1b.) Inverse Transform Sampling is a method of pseudo-random number sampling from any probability distribution given its cumulative distribution function (CDF).

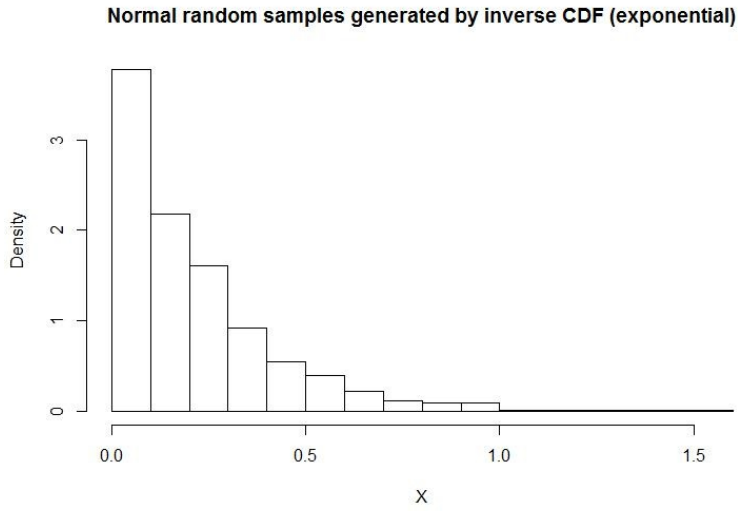
The steps followed are –

- uniformly sample a number "u" between 0 and 1, interpreted as a probability, and -- return the largest number "x" from the domain of the distribution p(X) such that
  p(-inf < X < x) <= u.

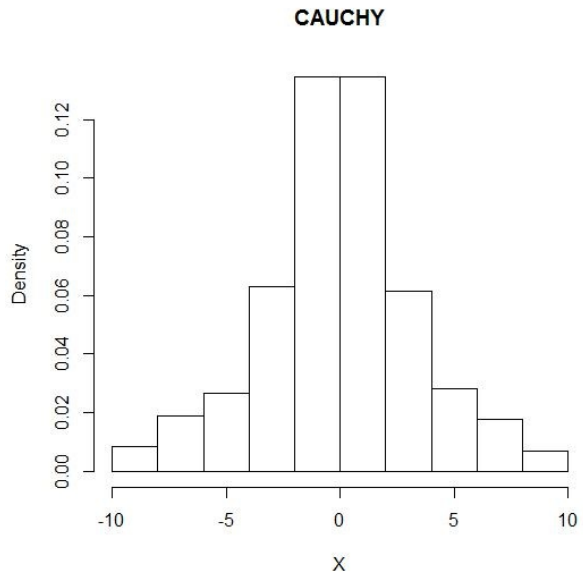
- This involves computing the quantile function of the distribution, which is nothing but the CDF.

Here, 1000 samples from X~exp(5) and Y~Cauchy(0,2) are generated, and then the chi-square goodness of fit test is used to determine the validity of those samples – to determine if they indeed belong to their respective probabilistic distribution or not.

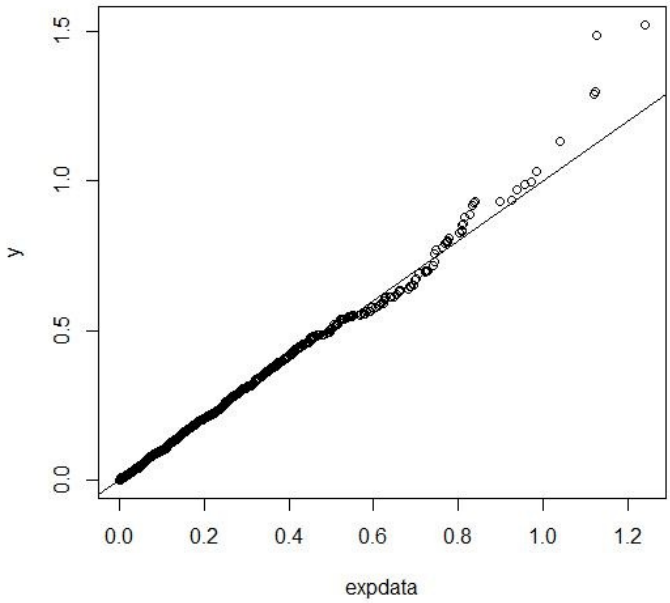
Another method called "QQ plots" is used to determine the goodness-of-fit. Q-Q plot is a graphical method to compare two probability distributions by plotting their quantiles against each other. If the two distributions being compared are similar, the points in the Q-Q plot pretty much follow a line-like shape ; else they take a curved plot appearance. Hence, a line is fit against the plot to check for any major deviation in the form of the plot.



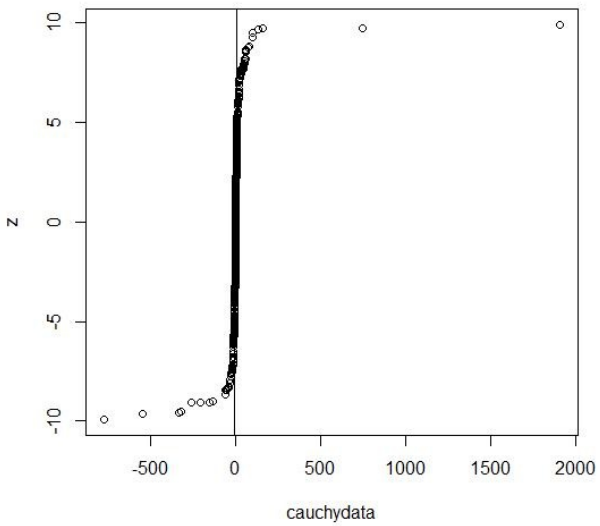
Histogram of random samples from exponential distribution



Histogram of random samples from Cauchy distribution



Q-Q plot of random samples from exponential distribution



Q-Q plot of random samples from Cauchy distribution

The chisquare value for exponential random samples generated = 0.09092481
The chisquare value for Cauchy random samples generated = 804.126

The chi square values are reasonably small ,and looking at the Q-Q plots as well as the histograms, one can infer that the samples belong to the exponential and Cauchy distributions respectively.

CODE :

```
#Function to generate random samples by inverse CDF method

################ Random samples from exponential distribution
##############
 g <- function(x, lambda) {
                                        rv <-  - 1/lambda * log(1 - x)
                                        return (rv)
                                        }

#Function call - main function .
 lambda <- 5
 y <- g(runif(1000, min=pexp(-10, rate=lambda), max=pexp(10, rate=lambda)), lambda)



################ Random samples from Cauchy distribution
##################
 cauc <- function(x, sc) {
                                        rvc <-  sc * tan (pi * (x-0.5))
                                        return (rvc)
                                        }

#Function call - main function .
 loc <- 0
 scale <- 2
set.seed(2)
 z <- cauc(runif(1000, min=pcauchy(-10, location = 0, scale =2, lower.tail = TRUE,
log.p = FALSE), max=pcauchy (10, location = 0, scale=2, lower.tail = TRUE, log.p =
FALSE)), scale)

#generating the histogram
#generating the histogram
hist(y,prob=TRUE,xlab='X',main='Normal random samples generated by inverse CDF
(exponential)')
x11()
hist(z,prob=TRUE,xlab='X',main='CAUCHY')



########----------------------------------------------------------------
######## chi- square goodness of fit tests.


set.seed(2)
cauchydata <- rcauchy(1000, location = 0, scale = 2)
ChiCauchy <- (sum(z - cauchydata) ^2 )/ sum (cauchydata)
ChiCauchy

set.seed(2)
expdata <- rexp(1000, rate = 5)
```

```
ChiExponential <- (sum(y - expdata) ^2)/sum(expdata)
ChiExponential

##########
x11()
qqplot(expdata,y)
abline(0,1)
x11()
qqplot(cauchydata,z)
abline(0,1)
```

1c) Random Sampling using Acceptance-Rejection Method

Method : A RV having density function "g(x)" can be used as a basis for generating 'Y' from "g" and then accepting this generated value with a probability proportional to f(Y)/ g(Y).

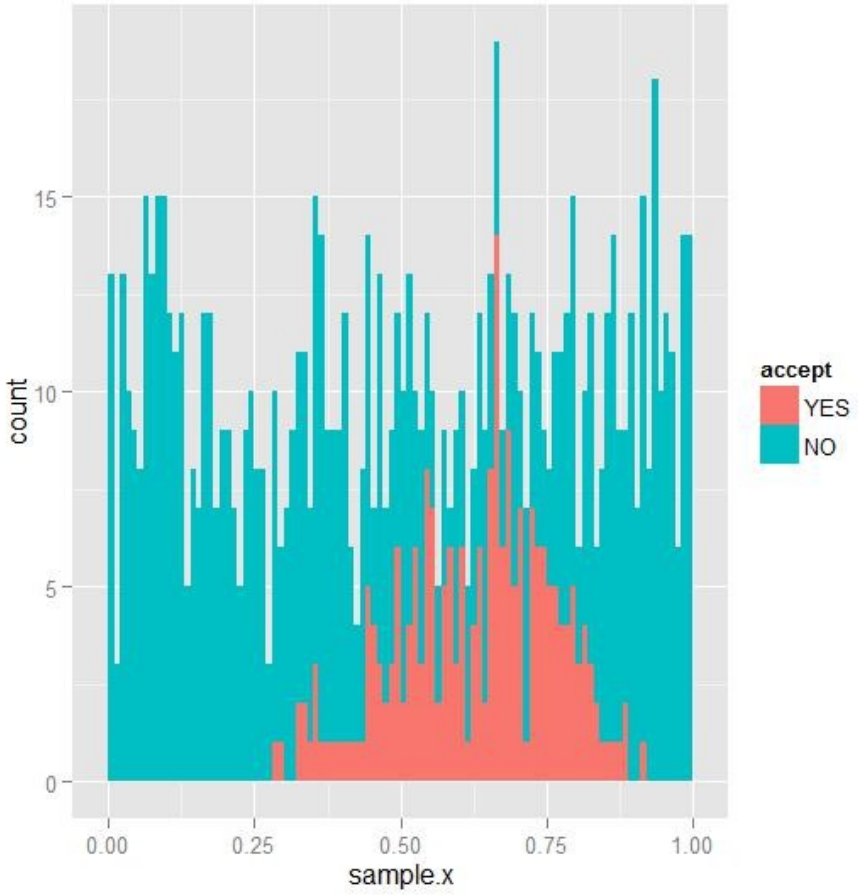
If "c" is a constant,

$$f(y)/g(y) <= c \text{ ; for all y}$$

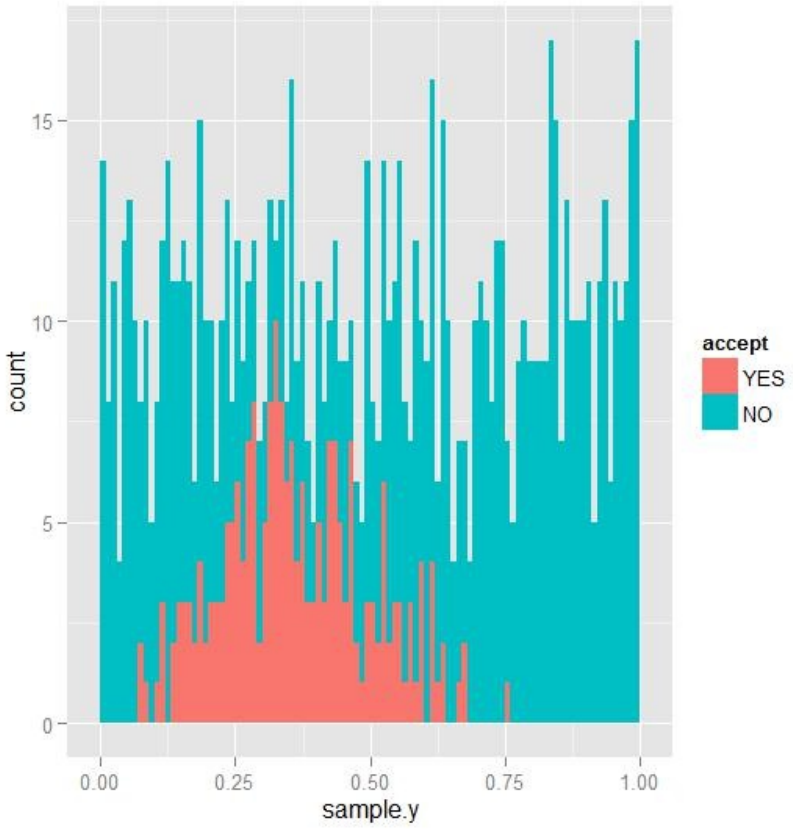
Then
Step 1 – generate "Y" having density "g".
Step 2 – generate a random number U.
Step 3 – If U = f(Y)/cg(Y) , X = Y. Else, return to Step 1.

This technique is applied to two Beta distributions B1(8,5) and B2(4,7).

It is noticed that majority of the samples generated have been rejected. The plot of actually generated samples vs rejected samples is as shown –



For Beta (8,5)



For Beta (4,7)

Next, we determine the independency between the samples generated above between B1(8,5) and B2(4,7).

This is done using Fisher's exact test / 2-Way Contingency Table.

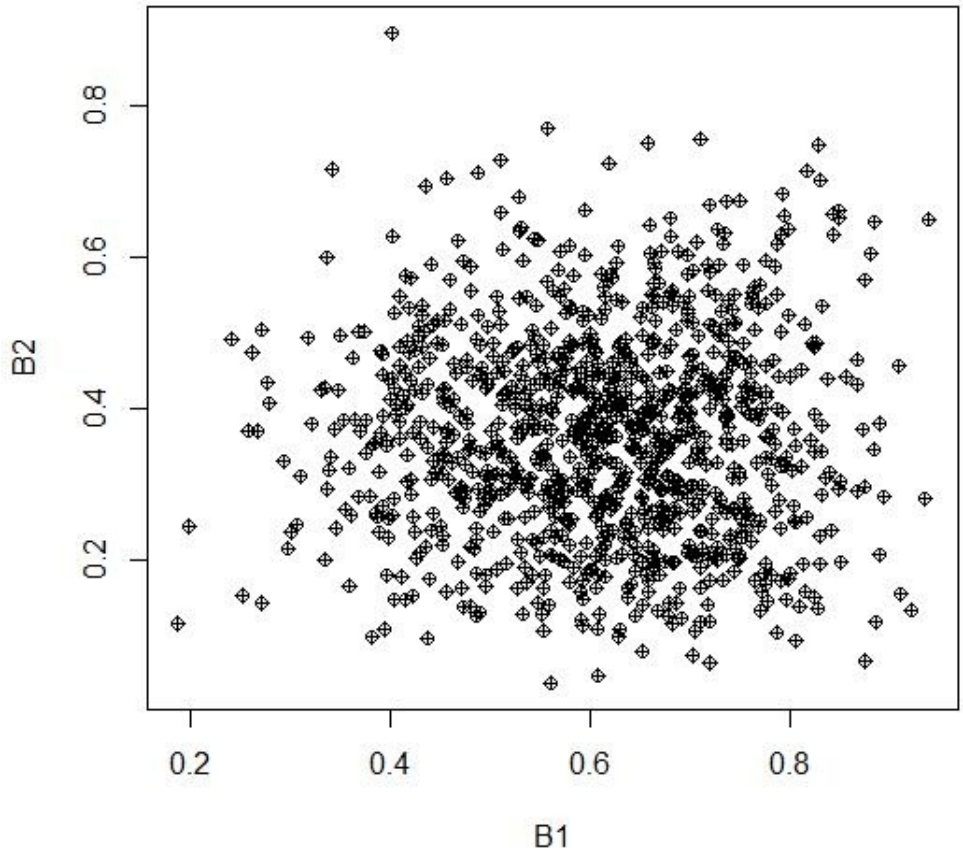
	(0.0361,0.251)	(0.251,0.466)	(0.466,0.681)	(0.681,0.896)
(0.186,0.375)	10	22	9	1
(0.375,0.563)	57	183	61	6
(0.563,0.752)	113	275	116	3
(0.752,0.94)	35	73	32	4

The in-built function in R , "chisq.test" can be used to perform the test of independence.
The Null Hypothesis is that B1 and B2 are independent.

The p-value derived from the above test = 0.3629
Since this is above 0.05, the Null hypothesis holds.

Thus, B1 and B2 are independent.

This is further corroborated using a covariance plot between B1 and B2 –



Clearly, the covariance plot doesn't have an increasing or a decreasing trend, and the trend is scattered across the plot. This shows that the two distributions generated do not affect each other in any way.

CODE:

```
### Problem 3 - Random Sampling using Beta function - Acceptance/Rejection
method ###

#---------------------------------------------------------------------------
#### Beta function with parameters = 8 and 5
sample.x <- runif(1000,0,1)
accept = c()
set.seed(2)
for ( i in 1: length(sample.x))
{
        U = runif(1, 0 ,1)

        if (dunif(sample.x[i],0,1)* 5 * U <= dbeta(sample.x[i],8,5))
        {
                accept[i] = 'YES'
        }
        else
        {
                accept[i] = 'NO'
        }
}

T = data.frame (sample.x , accept = factor (accept, levels = c('YES','NO')))
library(ggplot2)
print(qplot(sample.x, data = T, geom = 'histogram', fill = accept, binwidth=0.01))
#---------------------------------------------------------------------------
#### Beta function with parameters = 4 and 7
sample.y <- runif(1000,0,1)
accept = c()
set.seed(2)
for ( i in 1: length(sample.y))
{
        U = runif(1, 0 ,1)

        if (dunif(sample.y[i],0,1)* 5 * U <= dbeta(sample.y[i],4,7))
        {
                accept[i] = 'YES'
        }
        else
        {
                accept[i] = 'NO'
        }
}

T = data.frame (sample.y , accept = factor (accept, levels = c('YES','NO')))
library(ggplot2)
x11()
print(qplot(sample.y, data = T, geom = 'histogram', fill = accept, binwidth=0.01))

######################### Testing for independence of X and Y
####################
set.seed(2)
```

```
B1 <- rbeta(1000, shape1=8, shape2=5)
B2 <- rbeta(1000, shape1=4, shape2=7)
cov(B1,B2)
x11()
plot (B1,B2,pch=10)
CT <- table(cut(B1,4), cut(B2,4))
CT
chisq.test(CT)
```