Question 1. Construct a Uniform Random Number Generator between 0 and 1 using the Linear Congruential Generator (LCG) for the choice of a priori integers due to Lehmer (1951), viz. Multiplier = 23, Shift = 0 and Modulus = $10^8 + 1$.

Generate random samples each of size 1000 from a $U(0,1)$ distribution using your own generator. Check **validity** of the generated random samples.

Solution:

Linear Congruential Generator (LCG):

$$X_i = (aX_{i-1} + c)\ mod\ M$$

Where a = multiplier

c = shift

M = modulus, and the first element is denoted as X_0 and is called seed value (priori).

Some restrictions are imposed on the parameters of the generator:

$$M > 0, 0 \leq a < M, 0 \leq c < M, 0 \leq X_0 < M$$

R Code:

```
rm(list=ls())
LGC <- function(a, c, M, n, y){
  x <- c()
  x[1] <- y
  for(i in 1:n)
    x[i+1] <- (a*x[i]+c) %% (M)
  return(x[-1]/M)
}

uni_01 <- LGC(a= 23, c= 0, M= 10^8, n= 1000, y= 123)
```

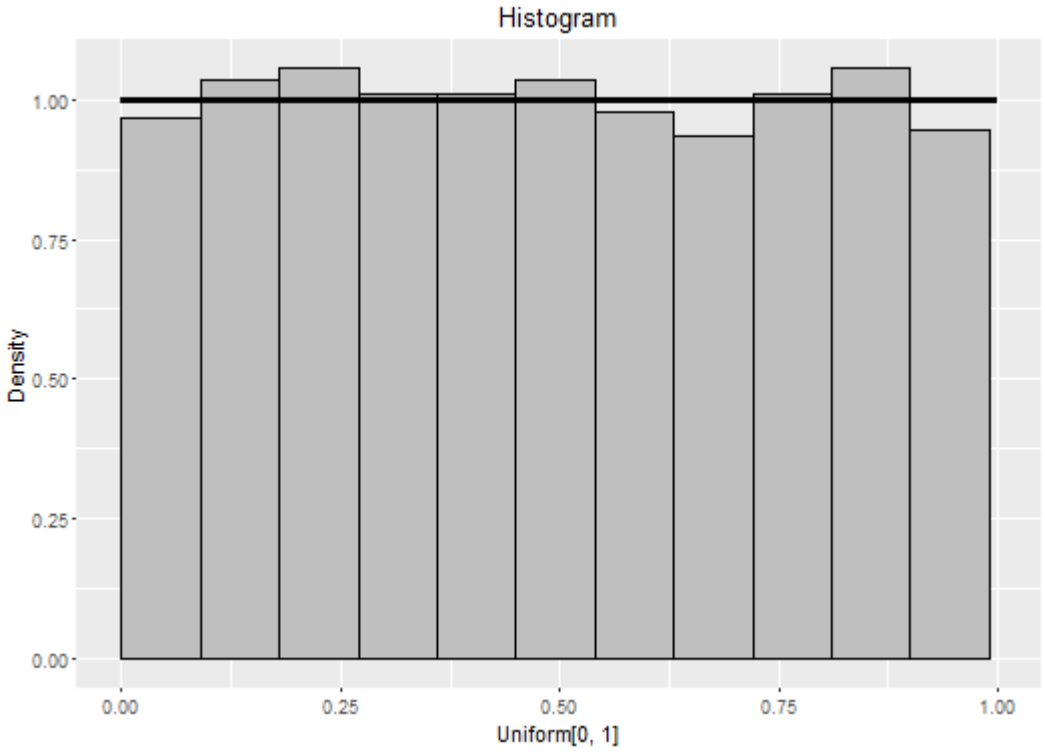
Conclusion: "**LCG**" function can be called to generate random numbers from U[0, 1] distribution using linear Congruential Generator (LCG). For given set of parameters 1000 random numbers are generated and are given in the appendix.

We will check **validity** of our random sample:

1. Histogram overlaid with a theoretical frequency curve:

R code:

```
library(ggplot2)
p <- ggplot(mapping = aes(x = uni_01))
p <- p + geom_histogram(aes(y=..density..), binwidth = .09,
                        colour="black", fill="gray75")
p <- p + stat_function(fun=dunif, args = list(min=0, max=1),
                           col = 'black', lwd=1.2) + xlim(0,1)
p + xlab("Uniform[0, 1]") + ylab("Density") + ggtitle("Histogram")
```

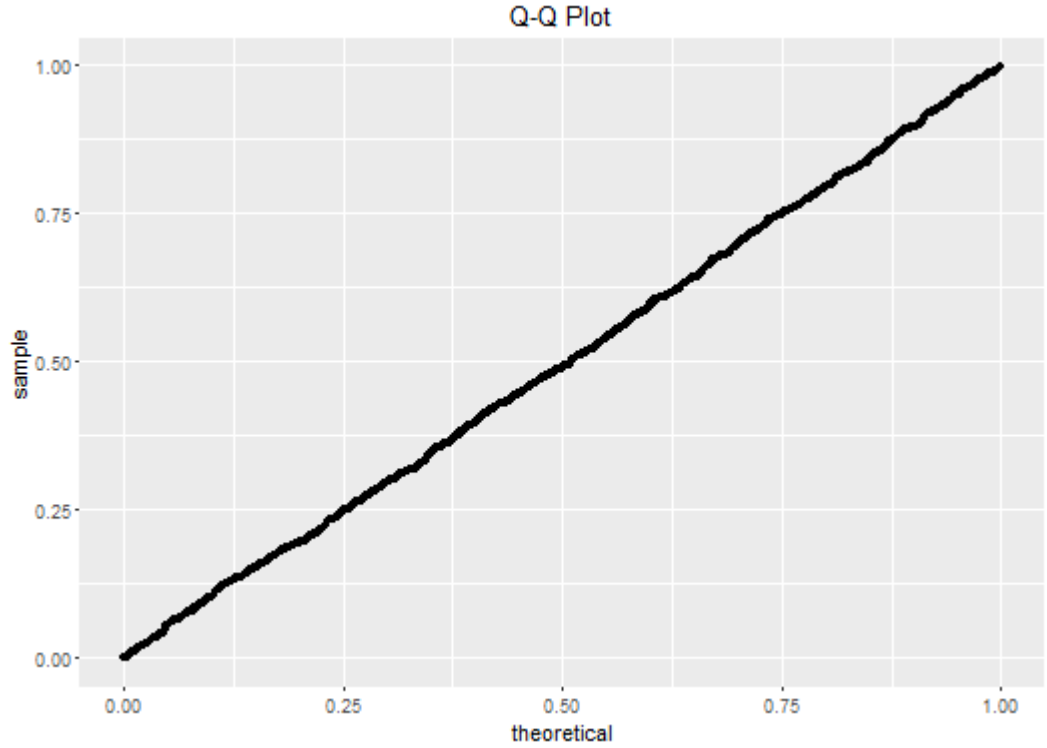


Conclusion: We see that observed sample density is close to expected density. No skewness is observed.

2. Q-Q Plot:

R Code:

```
q <- ggplot(mapping = aes(sample = uni_01)) + ggtitle("Q-Q Plot")
q + stat_qq(distribution = stats::qunif)
```



Conclusion: We observe that sample and theoretical quantiles lies along a straight line. No skewness is observed.

3. Chi - Square Test:

H_0: The data are consistent with the uniform distribution.
H_1: The data are not consistent with the uniform distribution.

R Code:

```
breaks <- seq(0, 1, length = 6)
intervals <- cut(uni_01, breaks, right=FALSE)
observed <- table(intervals)
prob <- c()
for(i in 1:5)
  prob[i] <- punif(breaks[i+1], 0, 1) - punif(breaks[i], 0, 1)
expected <- prob*1000
chisq.test(cbind(observed, expected))
```

Output:

```
        Pearson's Chi-squared test

data:  cbind(observed, expected)
X-squared = 0.40835, df = 4, p-value = 0.9818
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data fits well to Uniform[0, 1].

4. Kolmogorov-Smirnov Test:

H_0: Samples are from same distribution.
H_1: Samples are not from same distribution.

R Code:

```
ks.test(uni_01, "punif", 0, 1)
```

Output:

```
        One-sample Kolmogorov-Smirnov test

data:  uni_01
D = 0.015665, p-value = 0.9668
alternative hypothesis: two-sided
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data is coming from Uniform[0, 1].

All the validity test holds true, hence we conclude that Lehmer Linear Congruential Generator is a valid generator in generating random sample from Uniform[0, 1].

Question 2:

(i) Generate a random sample of size 1000 from a $N(7,3)$ distribution using a suitable algorithm. Check **validity** of the generated random sample. Now assuming that the generated random sample is a random sample from some $N(\mu, \sigma^2)$ distribution, find MLEs of the parameters and comment on your results.

Solution:

We will generate the required sample by generating 1000 random numbers from N(0, 1) and then transforming into N(7, 3). We will generate random sample from N(0, 1) by Acceptance/Rejection method. Algorithm is described as follows:

1. Generate X_1 from h(.) and U_1 from U(0, 1), independently. h(.) will be pdf of double exponential distribution. To generate data from double exponential, we generate "n" numbers from Exponential(1) distribution and take negative for half of them. To generate data from exponential distribution we use simple inversion method:

$$x = F^{-1}(u) = -\ln(1-u)$$

2. We will check if the following condition is satisfied:

$$cU_1 h(X_1) \leq f_y(X_1)$$

Where $c = \sqrt{\frac{2}{\pi}} e^{\frac{1}{2}}$ and $f_y(X_1)$ is the pdf from which we want to generate our random sample.

If satisfied then that X_1 will be our first data point from $f_y(.)$. If not, then go back to step 1.

3. Repeat step 1 and step 2as many times as the data points are required.
4. After generating sample from N(0, 1), we will transform them into N(7, 3) by following transformation:

$$Y = 7 + \sqrt{3} * Z$$

Where Z ~ N(0, 1) and Y ~ N(7, 3).

R Code:

```
set.seed(122)
rand_snormal <- function(n){
  uni_01 <- runif(2*n)
  y1 <- uni_01[1:n]
  y2 <- uni_01[n+1:1000]
  y <- c(y1, y2)
  x <- c()
  for(i in 1:n){                    # Random number from double
exponential(1)
    x[i] <- -log(1-y1[i])
    x[n +i] <- log(1-y2[i])
  }
  p <- c()
  i<-1
  while(length(p) < 1000){          # Acceptance/Rejection condition
    U = sample(uni_01, 1)
    z = sample(x, 1)
    if(exp(-abs(z))*1.315*U/2 <= dnorm(z, 0, 1)){
      p[i] <- z
      i <- i+1
    }
```

```
    else i <- i-1
  }
  return(p)
}

norm_01 <- rand_snormal(1000)                    # Generating from N(0, 1)
norm <- norm_01*sqrt(3) + 7                  # Transforming into N(7, 3)

mean(norm)
[1] 7.011245
> var(norm)
[1] 2.893641
```

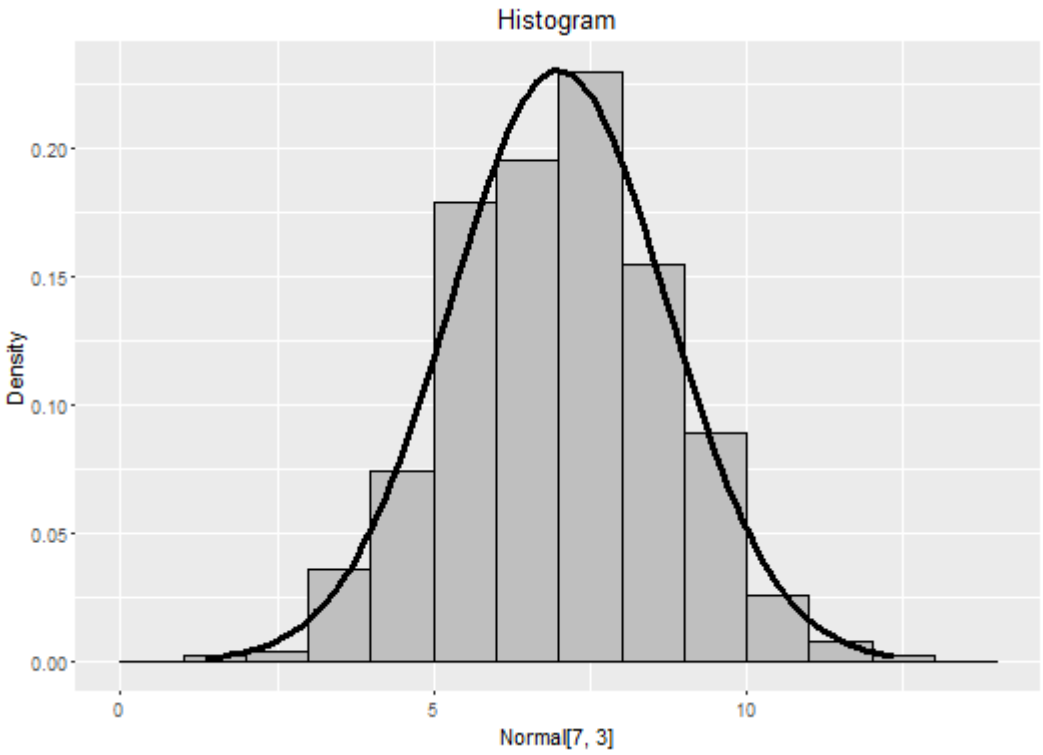
Conclusion: "**norm_01**" function can be called to generate random numbers from N[0, 1] distribution using Acceptance/Rejection Method. 1000 random numbers are generated and are given in the appendix.
MLE of population mean is given by the **sample mean** = 7.011
MLE of population variance is given by the **sample variance** = 2.893

We will check validity of our random sample:

1. Histogram overlaid with a theoretical frequency curve:

R Code:

```
library(ggplot2)
p <- ggplot(mapping = aes(x = norm))
p <- p + geom_histogram(aes(y=..density..), binwidth = 1, colour="black",
                        fill="gray75")
p <- p + stat_function(fun=dnorm, args = list(mean=7, sd=sqrt(3)),
                       col = 'black', lwd=1.2)
p + xlab("Normal[7, 3]") + ylab("Density") + ggtitle("Histogram")
```

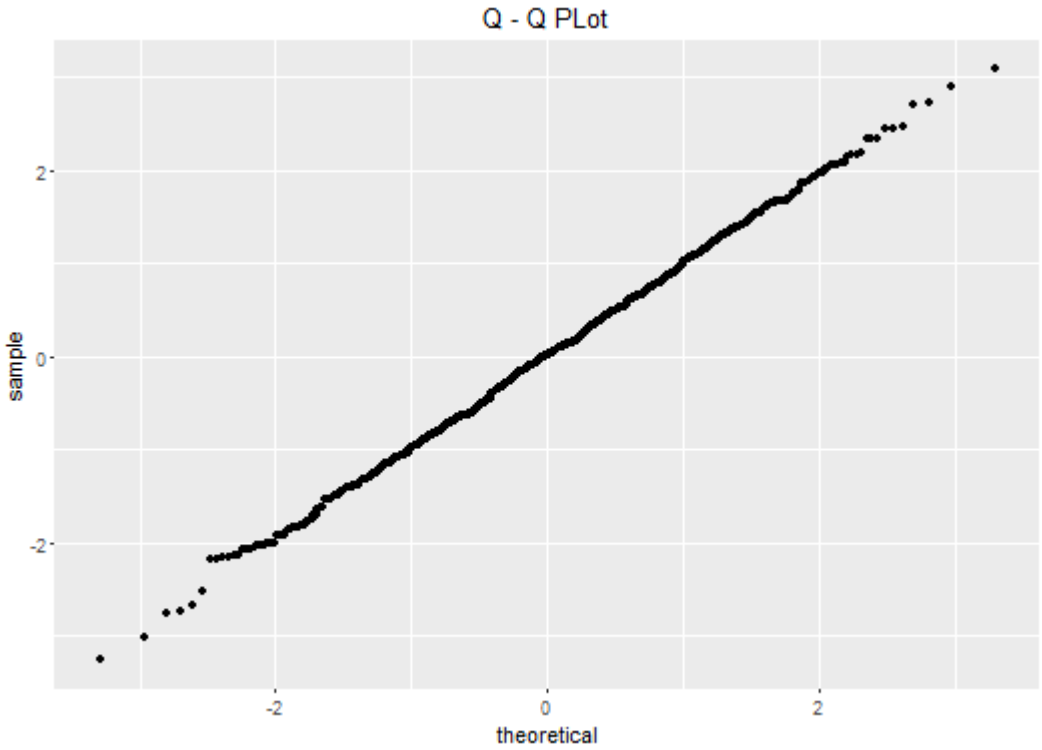


Conclusion: We see that observed sample density is close to expected density. No skewness is observed.

2. Q-Q Plot:

R Code:

```
q <- ggplot(mapping = aes(sample = norm_01))
q + stat_qq(distribution = stats::qnorm) + ggtitle("Q - Q PLot")
```



Conclusion: We observe that sample and theoretical quantiles lies along a straight line, i.e. they coincides.

3. Chi - Square Test:

H_0: The data are consistent with the normal distribution.
H_1: The data are not consistent with the normal distribution.

R Code:

```
breaks <- seq(min(norm), max(norm), length = 6)
intervals <- cut(norm, breaks, right=FALSE)
observed <- table(intervals)
prob <- c()
for(i in 1:5)
  prob[i] <- pnorm(breaks[i+1], 7, sqrt(3)) - pnorm(breaks[i], 7,
sqrt(3))
expected <- prob*1000
chisq.test(cbind(observed, expected))
```

Output:

```
        Pearson's Chi-squared test

data:  cbind(observed, expected)
X-squared = 0.20693, df = 4, p-value = 0.995
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data fits well to N[7, 3].

4. Kolmogorov-Smirnov Test:

H_0: The samples are from the same population.
H_1: The samples are not from the same population.

R Code:

```
ks.test(norm, "pnorm", mean=7, sd=sqrt(3))
```

Output:

```
        One-sample Kolmogorov-Smirnov test

data:  norm
D = 0.021345, p-value = 0.7524
alternative hypothesis: two-sided
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data is coming from N[7, 3].

All the validity test holds true, hence we conclude that our random number generator is a valid generator in generating random sample from N[0, 1], which can be transformed into N[7,3].

Question 2

(ii) Generate a random sample of size 1000 from a Gamma(5.8) distribution using a suitable algorithm. Check **validity** of the generated random sample. Now assuming that the generated random sample is a random sample from some Gamma(α) distribution, find MLE of the parameter and comment on your results.

Solution:
Here we will use the fact that, sum of "n" independent exponential distribution follows Gamma distribution and sum of Gamma distribution is also a Gamma distribution. So we will generate Gamma(5) from exponential distribution and Gamma(0.8) from Acceptance/Rejection method and then we will add them up.

Let X_1, X_2, X_3, X_4, X_5 be 5 independent observation from Exponential(1) distribution, then:

$$\sum_{i=1}^{5} X_i \sim Gamma(5)$$

Acceptance/Rejection Method:

1. Generate X_1 from h(.) and U_1 from U(0, 1) independently such that h(.) is generated from:

$$F_X^{-1}(u) = \begin{cases} \left(uc\Gamma(\alpha+1)\right)^{\frac{1}{\alpha}}, & 0 < u < \dfrac{1}{\Gamma(\alpha+1)} \\ -\ln\left\{\left(\dfrac{1}{c\Gamma(\alpha+1)} - u\right)c\Gamma(\alpha) + \dfrac{1}{e}\right\}, & u > \dfrac{1}{\Gamma(\alpha+1)} \end{cases}$$

2. Check, if $cU1h(X_1) \leq f_y(X_1)$ is true then $Y = X_1$ and if false then go back to step 1, where:

$$f_y(x) = \frac{1}{\Gamma(\alpha)} e^{-x} x^{\alpha-1}, \quad x > 0, 0 < \alpha < 1$$

$$h(x) = \begin{cases} \dfrac{1}{c\Gamma(\alpha)} x^{\alpha-1} & 0 < x < 1 \\ \dfrac{1}{c\Gamma(\alpha)} e^{-x} & x \geq 1 \end{cases} \qquad and \qquad c = \frac{1}{\Gamma(\alpha)}\left(\frac{1}{\alpha} + \frac{1}{e}\right)$$

3. Repeat step 1 and step 2as many times as the data points are required.

R code:

```
set.seed(1234)

uni_01 <- runif(2000, 0, 1)            # Random number from U[0, 1]
y <- -log(uni_01)                      # Random number from exp(1)

z <- c()
for(i in 1:2000)
  z[i] <- sum(sample(y, 5))      # Random number from gamma(5)

rand_gamma <- function(n, alpha){ # Random number from gamma(alpha)
  c <- (1/alpha + 1/exp(1))/gamma(alpha)
  x <- c()
  for(i in 1:(2*n)){
    if(uni_01[i] < 1/(c*gamma(1 + alpha)))
      x[i] <- (uni_01[i]*c*gamma(1 + alpha))^(1/alpha)
    else if(uni_01[i] > 1/(c*gamma(1 + alpha)))
      x[i] <- -log((1/(c*gamma(1 + alpha))-uni_01[i])*c*gamma(alpha) +
                  1/exp(1))
  }
  x <- na.omit(x)

  i <- 1
  p<-c()
  while(length(p) < n){
    u <- sample(uni_01, 1)
    if(x[i] < 1){
      if(u*(x[i]^-0.2)/gamma(alpha) < dgamma(x[i], shape=alpha)){
        p[i] <- x[i]
        i <- i + 1
      }
      else i <- i - 1
    }
    else if(x[i] > 1){
      if(u*(exp(-x[i]))/gamma(alpha) < dgamma(x[i], shape=alpha)){
        p[i] <- x[i]
        i <- i + 1
      }
      else i <- i - 1
    }
  }
  return(p)
}
gam <- rand_gamma(1000, 0.8)
gam <- gam + z

> mean(gam)
[1] 5.889198
> var(gam)
[1] 5.967602
```

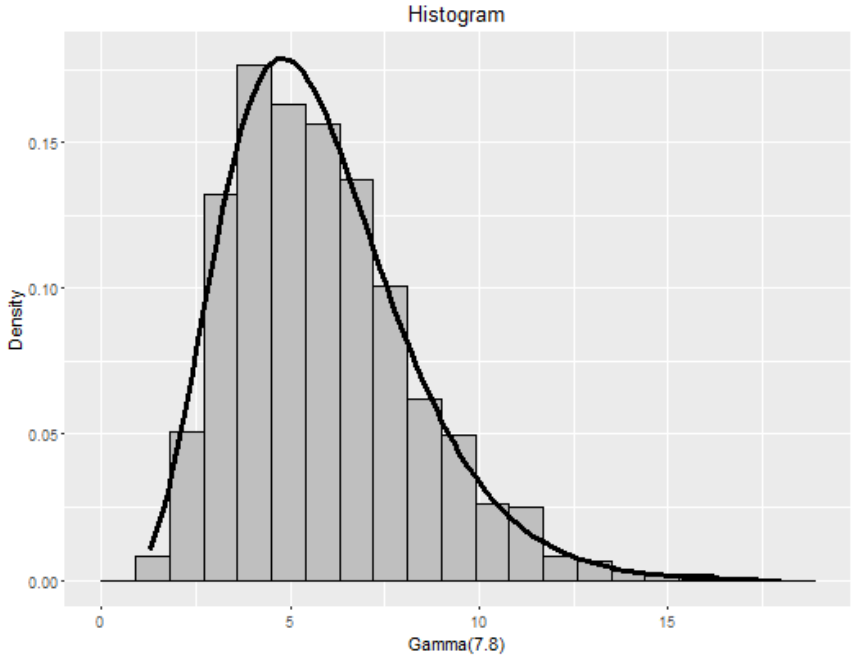
Conclusion: "**rand_gamma**" function can be called to generate 1000 random numbers from gamma(0.8) distribution, the numbers generate here are given in the appendix, then these numbers can be added to gamma(5) random number so as to obtain a random sample from gamma(5.8) distribution.

We will check validity of our random sample:

1. Histogram overlaid with a theoretical frequency curve:

R code:

```
library(ggplot2)
p <- ggplot(mapping = aes(x = gam))
p <- p + geom_histogram(aes(y=..density..), binwidth = 0.9, colour="black",
fill="gray75")
p <- p + stat_function(fun=dgamma, args = list(shape=5.8), col = 'black',
lwd=1.2)
p + xlab("Gamma(7.8)") + ylab("Density") + ggtitle("Histogram")
```

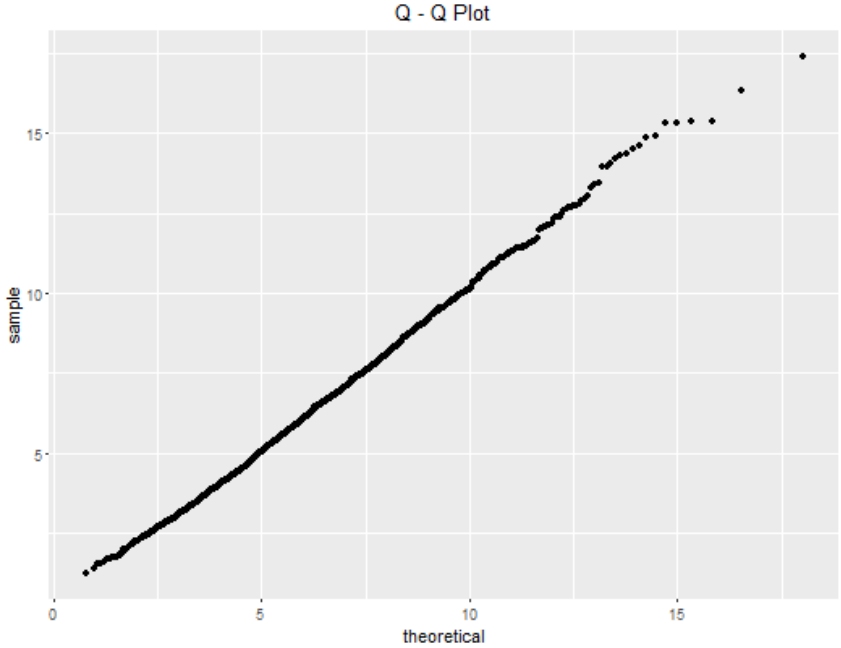


Conclusion: We see that observed sample density is close to expected density. Data is skewed to the right.

2. Q-Q Plot:

R code:

```
q <- ggplot(mapping = aes(sample = gam))+ ggtitle("Q-Q Plot")
q + stat_qq(distribution = stats::qgamma, dparams = list(shape=5.8))
```



Conclusion: We observe that sample and theoretical quantiles lies along a straight line, i.e. they coincides.

3. Chi-Square Test:

H_0: The data are consistent with the gamma distribution.
H_1: The data are not consistent with the gamma distribution.

R code:

```
breaks <- seq(0, max(gam), length = 6)
intervals <- cut(gam, breaks, right=FALSE)
observed <- table(intervals)
prob <- c()
for(i in 1:5)
  prob[i] <- pgamma(breaks[i+1], 5.8) - pgamma(breaks[i], 5.8)
expected <- prob*1000
chisq.test(cbind(observed, expected))
```

Output:

```
        Pearson's Chi-squared test

data:  cbind(observed, expected)
X-squared = 1.0292, df = 4, p-value = 0.9053
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data fits well to Gamma(5.8).

4. Kolmogorov-Smirnov Test:

H_0: The samples are coming from the same population.
H_1: The samples are not coming from the same population.

R Code:

```
ks.test(gam, "pgamma", shape=5.8)
```

Output:

```
        One-sample Kolmogorov-Smirnov test

data:  gam
D = 0.021098, p-value = 0.3355
alternative hypothesis: two-sided
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data is coming from Gamma(5.8).

All the validity test holds true, hence we conclude that our random number generator is a valid generator in generating random sample from Gamma(0.8), which can be transformed into Gamma(5.8).

Question 3

(i) Generate a random sample of size 1000 from a $N_2\left[\begin{pmatrix}9\\2\end{pmatrix}, \begin{pmatrix}8 & 2.3\\2.3 & 5\end{pmatrix}\right]$ distribution using a suitable algorithm. Check validity of the generated random sample using 3-D Histogram overlaid with the Theoretical Bivariate Normal Frequency Curve. Now assuming that the generated random sample is a random sample from some $N_2\left[\begin{pmatrix}\mu_1\\\mu_2\end{pmatrix}, \begin{pmatrix}\sigma_1^2 & \sigma_{12}\\\sigma_{21} & \sigma_2^2\end{pmatrix}\right]$ distribution, find MLEs of the parameters and comment on your results.

Solution: A p-dimension random vector is said to follow a multivariate normal distribution, if pdf is given as:

$$f_y(x;\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right); x \in R^p, \mu \in R^p$$

Here we will generate samples from N[0, 1] and then transform them into bivariate normal sample.

We will follow the following steps:

1. Using Spectral decomposition of Σ we can write: $\Sigma = U\Lambda U^T$, where U is the matrix of p orthonormalized eigenvectors of Σ and $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ is the matrix of corresponding eigenvalues of Σ.
2. Define $A = U\Lambda^{1/2}U^T$, which is completely specified in terms of eigenvalues and eigenvectors of Σ.
3. If $Z \sim N(0, I_P)$ then $AZ + \mu \sim N_P(\mu, AIA^T)$. Then we can define:

$$Y = AZ + \mu$$

Hence we have started with p univariate N(0, 1) random variables as $Z_1, Z_2, \ldots, Z_P$ and then stacking them in a column vector and then multiplying with any non-singular matrix A and adding p-dimensional vector μ.

R code:

```
set.seed(1)
mu <- matrix(c(9, 2), nrow=2, ncol=1)
sigma <- matrix(c(8, 2.3, 2.3, 5), nrow=2, ncol=2)

u <- svd(sigma)$u
eig <- diag(eigen(sigma)$values, 2)

z <- matrix(c(rnorm(1000, 0, 1), rnorm(1000, 0, 1)), nrow=2)
a <- u %*% t(eig^0.5) %*% u
b <- a %*% z

b[1,] <- b[1,] + 9
b[2,] <- b[2,] + 2
> mean(b[1, ])
[1] 8.880603
> mean(b[2, ])
[1] 2.017876
> var(b[1, ])
[1] 9.072315
> var(b[2, ])
[1] 5.089713
> cov(b[1, ], b[2, ])
[1] 2.485204
```

MLE's are:

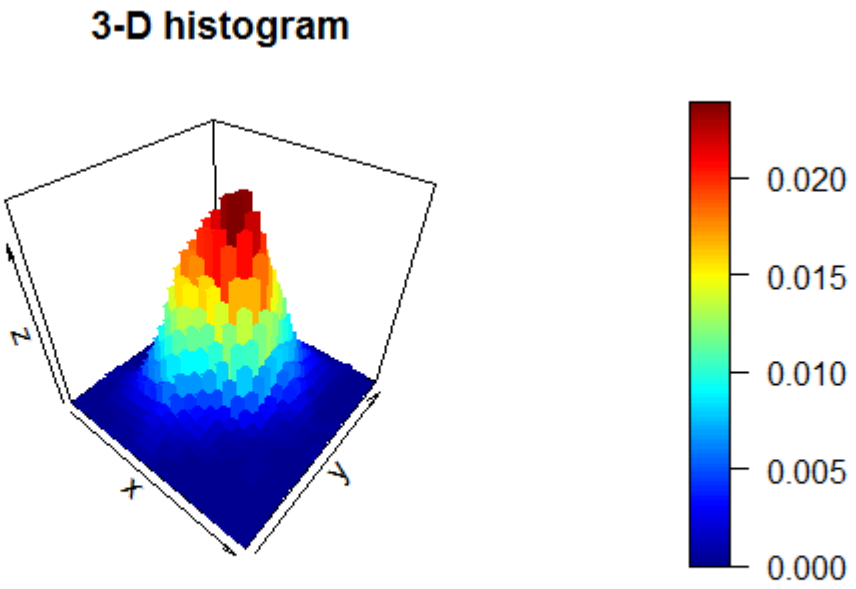
$$\hat{\mu} = \begin{pmatrix}8.88060\\2.01787\end{pmatrix} \text{ and } \hat{\Sigma} = \begin{pmatrix}9.07231 & 2.48520\\2.48520 & 5.08971\end{pmatrix}$$

We will check validity of our random sample:

1. 3-D Histogram:

R code:

```
library(plot3D)
library(MASS)
a <- kde2d(b[1,],b[2,])
hist3D(z=a$z, main="3-D histogram")
```



Question 3

(ii) Generate a random sample of size 1000 from a MOBE (5,5,2) distribution using a suitable algorithm. Suppose $\{Y_{1i}, Y_{2i}: i = 1, \dots, 1000\}$ is the generated sample then observe that the proportions of sample values in each of the three classes viz. $I_1 = \{Y_1 < Y_2\}, I_2 = \{Y_2 < Y_1\}$ and $I_3 = \{Y_1 = Y_2\}$ are in accordance with the model parameters.

Solution: Pdf of Marshall-Olkin Bivariate Exponential Distribution is given by:

$$f_{Y_1Y_2}(y_1, y_2) = \begin{cases} \lambda_1 e^{-\lambda_1 y_1}\,(\lambda_2 + \lambda_3) e^{-(\lambda_2+\lambda_3)y_2}, & y_1 < y_2 \\ \lambda_2 e^{-\lambda_2 y_2}\,(\lambda_1 + \lambda_3) e^{-(\lambda_1+\lambda_3)y_1}, & y_1 > y_2 \\ \dfrac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} e^{-(\lambda_1+\lambda_2+\lambda_3)y} & y_1 = y_2 = y_3 \end{cases}$$

To generate random sample from this distribution:

1. First generate independent samples from $X_1 \sim exp(\lambda_1)$, $X_2 \sim exp(\lambda_2)$ and $X_3 \sim exp(\lambda_3)$.
2. F(x) = 1- $e^{-\lambda x}$, we have used the formula:

$$x = -\frac{log(1-u)}{\lambda}$$

3. Define: $Y_1 = \min(X_1, X_2)$ and $Y_2 = \min(X_1, X_3)$

R Code:

```
set.seed(123)

MOBE <- function(n, l1, l2, l3){
  uni_01 <- runif(n)
  uni_02 <- runif(n)
  uni_03 <- runif(n)

  x1 <- -log(uni_01)/l1
  x2 <- -log(uni_02)/l2
  x3 <- -log(uni_03)/l3

  y1 <- c()
  y2 <- c()
  for(i in 1:1000){
    y1[i] <- min(x1[i], x3[i])
    y2[i] <- min(x2[i], x3[i])
  }
  y <- rbind(y1, y2)
  return(y)
}
y <- MOBE(1000, 5, 5, 2)


> sum(y[1, ] < y[2,])
[1] 415
> sum(y[1, ] > y[2,])
[1] 424
> sum(y[1, ] == y[2, ])
[1] 161
```

Conclusion: The sample is generated and stored in "y". The number of sample points falling in each class is:

$$\mathrm{I}_1 = \{\mathrm{Y}_1 < \mathrm{Y}_2\} = 415;\ \mathrm{I}_2 = \{\mathrm{Y}_2 < \mathrm{Y}_1\} = 424;\ \mathrm{I}_3 = \{\mathrm{Y}_1 = \mathrm{Y}_2\} = 161$$

Further expected number of observation in 3 classes are: 416, 416 and 166 respectively.

Question 3

(iii) Generate a random sample of size 1000 from a BBBE (5,5,2) distribution using a suitable algorithm.

Solution: The pdf of Block and Basu Bivariate Exponential Distribution is given by:

$$f_{Y_1Y_2}(y_1,y_2) = \begin{cases} \left(\frac{\lambda_2+\lambda_3}{\lambda_1+\lambda_2+\lambda_3}\right)^{-1} \lambda_1 e^{-\lambda_1 y_1}(\lambda_2+\lambda_3)\, e^{-(\lambda_2+\lambda_3)y_2}, & y_1 < y_2 \\ \left(\frac{\lambda_2+\lambda_3}{\lambda_1+\lambda_2+\lambda_3}\right)^{-1} \lambda_2 e^{-\lambda_2 y_2}\,(\lambda_1+\lambda_3) e^{-(\lambda_1+\lambda_3)y_1}, & y_1 > y_2 \end{cases}$$

To generate random sample from this distribution:

1. First generate independent samples from $X_1 \sim exp(\lambda_1)$, $X_2 \sim exp(\lambda_2)$ and $X_3 \sim exp(\lambda_3)$.
2. F(x) = 1- $e^{-\lambda x}$, we have used the formula:

$$x = -\frac{log(1-u)}{\lambda}$$

3. Define: $Y_1 = \min(X_1, X_2)$ and $Y_2 = \min(X_1, X_3)$. Accept if $Y_1 \neq Y_2$ otherwise go back to step 1.

R Code:

```
set.seed(123)

BBBE <- function(n, l1, l2, l3){
  uni_01 <- runif(n)
  uni_02 <- runif(n)
  uni_03 <- runif(n)
  x1 <- -log(uni_01)/l1
  x2 <- -log(uni_02)/l2
  x3 <- -log(uni_03)/l3
  y1 <- c()
  y2 <- c()
  i <- 1
  while(length(y1) < n){
    y1[i] <- min(sample(x1, 1), sample(x3, 1))
    y2[i] <- min(sample(x2, 1), sample(x3, 1))
    if(y1[i] == y2[i])
      i <- i - 1
    else i <- i + 1
  }
  y <- rbind(y1, y2)
  return(y)
}
y <- BBBE(1000, 5, 5, 2)

> sum(y[1, ] < y[2,])
[1] 496
> sum(y[1, ] > y[2,])
[1] 504
> sum(y[1, ] == y[2, ])
[1] 0
```

Conclusion: The sample is generated and stored in "y". The number of sample points falling in each class is:

$$I_1 = \{Y_1 < Y_2\} = 496;\ I_2 = \{Y_2 < Y_1\} = 504; I_3 = \{Y_1 = Y_2\} = 0$$

Further expected number of observation in 3 classes are: 500, 500 and 0 respectively.

Question 4.

(i) Generate a random sample of size 1000 from a Weibull (7,3) distribution using a suitable algorithm. Check **validity** of the generated random sample. Now assuming that the generated random sample is a random sample from some Weibull (α, λ) distribution, find MLEs of the parameters using the method of Profile Log Likelihood function before using the Newton-Raphson Algorithm and comment on your results.

Solution: Pdf of Weibull distribution is given by:

$$f_Y(x) = \alpha\lambda x^{\alpha-1}e^{-\lambda x^{\alpha}}; \quad x > 0, \alpha > 0, \lambda > 0$$

$$F_Y^{-1}(u) = 1 - e^{-\lambda x^{\alpha}}$$

From the CDF we can generate random sample from Weibull(7, 3), in order to find MLE, we will use Profile Log Likelihood method, which involves Newton-Raphson method.

1. Log likelihood function is given by l(α , λ) as a function of λ in terms of α:

$$l(\alpha, \lambda) = n \ln \lambda - \lambda \sum_i^n x_i^{\alpha} + c(constant)$$

2. We will differentiate log-likelihood and equate it to zero, then we get:

$$\hat{\lambda} = \frac{n}{\sum_i^n x_i^{\alpha}}$$

3. Substituting the value of λ in l(α , λ) equation and call it h(α). We differentiate h(α) and get:

$$h'(\alpha) = \frac{n}{\alpha} - \frac{n}{\sum_i^n x_i^{\alpha}} \sum_i^n x_i^{\alpha} \ln x_i + \sum_i^n \ln x_i$$

$$h''(\alpha) = -\frac{n}{\alpha^2} - \frac{n}{\sum_i^n x_i^{\alpha}} \sum_i^n x_i^{\alpha} (\ln x_i)^2 + \frac{n}{(\sum_i^n x_i^{\alpha})^2} \left(\sum_i^n x_i^{\alpha} \ln x_i \right)^2$$

4. We use the Newton Raphson Method to maximize h(α)(single parameter) where we iterate:

$$\alpha^{(1)} = \alpha^{(0)} - \frac{h'(\alpha)}{h''(\alpha)}; \quad where\ \alpha = \alpha^{(0)}$$

R code:

```
set.seed(123)

rand_web <- function(n, alpha, lambda){
  webu <- (-log(runif(n))/lambda)^(1/alpha)
}

webu <- rand_web(1000, 7, 3)
```

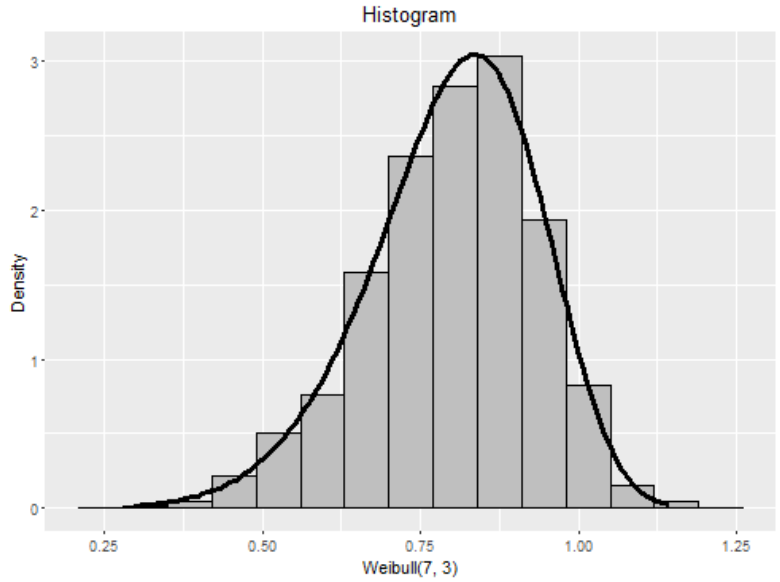
Conclusion: "**rand_web**" function can be called to generate 1000 random numbers from Weibull(7, 3) distribution, the numbers generate here are given in the appendix.

We will check validity of our random sample:

1. Histogram overlaid with a theoretical frequency curve:

R code:

```
library(ggplot2)
p <- ggplot(mapping = aes(x = webu))
p <- p + geom_histogram(aes(y=..density..), binwidth = .07, colour="black",
                        fill="gray75")
p <- p + stat_function(fun=dweibull, args = list(shape=7, scale=1/3^(1/7)),
                       col = 'black', lwd=1.2)
p + xlab("Weibull(7, 3)") + ylab("Density") + ggtitle("Histogram")
```

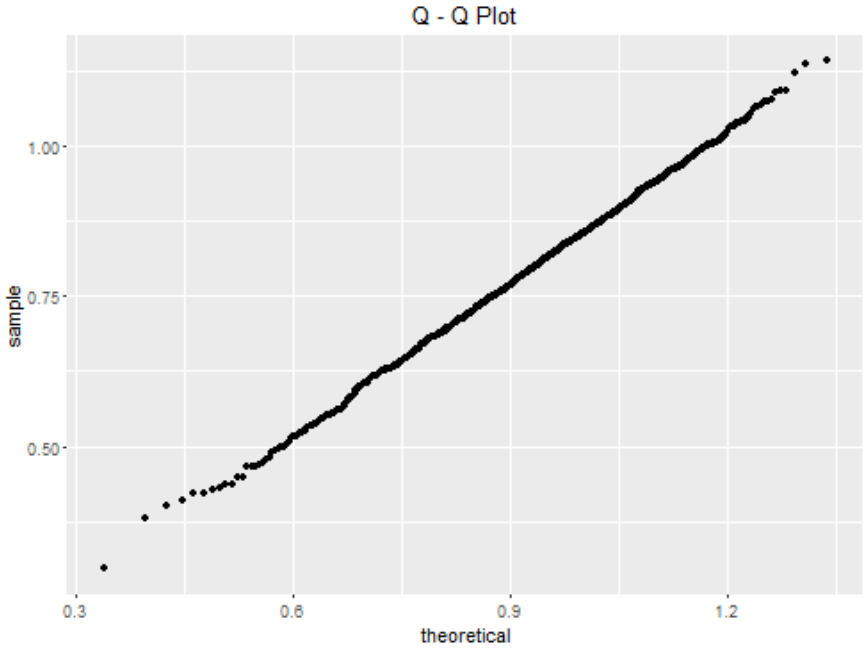


Conclusion: We see that observed sample density is close to expected density.

2. Q-Q Plot:

R code:

```
q <- ggplot(mapping = aes(sample = webu)) + ggtitle("Q - Q Plot")
q + stat_qq(distribution = stats::qweibull, dparams = list(shape=7))
```



Conclusion: We observe that sample and theoretical quantiles lies along a straight line, i.e. they coincides.

3. Chi-Square Test:

H_0: The data are consistent with the Weibull distribution.
H_1: The data are not consistent with the Weibull distribution.

R code:

```
breaks <- seq(min(webu), max(webu), length = 6)
intervals <- cut(webu, breaks, right=FALSE)
observed <- table(intervals)
prob <- c()
for(i in 1:5)
  prob[i] <- pweibull(breaks[i+1], 7, 1/3^(1/7)) - pweibull(breaks[i],
7, 1/3^(1/7))
expected <- prob*1000
chisq.test(cbind(observed, expected))
```

Output:

```
        Pearson's Chi-squared test

data:  cbind(observed, expected)
X-squared = 0.17233, df = 4, p-value = 0.9965
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data fits well to Weibull(7, 3).

4. Kolmogorov-Smirnov Test:

H_0: The samples are coming from the same population.
H_1: The samples are not coming from the same population.

R Code:

```
ks.test(webu, "pweibull", 7, 1/3^(1/7))
```

Output:

```
        One-sample Kolmogorov-Smirnov test

data:  webu
D = 0.014051, p-value = 0.9891
alternative hypothesis: two-sided
```

Conclusion: We observe that p-value is insignificant, hence we conclude that data is coming from Weibull(7, 3).

To find MLE using Profile Log Likelihood:

R Code:

```
al <- 8                          # Initial guess for the parameter
d <- 1
n <- 1000
while(d > 0.00001){
  ao <- al
  h1 <- n/al-(n/sum(webu^al))*sum(webu^al*log(webu)) + sum(log(webu))
  h2 <- -(n/al^2) - n*(sum((webu^al)*log(webu)^2)/sum(webu^al))+
          n*sum((webu^al)*log(webu))^2/(sum(webu^al))^2
  al <- ao - h1/h2
  d <- abs(al - ao)
}

> alpha_hat <- al
> alpha_hat
[1] 7.054301
> lambda_hat <- n/sum(webu^alpha_hat)
> lambda_hat
[1] 2.996589
```

Conclusion: The MLE of the parameters is given by: $\hat{\boldsymbol{\alpha}}$ = 7.0543 and $\hat{\boldsymbol{\lambda}} = 2.99658$.

All the validity test holds true, hence we conclude that our random number generator is a valid generator in generating random sample from Weibull(7, 3).

Question 4

(ii) Generate a random sample of size 1000 from the following mixture of two distributions:

$$f_X(x\,;\boldsymbol{\theta}) = \pi f_0(x\,;\boldsymbol{\theta_0}) + (1-\pi) f_1(x\,;\boldsymbol{\theta_1})$$

where $\pi = \frac{2}{5}$ and, f_0 and f_1 are defined the density functions of $N(3,2)$ and $N(10,3)$ distributions respectively. Clearly $\boldsymbol{\theta_0} = (3,2)^T$, $\boldsymbol{\theta_1} = (10,3)^T$ and $\boldsymbol{\theta} = \left[\frac{2}{5}, (3,2)^T (10,3)^T\right]^T$. Now assuming that the generated random sample is a random sample from some mixture of two normal distributions with unknown parameters, find the MLEs of the parameters viz., $\pi, \boldsymbol{\theta_0} = (\mu_0, \sigma_0^2)$ and $\boldsymbol{\theta_1} = (\mu_1, \sigma_1^2)$ using the Expectation-Maximization (EM) Algorithm.

Solution:

To generate the sample from the mixture of 2 distributions, we first generate the required number of samples from the two univariate normal distributions independently. Then we generate the mixture by taking proportion of data coming from $f_0(x\,;\theta_0)$ to be equal to π and that from $f_1(x\,;\theta_1)$ to be equal to (1 – π).

R Code:

```
set.seed(123)

rand_snormal <- function(n){
  uni_01 <- runif(2*n)
  y1 <- uni_01[1:n]
  y2 <- uni_01[n+1:1000]
  y <- c(y1, y2)
  x <- c()
  for(i in 1:n){        # Random number from double exponential(1)
    x[i] <- -log(1-y1[i])
    x[n +i] <- log(1-y2[i])
  }

  p <- c()
  i<-1
  while(length(p) < 1000){       # Acceptance/Rejection condition
    U = sample(uni_01, 1)
    z = sample(x, 1)
    if(exp(-abs(z))*1.315*U/2 <= dnorm(z, 0, 1)){
      p[i] <- z
      i <- i+1
    }
    else i <- i-1
  }
  return(p)
}

> rn1 <- rand_snormal(1000)*sqrt(2) + 3
> c(mean(rn1), var(rn1))
[1] 3.009182 1.929094
> rn2 <- rand_snormal(1000)*sqrt(3) + 10
> c(mean(rn2), var(rn2))
[1] 10.083094  2.975097
```

Conclusion: We have generated 2 samples from $N(3,2)$ and $N(10,3)$ distribution in "rn1" and "rn2" respectively.

Then from the mixture we try to find MLE of parameters by using Expectation-Maximization (EM) Algorithm:

1. A initial guess is made about the parameters:

$$\theta^t = (\pi^t, {\mu_0}^t, {\mu_1}^t, {\Sigma_0}^t, {\Sigma_1}^t)^T$$

2. Next we calculate $\delta_i{}^t = \dfrac{f_1(x_i;{\theta_1}^t)\,x\,(1-\pi^t)}{f_1(x_i;{\theta_1}^t)\,x\,(1-\pi^t)+ f_0(x_i;{\theta_0}^t)\,x\,(\pi^t)}$
3. The next iterate of the parameter vector is obtained as:

$$\hat{\pi}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}(1-{\delta_i}^t)$$

$$\widehat{\mu_0}^{(t+1)} = \frac{1}{\sum_{i=1}^{n}(1-{\delta_i}^t)}\sum_{i=1}^{n}(1-{\delta_i}^t)\,x_i$$

$$\widehat{\mu_1}^{(t+1)} = \frac{1}{\sum_{i=1}^{n} \delta_i{}^t)} \sum_{i=1}^{n} \delta_i{}^t \; x_i$$

$$\widehat{\Sigma_0}^{(t+1)} = \frac{1}{\sum_{i=1}^{n} (1 - \delta_i{}^t)} \sum_{i=1}^{n} (1 - \delta_i{}^t)(x_i - \widehat{\mu_0}^{(t+1)})(x_i - \widehat{\mu_0}^{(t+1)})^T$$

$$\widehat{\Sigma_1}^{(t+1)} = \frac{1}{\sum_{i=1}^{n} \delta_i{}^t} \sum_{i=1}^{n} \delta_i{}^t \; (x_i - \widehat{\mu_1}^{(t+1)})(x_i - \widehat{\mu_1}^{(t+1)})^T$$

After this we will get new parameters:

$$\hat{\theta}^{(t+1)} = (\hat{\pi}^{(t+1)}, \widehat{\mu_0}^{(t+1)}, \widehat{\mu_1}^{(t+1)}, \widehat{\Sigma_0}^{(t+1)}, \widehat{\Sigma_1}^{(t+1)})$$

4. We will repeat this procedure till convergence criteria is satisfied i.e.

$$| \hat{\theta}^{(t+1)} - \hat{\theta}^{t}) < \varepsilon |$$, for some pre-specified threshold ε.

R Code:

```
x <- c(sample(rn1, 400), sample(rn2, 600))
p <- 0.4
m1 <- 6
m2 <- 18
v1 <- 4
v2 <- 8
d <- rep(1, 5)
a <- 0.00001
i <- 0
while(sum(d > a) > 0){
  d1 <- c(p, m1, m2, v1, v2)
  delta <- (dnorm(x, m2, sqrt(v2))*(1-p))/(dnorm(x, m1, sqrt(v2))*p +
                                           dnorm(x, m2, sqrt(v2))*(1-p))
  p <- sum(1 - delta)/1000
  m1 <- sum((1-delta)*x)/sum(1-delta)
  m2 <- sum((delta)*x)/sum(delta)
  v1 <- sum((1-delta)*(x-m1)^2)/sum(1-delta)
  v2 <- sum((delta)*(x-m2)^2)/sum(delta)
  d2 <- c(p, m1, m2, v1, v2)
  d <- abs(d2 - d1)
  d1 <- d2
  i <- i+1
}
> # Number of iterations performed:
> i
[1] 13
> # Parameter estimates are:
> d1
[1]  0.4109478  3.0905900 10.1426955  2.3751448  2.8903687
```

Conclusion: MLE'S of the parameters obtained by EM algorithm are very close to the actual parameters, hence EM algorithm is a valid technique to obtain MLE of parameters from a mixture of 2 distributions.