STAT 428 Statistical Computing

Homework 1 Solutions

Department of Statistics, University of Illinois at Urbana-Champaign

January 22, 2017

Ex.1

(a)

Since we have the pdf of X: $\lambda X^{\lambda-1}$ for $\lambda > 0$ and $0 < x < 1$, we can derive the cdf of x:

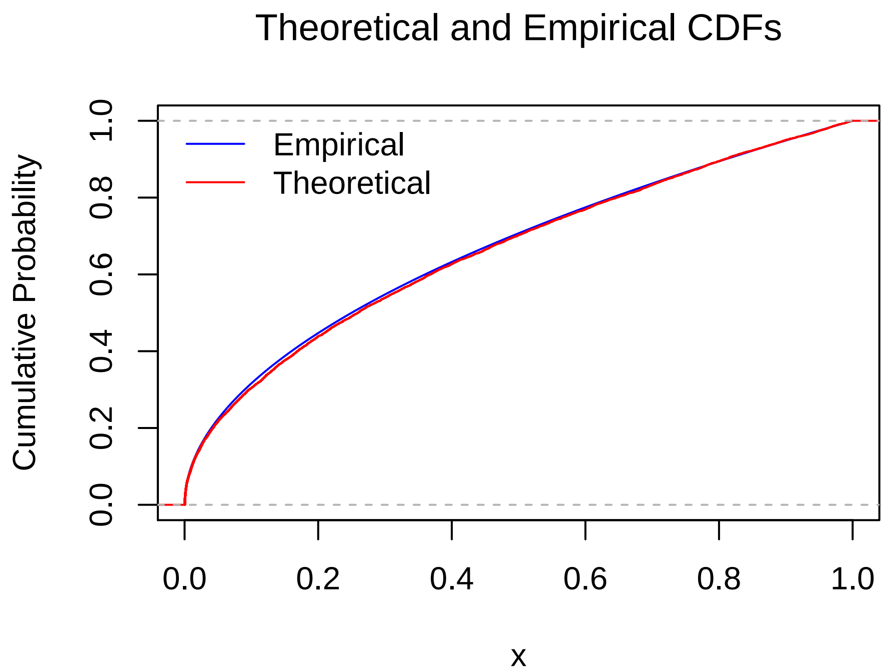
$$F(x) = \int_0^x f(t)dt = x^\lambda (0 < x < 1)$$

Consider a random variable U that is uniformly distributed on (0,1) and define the random variable $X = F^{-1}(U) = U^{1/\lambda}$, X has cdf F.

```
randf <- function(n, lambda){
    Usample = runif(n,0,1)
    Xsample = Usample^(1/lambda)
    return(Xsample)
}
```

(b)

```
#Draw a large sample
n = 10000
lambda = 0.5
Xsample = randf(n, lambda)

#Plot the theoretical CDF
x = seq(0,1,l=10000)
y = x^lambda
plot(x, y, type = 'l', col = 'blue', xlim = c(0,1),ylab='Cumulative Probability')
title(main = "Theoretical and Empirical CDFs", font.main = 1)

#Plot the empirical CDF
empF = ecdf(Xsample)
lines(empF, col = 'red')
legend("topleft", col = c('blue','red'), lty = 1, c("Empirical", "Theoretical"), bty = "n")
```



Ex.2

A significant feature of R language is the frequent use of vectorization. Try to vectorize your data can not only simplify your code but also make your program faster. The family of apply() functions is exactly designed for this purpose

(a)

Solution 1 (Mapply version)

```
randz1 <- function(m, n, k, mu, sigmaSq){
    #Generate N(mu,sigma) of sample size n for m times
  #Each column of norm.mat represents a sample of size n from a N(mu,sigma) distribution
    norm.mat = mapply(function(mean_,sd_){rnorm(n,mean_,sd_)}, rep(mu,m), rep(sqrt(sigmaSq),m))
    #Sort each column of the matrix in an ascending order
    norm.mat.sort = apply(norm.mat, MARGIN = 2, FUN = sort)
    #Take the k-th order statistic
    return(norm.mat.sort[k,])
}
```

Solution 2 (Replicate version)

```
randz2 <- function(m, n, k, mu, sigmaSq){
    norm.mat = replicate(m,rnorm(n,mu,sqrt(sigmaSq)))
```

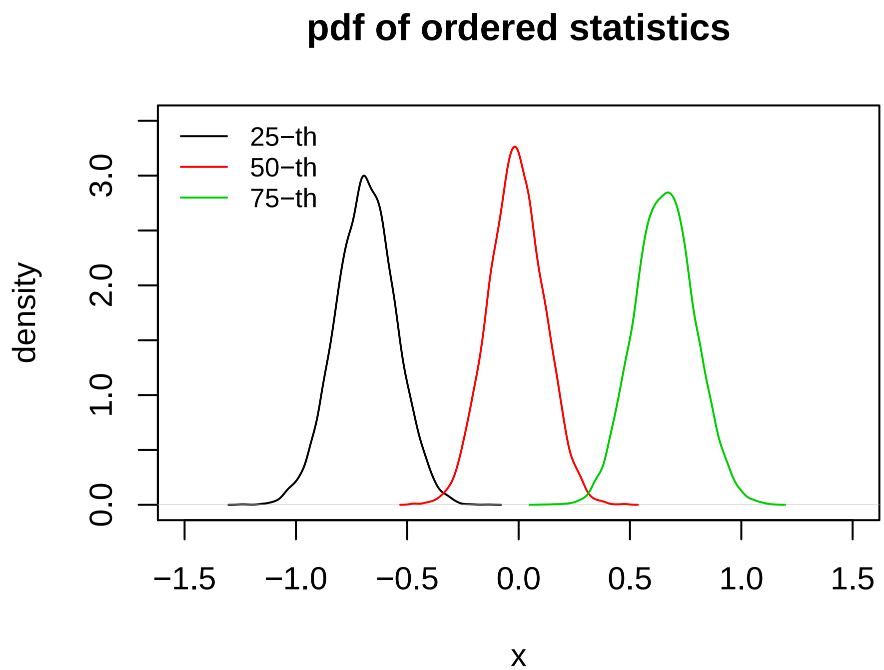
```
    norm.mat.sort = apply(norm.mat, MARGIN = 2, FUN = sort)
    return(norm.mat.sort[k,])
}
```

Solution 3 (For-loop version)

```
randz3 <- function(m, n, k, mu, sigmaSq){
  #create an empty array to store the simulated values
  kth_vector = numeric(m)
  for(i in 1:m){
    normSample = rnorm(n,mu,sqrt(sigmaSq))
    normSample.sort = sort(normSample)
    #Extract the kth smallest value of the sample
    kth_vector[i] = normSample.sort[k]
  }
    return(kth_vector)
}
```

(b)

```
#Set the default values
n = 100
m = 10000
mu = 0
sigmaSq = 1
#Draw large samples of the 25-th, 50-th and 75-th order statistic
k = c(25,50,75)
Zsample1 = randz1(m, n, k[1], mu, sigmaSq)
Zsample2 = randz1(m, n, k[2], mu, sigmaSq)
Zsample3 = randz1(m, n, k[3], mu, sigmaSq)

#Construct estimates of the PDFs and plot the PDFs
plot(density(Zsample1), type = "l", col = 1, xlim = c(-1.5,1.5), ylim = c(0,3.5),
     xlab = "x", ylab = "density",main="pdf of ordered statistics")
lines(density(Zsample2), col = 2)
lines(density(Zsample3), col = 3)
legend("topleft", col = 1:3, lty = 1, c("25-th", "50-th", "75-th"), bty = "n", cex = 0.8)
```



Ex.3

(a)

Solution 1(Geometric Distributon Version)

X follows geometric distribution and can be considered as the number of Bernoulli trials required to observe the 1st success.

```
rand.sample.geom<-function(n,p){
  x<-rep(0,n)
  for(i in 1:n){
  trial<-sample(c(0,1),100,replace=T,prob=c(1-p,p))
  x[i]<-which(trial==1)[1]
  }
  x
}
```

Solution 2 (Inverse CDF Version)

```
randx <- function(n,p){
  #Take care of the upper bound of x
  #The better way here is to check if the cumulative probabiblity equals to 1
    x = 1:1000
    prob = p*((1-p)^(x-1))
    Fx = cumsum(prob)
```

```
    Usample = runif(n,0,1)
    Xsample = rep(1,n)
    for(i in 1:1000){
        Xsample = Xsample + (Usample>Fx[i])
    }
    return(Xsample)
}
```

(b)

X is a special case of Y when $k = 1$.

We can use the following R function to draw samples from the distribution from Y.

Solution 1(Geometric Distributon Version)

Y denotes the number of Bernoulli trials required to observe the k th success.

```
rand.sample.bn<-function(n,p,k){
  x<-rep(0,n)
  for(i in 1:n){
    trial<-sample(c(0,1),100,replace=T,prob=c(1-p,p))
    x[i]<-which(cumsum(trial)==k)[1]
  }
  x
}
```

Solution 2 (Inverse CDF Version)

```
randy <- function(n,p,k){
    x = k:(1000+k)
    prob = choose(x-1, k-1)*(p^k)*((1-p)^(x-k))
    Fx = cumsum(prob)
    Usample = runif(n,0,1)
    Xsample = rep(k,n)
    for(i in 1:1001){
        Xsample = Xsample + (Usample>Fx[i])
    }
    return(Xsample)
}
```

Ex.4

(a)

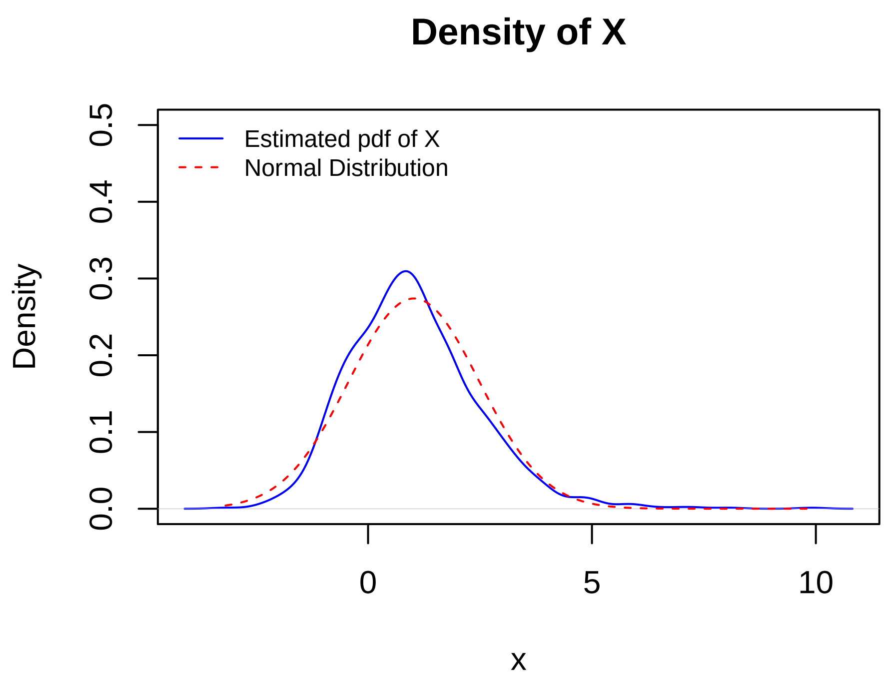
Y follows $exp(1)$ distribution.

```
rand.sample<-function(n){
  y<-rexp(n,1)
  x<-rnorm(n,y,1)
```

```
  x
}
```

(b)

```
x.sample<-rand.sample(1000)
mu<-mean(x.sample)
sigmaSq<-sd(x.sample)
x<-seq(range(x.sample)[1],range(x.sample)[2],0.01)
plot(density(x.sample),col="blue",ylim=c(0,0.5),main="Density of X",xlab='x')
lines(x,dnorm(x,mu,sigmaSq),lty=2,col="red")
legend("topleft",lty=c(1,2), col=c("blue","red"),
       legend=c("Estimated pdf of X","Normal Distribution"),cex=0.75,bty="n")
```



The estimated pdf of X is nearly the same as the density of normal distribution of which the mean is the sample mean of X and the variance is the sample variance of X. But it is also noticeable that the estimated density of X is right skewed.

Ex.5 (Problem 3.3)

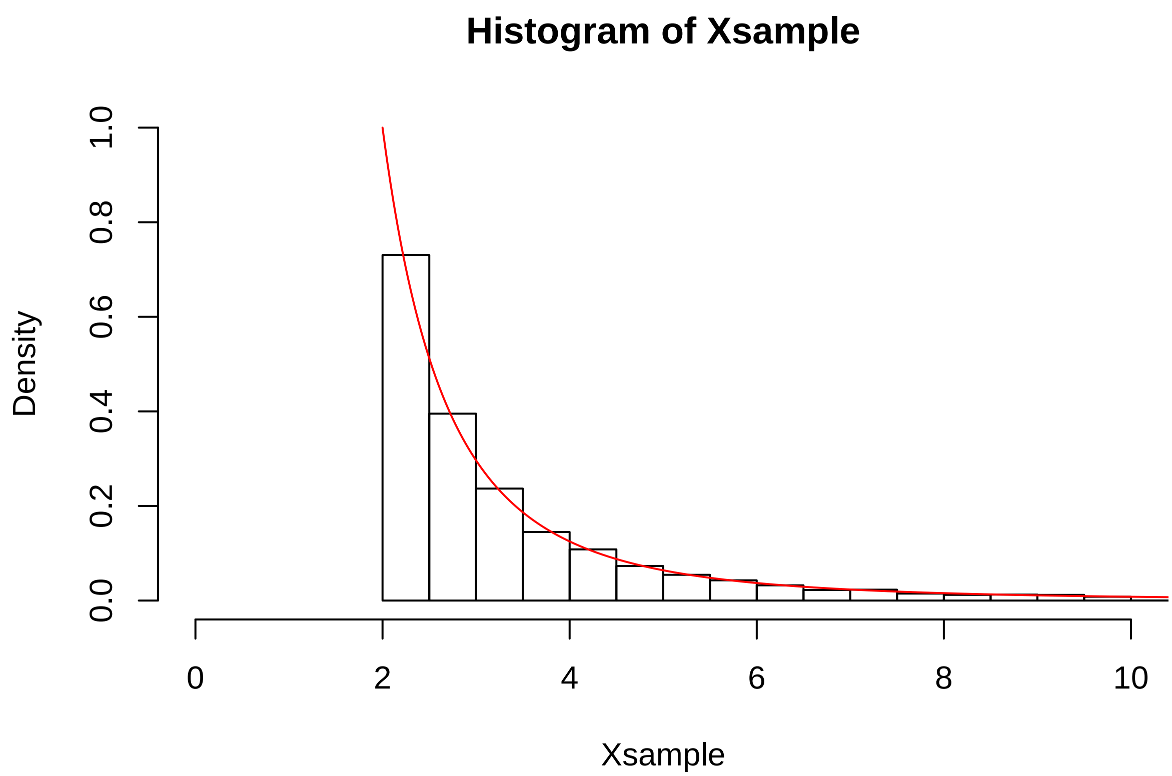
Let $X = F^{-1}(U)$ where U~uniform(0,1)

$$X = \frac{b}{(1-u)^{\frac{1}{a}}} \qquad 0 \le u \le 1, x \ge b$$

Since $a = b = 2$, we have

$$X = \frac{2}{(1-u)^{\frac{1}{2}}} \qquad 0 \le u \le 1, x \ge 2$$

```
Usample = runif(10000,0,1)
Xsample = 2/sqrt(1-Usample)
x = seq(2,100,l=10000)
y = 8/(x^3)
hist(Xsample,breaks=c(seq(2,10,.5),max(Xsample)),freq=F,xlim=c(0,10),ylim=c(0,1))
lines(x,y,col="red")
```



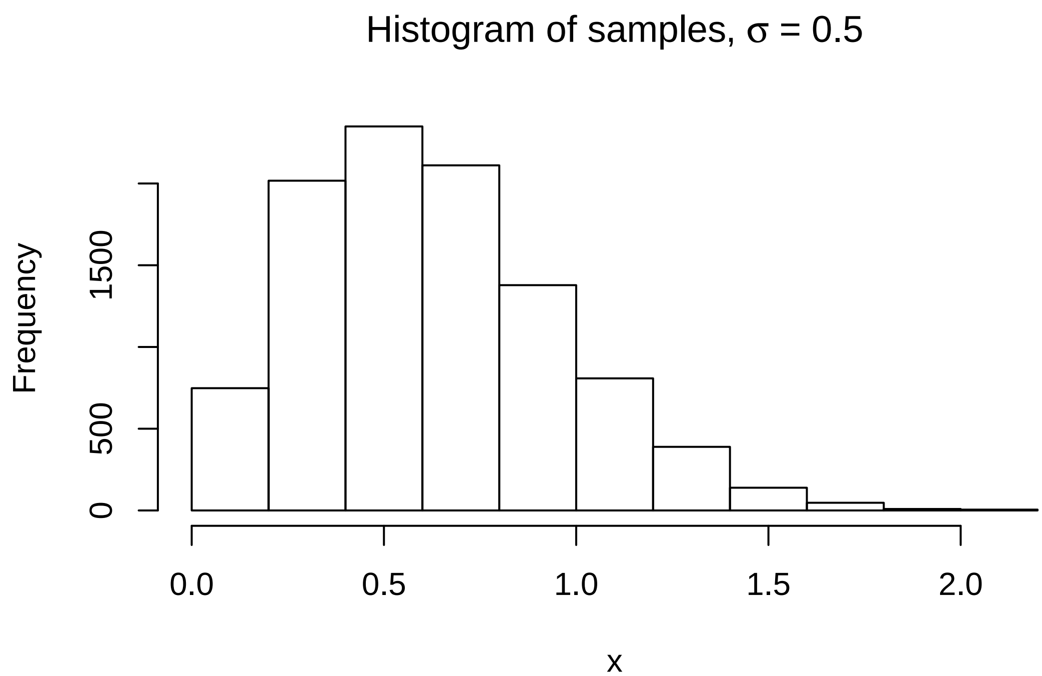
Ex.6 (Problem 3.4)

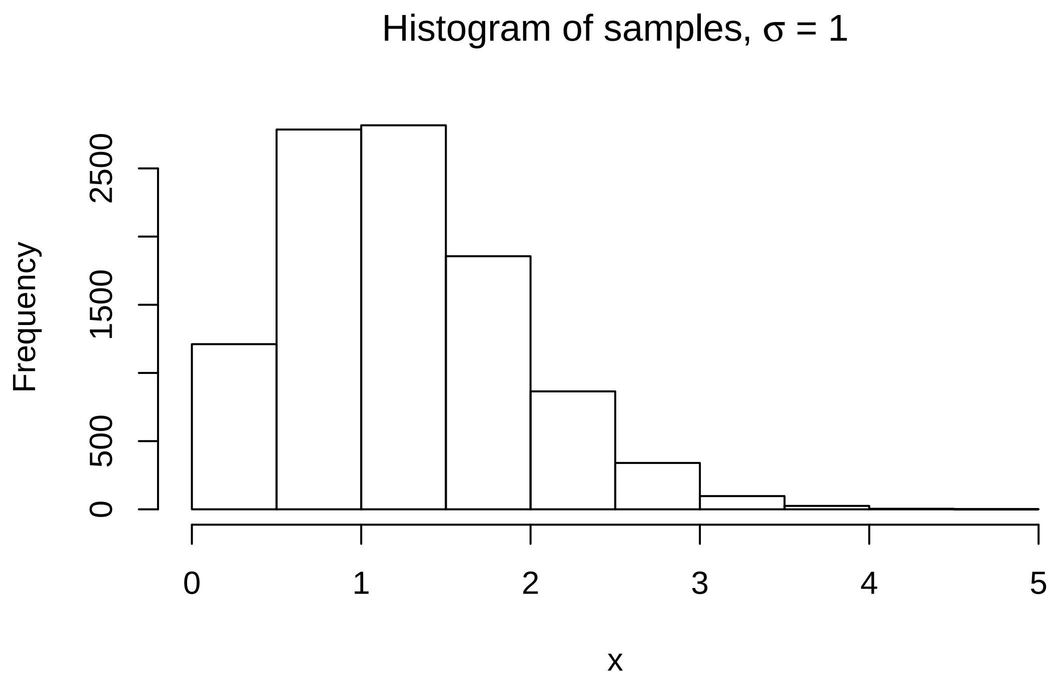
Since we have the pdf of X: $\lambda X^{\lambda-1}$ for $\lambda > 0$ and $0 < x < 1$, we can derive the cdf of x:

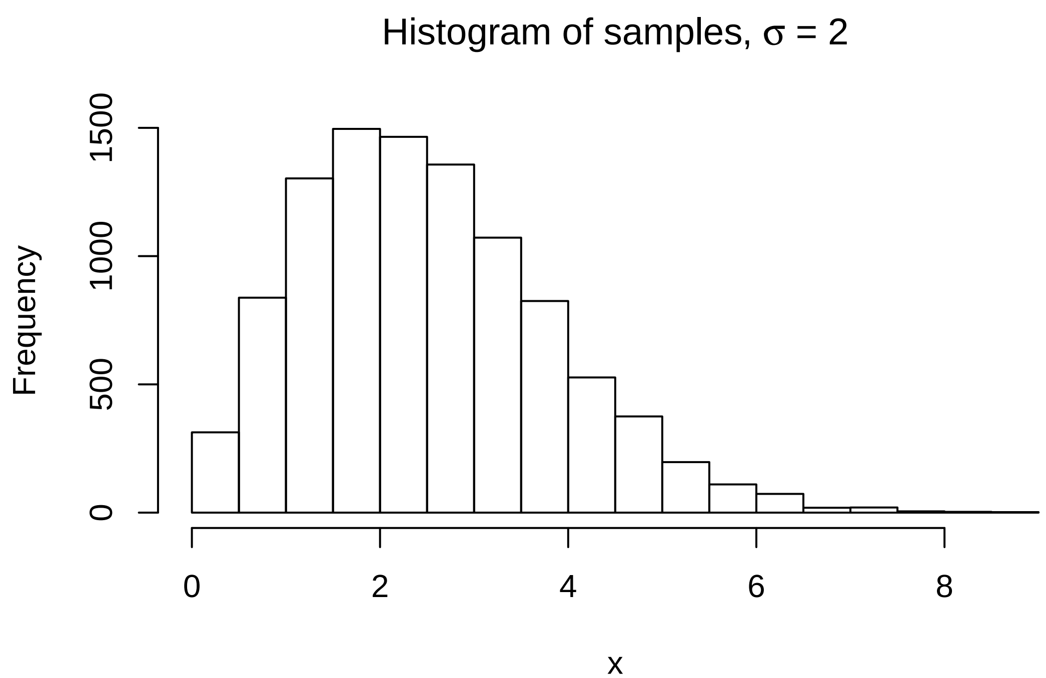
$$\begin{aligned} F(x) = \int_0^x f(t)dt &= \int_0^x \frac{t}{\sigma^2} e^{-\frac{t^2}{2\sigma^2}} dt \\ &= -e^{-\frac{t^2}{2\sigma^2}} \mid_0^x \\ &= 1 - e^{-\frac{x^2}{2\sigma^2}} \end{aligned}$$

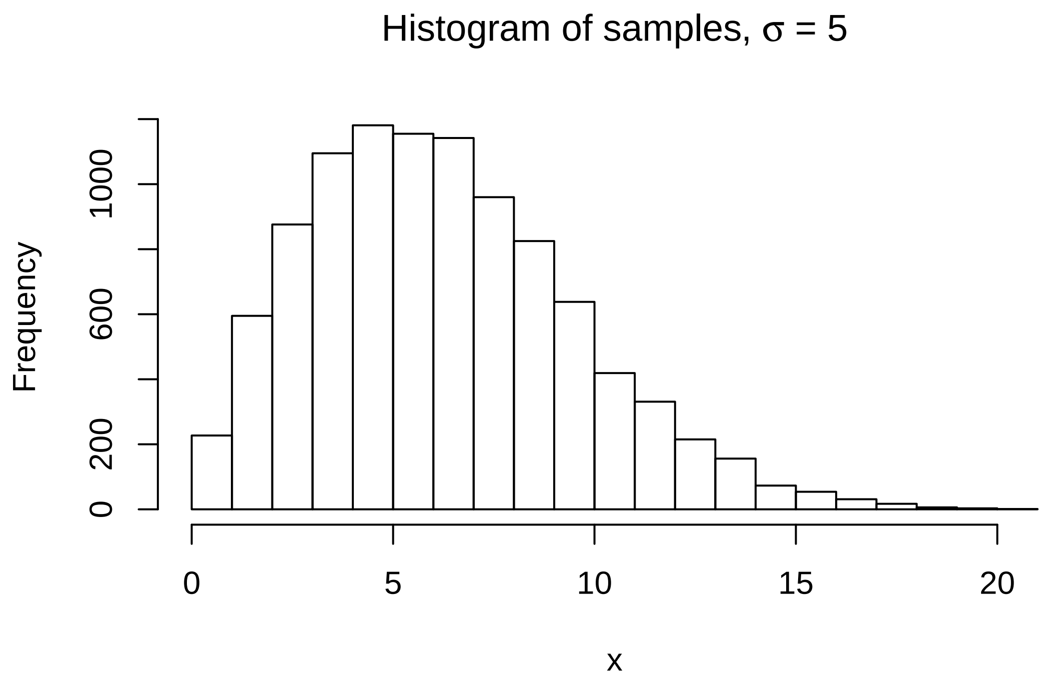
Consider a random variable U that is uniformly distributed on (0,1) and define the random variable $X = F^{-1}(U) = \sqrt{-2\sigma^2 \ln(1-U)}$, X has cdf F.

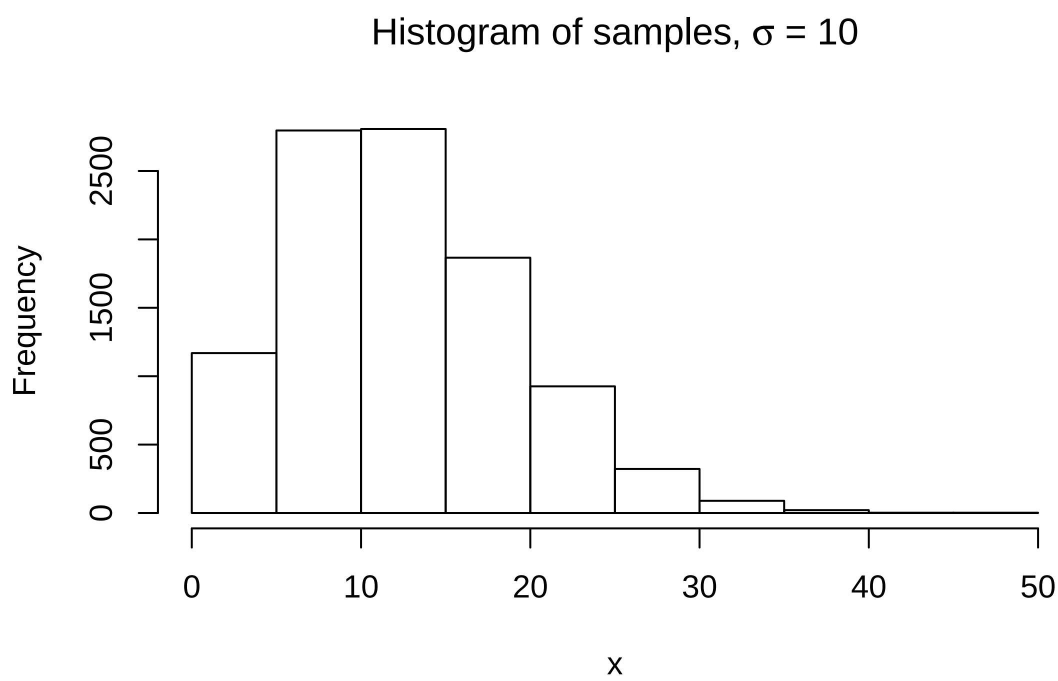
```r
sigma.list = c(.5,1,2,5,10)
randf <- function(sig,n){
    Usample = runif(n,0,1)
    Xsample = sqrt(-2*sig^2*log(1-Usample))
    return(Xsample)
}

for(i in 1:length(sigma.list)){
  hist(randf(sigma.list[i],10000),xlab="x",
       main=substitute(paste("Histogram of samples, ",sigma," = ",s),list(s=sigma.list[i])))
}
```




Histogram of samples, σ = 1
Frequency
0
500
1500
2500
0
1
2
3
4
5
x


Histogram of samples, σ = 2
Frequency
0
500
1000
1500
0
2
4
6
8
x


Histogram of samples, σ = 5
Frequency
0
200
600
1000
0
5
10
15
20
x


Histogram of samples, σ = 10
Frequency
0
500
1500
2500
0
10
20
30
40
50
x

Ex.7 (Problem 3.5)

```r
probs = c(.1,.2,.2,.2,.3)
Fx = cumsum(probs)
Usample = runif(1000,0,1)
Xsample = rep(0,1000)
for(i in 1:5){
  Xsample = Xsample+(Usample>Fx[i])
}

theoXsample = sample(0:4,1000,replace=T,probs)

#table is pretty convenient here since the factors are exacty the values of the random variable
rbind(Empirical = table(Xsample)/1000,Theoretical = probs)
```

```
##                 0     1     2     3    4
## Empirical   0.101 0.208 0.203 0.198 0.29
## Theoretical 0.100 0.200 0.200 0.200 0.30
```

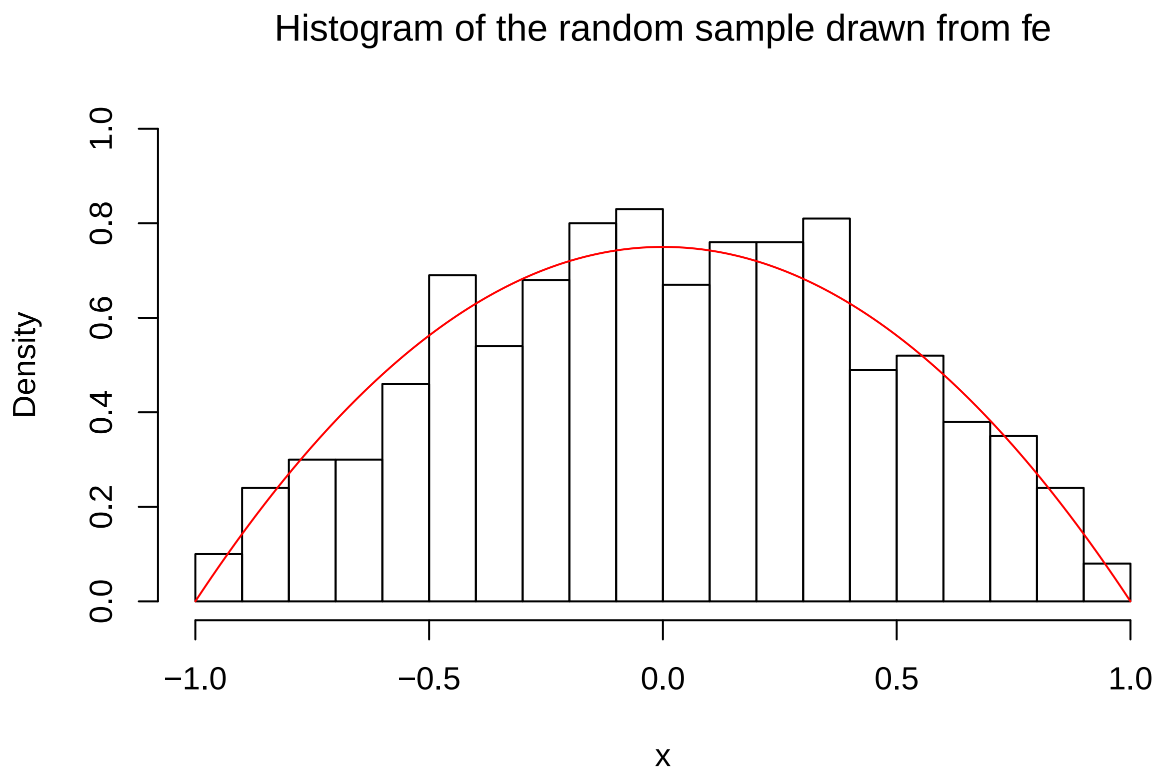
```r
rbind(Empirical = table(theoXsample)/1000,Theoretical = probs)
```

```
##                 0     1     2     3     4
## Empirical   0.097 0.187 0.198 0.211 0.307
## Theoretical 0.100 0.200 0.200 0.200 0.300
```

The empirical probabilities is very close to the theoretical probabilies

Ex.8 (Problem 3.9)

```r
randf <- function(n){
  Usample1 = runif(n,-1,1)
  Usample2 = runif(n,-1,1)
  Usample3 = runif(n,-1,1)
  #if |U3| > |U1| and |U3| > |U2|
  index.U3 = (abs(Usample3)>abs(Usample1)) & (abs(Usample3)>abs(Usample2))
  #select
  Xsample = c(Usample2[index.U3],Usample3[!index.U3])
  return(Xsample)
}

hist(randf(1000),breaks = 20,freq=F, ylim=c(0,1),
     main=expression("Histogram of the random sample drawn from fe"),xlab = "x")
#Superimposed density plot
x=seq(-1,1,l=1000)
lines(x,3/4*(1-x^2),col="red")
```



Bonus

In order to apply acceptance-rejection method, we set $g(x, y) = 2$, where $0 < x < 1$, $0 < y < 1$, $x + y < 1$. We can derive a upper bound for the mode of f(x,y) by

$$f(x, y) = 60x^2y < 60x^2(1-x) \leq \frac{80}{9}$$

the equality holds when $x = \frac{2}{3}$. It is easy to show that $\frac{f(x,y)}{g(x,y)} < \frac{40}{9}$ when $f(x, y) > 0$.

The code to generate a sample of size n is as follow.

```
randmultivar <- function(n){
    #Acceptance-Rejection Sampling
    c = 40/9
    cnt = 0 #count the number of accepted sample points
    x.sample = matrix(nrow = n, ncol = 2, data = 0)
    while(cnt < n){
      X = runif(1,0,1)
      Y = runif(1,0,1)
      if(X+Y<1){
        U = runif(1,0,1)
        ratio = 30*(X^2)*Y/c
        #Accept when U<ratio
```

```
        if(U<ratio){
          cnt = cnt + 1
          x.sample[cnt,] = c(X,Y)
        }
      }
    }
    return(x.sample)
}
```

Though we can extend acceptance-rejection method to handle the multivariate cases, it would be formidable to find a 'samplable' random variable with a close pdf to the one we are interested in. Using uniform distribution in those cases might lead to a very large c, therefore the acceptance-rejection method becomes less efficient.