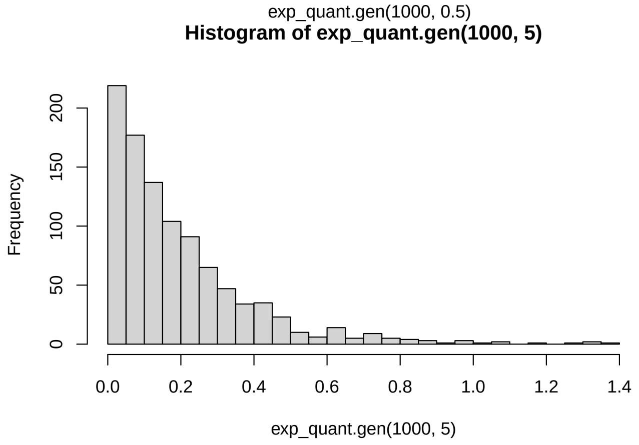
Experiments with parameter lambda

For 3 different values of lambda create 1000 random samples and evaluate the quality of your random number generator using qq-plots.

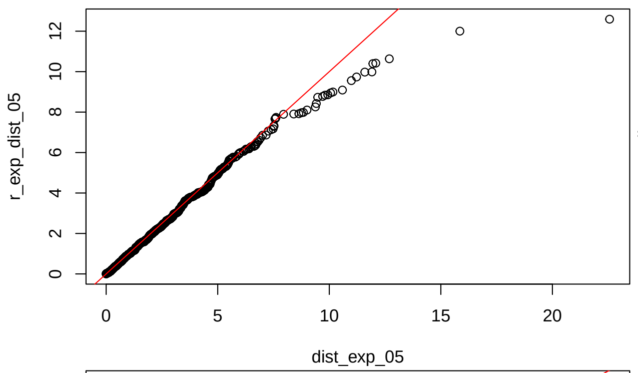
```r
hist(exp_quant.gen(1000, 0.5), breaks = 20)
hist(exp_quant.gen(1000, 1), breaks = 20)
hist(exp_quant.gen(1000, 5), breaks = 20)
```

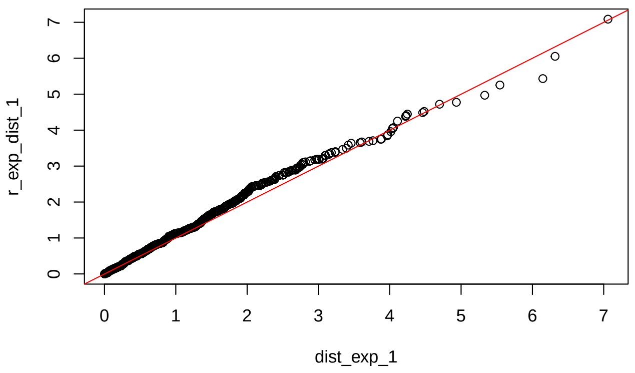


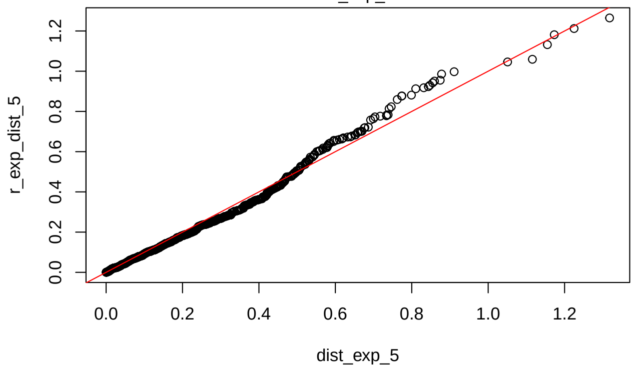




```r
dist_exp_05 <- exp_quant.gen(1000, 0.5)
dist_exp_1 <- exp_quant.gen(1000, 1)
dist_exp_5 <- exp_quant.gen(1000, 5)

r_exp_dist_05 <- rexp(1000, 0.5)
r_exp_dist_1 <- rexp(1000, 1)
r_exp_dist_5 <- rexp(1000, 5)
qqplot(dist_exp_05, r_exp_dist_05)
abline(a=0,b=1, col = "red")
qqplot(dist_exp_1, r_exp_dist_1)
abline(a=0,b=1, col = "red")
qqplot(dist_exp_5, r_exp_dist_5)
abline(a=0,b=1, col = "red")
```







Conclusions: For all tested values of parameter lambda upper quantiles of the QQ-plot demonstrate bigger deviations from the quantiles of theoretical exponential distribution. Deviations might be introduced as the result of PRNG of the uniform distribution, used to obtain the exponential distribution.

Therefore, we can obtain the random sample, drawn from the exponential distribution based on the inversion method under following conditions:

- We know the quantile function of the exponential distribution or we can obtain it relatively easy
- We have a good PRNG algorithm for generation of uniformly distributed sample
- CDF of the target distribution can be inverted

Task 3

Description of acceptance-rejection approach:

- Generate a random number coming from g beta(alpha = 2, beta = 2).
- Generate u coming from instrumental/envelope distribution.
- Accept $x = y$ if $u \leq \frac{f(x)}{cg(x)}$, reject $x = y$ if $u \geq \frac{f(x)}{cg(x)}$ and start over.
- The accepted value x comes then from density function f.
- To obtain n values, method has to be accepted until n values are accepted.

General properties of acceptance-rejection approach:

- Constant c is always larger than 1.
- Instrumental distribution should be chosen that has the smallest value c.
- To obtain n samples from f requires then roughly cn random samples from g and from instrumental distribution.

Write a function which uses an acceptance-rejection approach to sample from a beta distribution.

Implementation of the acceptance-rejection approach for sampling from beta distribution with uniform distribution as an instrumental distribution.

```r
norm_accept_reject <- function(n_samples_ar, const, alpha, beta){
iter <- 0
accepted <- 0
x <- numeric(n_samples_ar)
acc_rate <- 0
while(accepted<n_samples_ar)
{
u <- runif(1)
iter <- iter + 1
y <- runif(1)

if (dbeta(y, alpha, beta)/(const*dunif(y)) >= u)
  {
  accepted <- accepted+1
  x[accepted] <- y
  }
}
acc_rate <- accepted/iter
return(list(acc_rate, x))
}
```

What is a natural candidate for a proposal(instrumental) distribution?

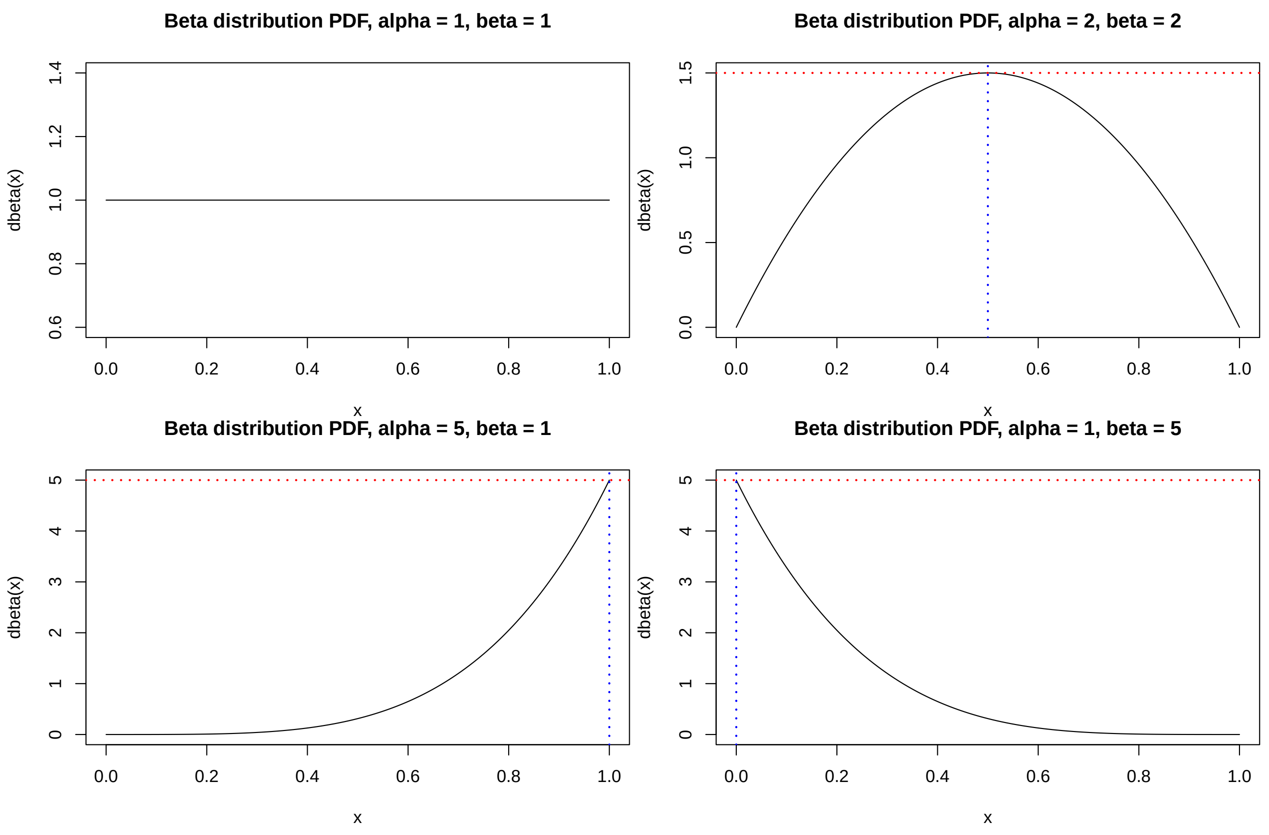
Among the candidates for the instrumental distribution I would consider uniform distribution as well as the beta distribution. There is no need to compute the inverse function for the CDF of the beta distribution as acceptance-rejection method requires knowledge only about the PDF. Let's proceed with the uniform distribution as the instrumental one, though it might yield worse results in terms of rejection rate, compared to the beta-distribution.

Other distributions might be risky to use as the envelope distribution for beta distribution as it is very flexible in terms of it's shape and can mimic uniform, normal and exponential distributions depending on parameters alpha and beta.

In order to ensure the smallest possible value of the c and in turn ensure the smallest possible rejection rate for the chosen instrumental distribution, we can use the maximum value of the beta distribution.

To illustrate it we can plot beta distribution with different parameters alpha and beta. In order to ensure the condition $f(x) \leq cg(x)$ for all x, we can choose constant c for instrumental uniform distribution $g(x)$, which satisfies this condition.

```r
x <- seq(0, 1, 0.0001)

plot(x, dbeta(x, 1,1), type = 'l', main = "Beta distribution PDF, alpha = 1, beta = 1", ylab = "dbeta(x

plot(x, dbeta(x, 2,2), type = 'l', main = "Beta distribution PDF, alpha = 2, beta = 2", ylab = "dbeta(x
abline(c(1.5, 0), col = "red", lty=3, lwd=2)
abline(v = 0.5, col = "blue", lty=3, lwd=2)

plot(x, dbeta(x, 5,1), type = 'l', main = "Beta distribution PDF, alpha = 5, beta = 1", ylab = "dbeta(x
abline(h = 5, col = "red", lty=3, lwd=2)
abline(v = 1, col = "blue", lty=3, lwd=2)

plot(x, dbeta(x, 1,5), type = 'l', main = "Beta distribution PDF, alpha = 1, beta = 5", ylab = "dbeta(x
abline(h = 5, col = "red", lty=3, lwd=2)
abline(v = 0, col = "blue", lty=3, lwd=2)
```



Finding a good constant to keep the rejection proportion small

To identify, whether the optimal constant c has been chosen, we can choose different constants and compare the sample rejection rate for different c.

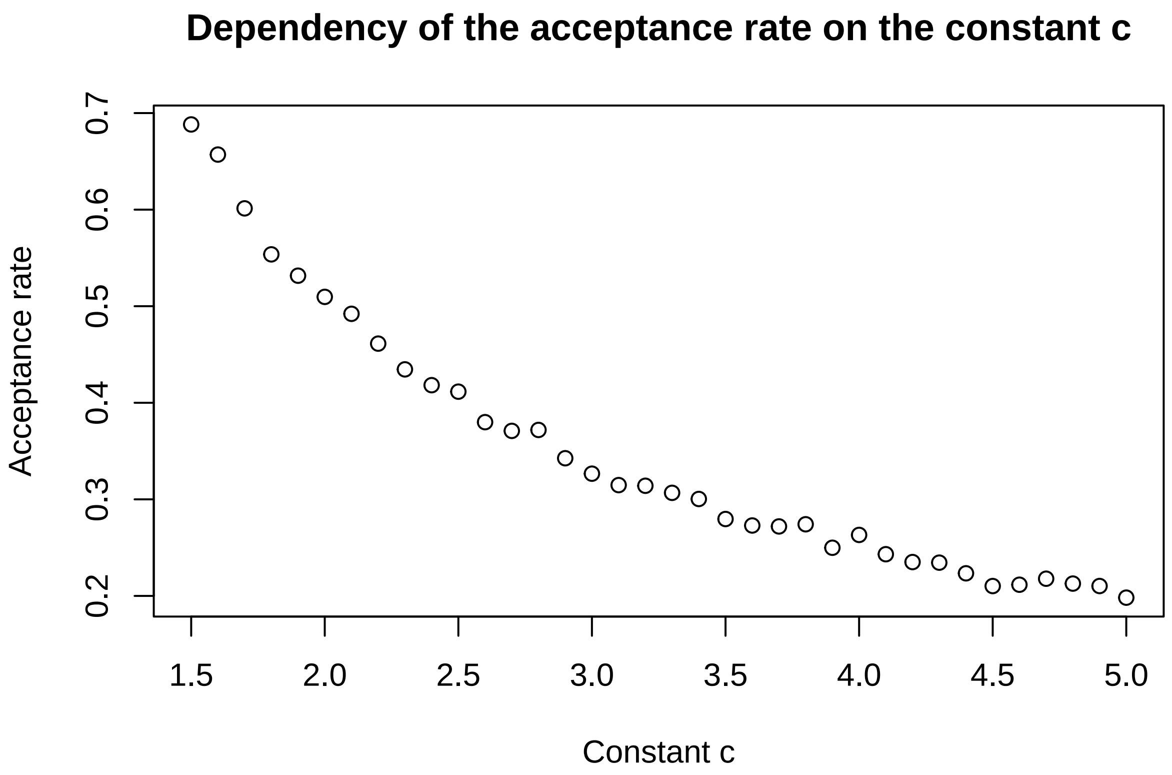
Description of the approach:

- Generate linear space sample for constant c.
- For each constant c generate 1000 samples based on the acceptance-rejection algorithm implemented above with alpha $= 2$, beta $= 2$ and calculate the acceptance rate

It's to be expected that the higher the c, the lower is the acceptance rate because of the relations between the target and the instrumental distribution.

```r
test_c <- seq(from = 1.5, to = 5, by = 0.1)

acc_rate <- c()
for(i in 1:length(test_c)) {
  k <- 0
  k <- norm_accept_reject(1000, test_c[i], 2, 2)
  acc_rate <- append(acc_rate, k[[1]])
}

comp <- data.frame(cbind(test_c, acc_rate))
```

```r
plot(comp$test_c, comp$acc_rate, main = "Dependency of the acceptance rate on the constant c", xlab = "C
```



###

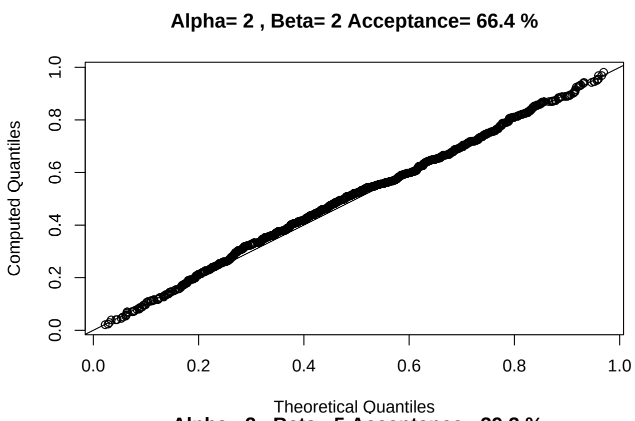
Function for arbitrary beta distribution

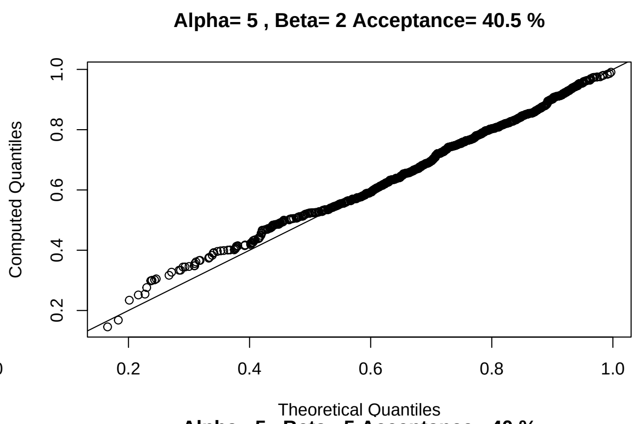
We define the function for acceptance-rejection algorithm, where constant c is defined based on the maximum value of the of beta probability density function. It ensures that we choose the smallest value for constant c possible.

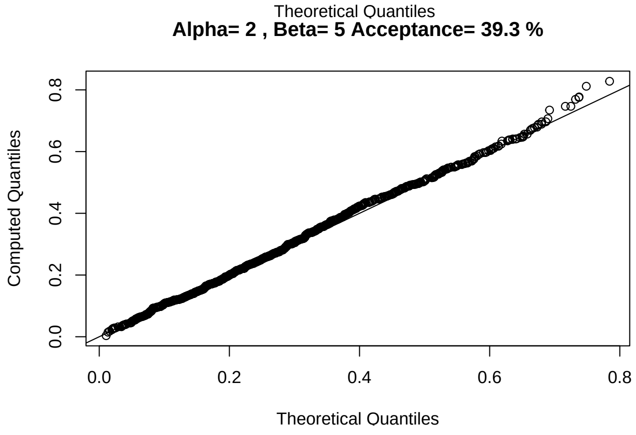
```r
norm_accept_reject_arbitrary <- function(n_samples_ar, alpha, beta){
iter <- 0
accepted <- 0
x <- numeric(n_samples_ar)
seq_x <- seq(0, 1, 0.01)
c <- max(dbeta(seq_x, alpha, beta))
#print(c)
while(accepted<n_samples_ar)
{
u <- runif(1)
iter <- iter + 1
y <- runif(1)
if (dbeta(y, alpha, beta)/(c*dunif(y)) >= u)
  {
  accepted <- accepted+1
  x[accepted] <- y
  }
}
acc_rate <- (accepted/iter)
return(list(acc_rate, x))
}
```

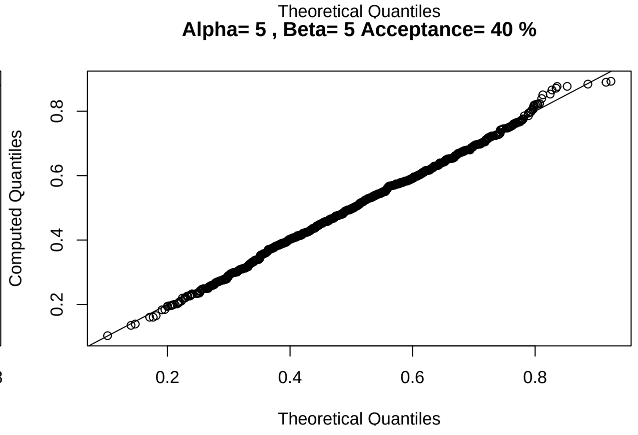
In order to identify the similarity between the generated sample from the beta distribution compared to the theoretical beta distribution we can use qq-plots for various values of parameters alpha and beta.

```r
l_params_beta <- list()
l_params_beta[[1]] <- list(2, 2)
l_params_beta[[2]] <- list(5, 2)
l_params_beta[[3]] <- list(2, 5)
l_params_beta[[4]] <- list(5, 5)

for(i in 1:length(l_params_beta)){
  alpha <- 0
  beta <- 0
  alpha <- l_params_beta[[i]][[1]]
  beta <- l_params_beta[[i]][[2]]

  res_acc <- norm_accept_reject_arbitrary(1000, alpha, beta)

  plot_n <- paste("Alpha=", toString(alpha), ", Beta=", toString(beta), "Acceptance=", round(res_acc[[1
  qqplot(res_acc[[2]], rbeta(1000, alpha, beta), xlab="Theoretical Quantiles", ylab="Computed Quantiles
  abline(c(0, 1))
}
```









qq-plots above illustrate that the acceptance-rejection method with automatic identification of optimal value of constant c provides relatively good sample, comparable to the distribution drawn from the thoeretical one.