Final

2019150432 임효진

June 22, 2021

1

(a)

```
ISg1=function() {
  u=runif(1)
  return(u^2)
}
ISg2=function() {
  u=runif(1)
  return(2*u-u^2)
}

compY=function(){
  i=rbinom(1, 1, 0.5)
  if(i==0){y=ISg1()}
  else{y=ISg2()}
  return(y)
}

simY=replicate(10^4, compY())
```

First we generate an integer $k \in \{0, 1\}$ where $P(0) = P(1) = 0.5$. If k=1 we generate y from $g_1(y)$ and vice versa. To generate from each pdf, we will calculate the cdf for reach pdf. and the inverse of the cdf which is

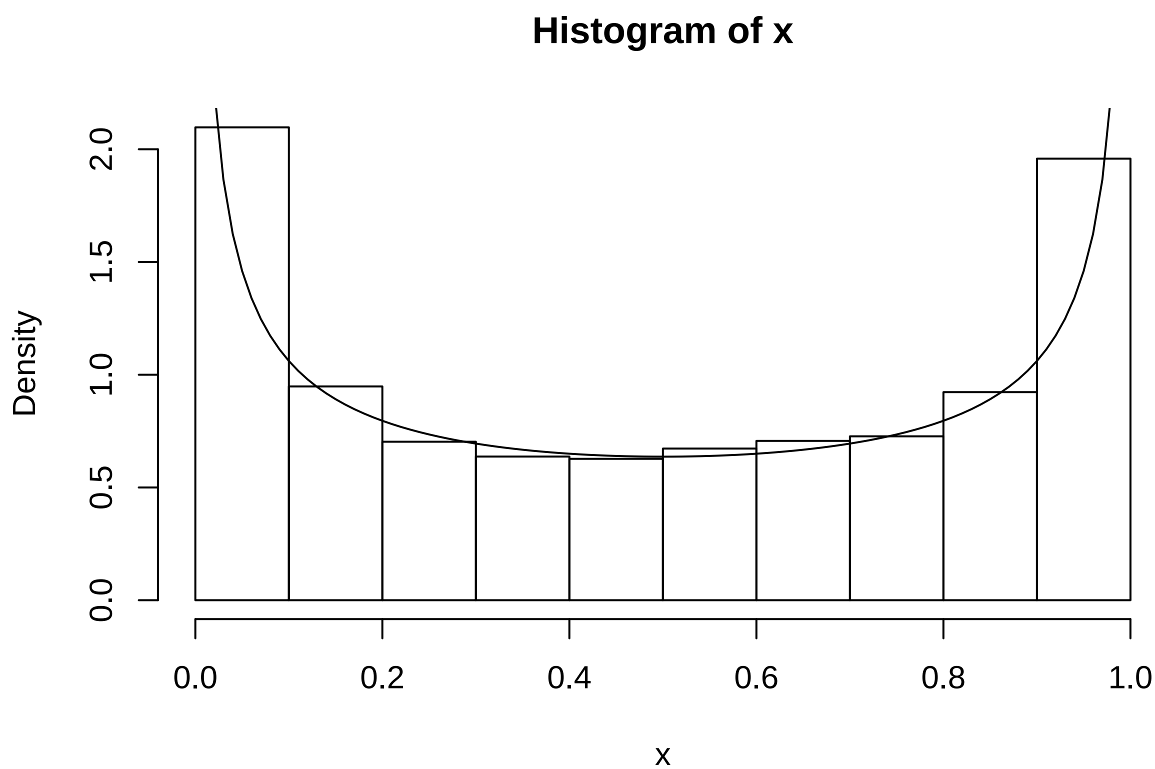
$$G_1(y)^{-1} = y^2$$

$$G_2(y)^{-1} = 2y - y^2$$

(b)

```
n=10^4
f=function(x) 1/(pi*sqrt(x)*sqrt(1-x))
g=function(x) (1/(2*sqrt(x))+1/(2*sqrt(1-x)))/2
M=optimize(f=function(x){f(x)/g(x)},
```

```
            maximum=T, interval=c(0,1))$objective
u=runif(n)
y=replicate(10^4, compY())
x=y[u<f(y)/(M*g(y))]

hist(x, freq = F)
curve(f(x), add=T)
```



To simulate random variable X with rejection sampling using g(y), first we should generate random variable Y from pdf g(y). Here we used the compY() from (a). Then we should generate u which follows an uniform distribution. If $U < \frac{1}{M} f_X(Y)/g(Y)$, set X=Y. Else, discard Y and U, and start the process again.

(c)

Suppose that X follows an half-cauchy distribution with sigma=1, and Y|X follows a standard normal distribution. Than using the method of composition we can easily see that Y follows a beta distribution with $\alpha = \beta = 0.5$. When simulating Y|X we can use the box-muller transformation.

2

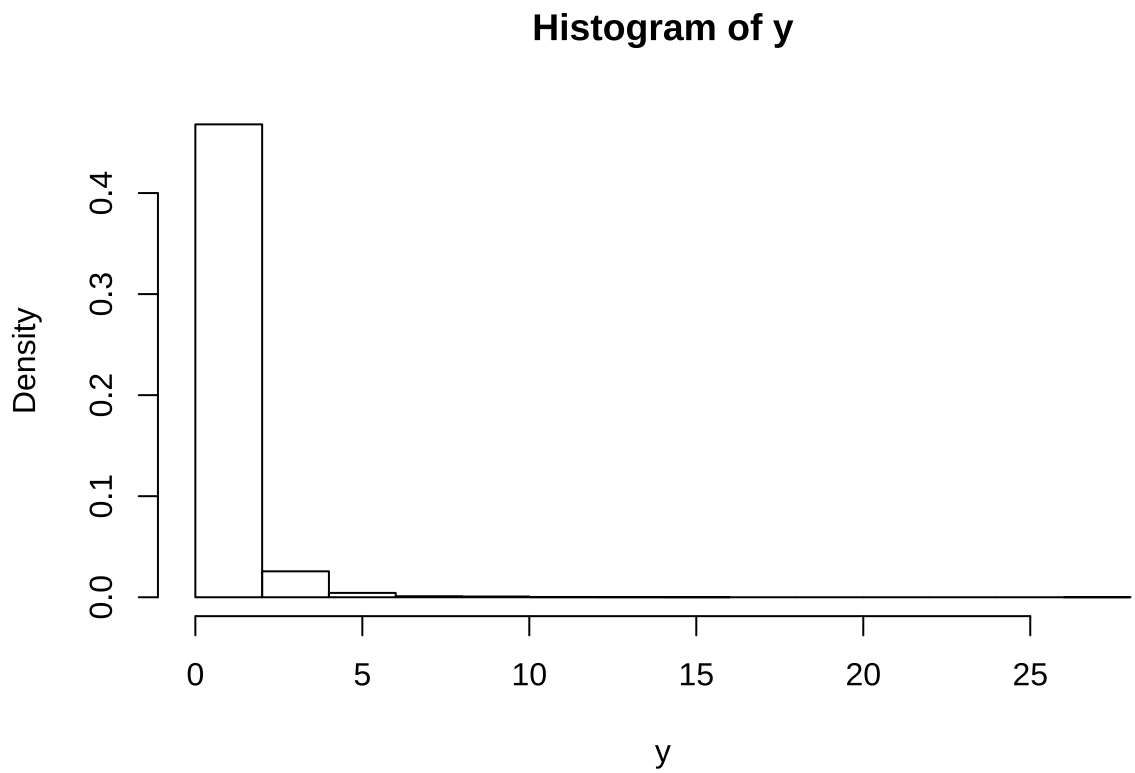
```r
simT=function(r){
  x=rnorm(1, 0, 1)
  y=rchisq(1, r)
  t=x/sqrt(y/r)
  return(t)
}

x=replicate(10^4, simT(4))
y=replicate(10^4, simT(5))
mean(abs(x-y))
```

```
## [1] 1.424602
```

3

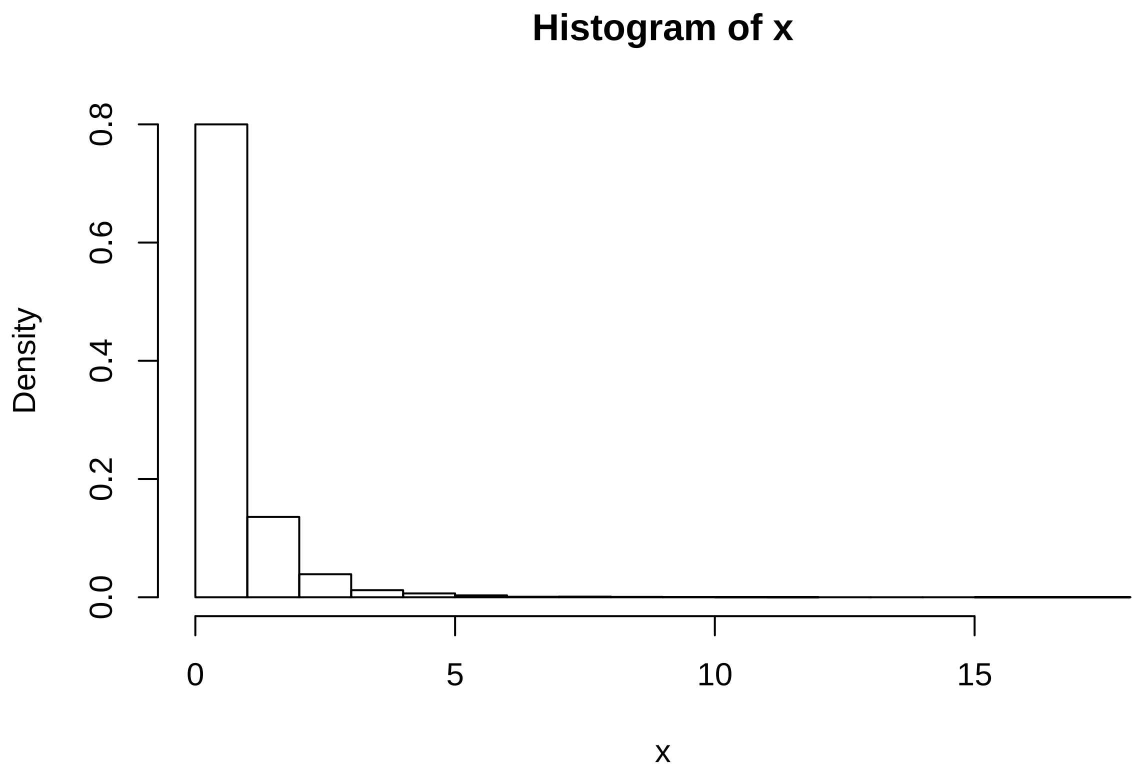
(a)

```r
n=10^4
u=runif(n)
y=(-16/(u-1))^(1/4)-2
hist(y, freq = F)
```



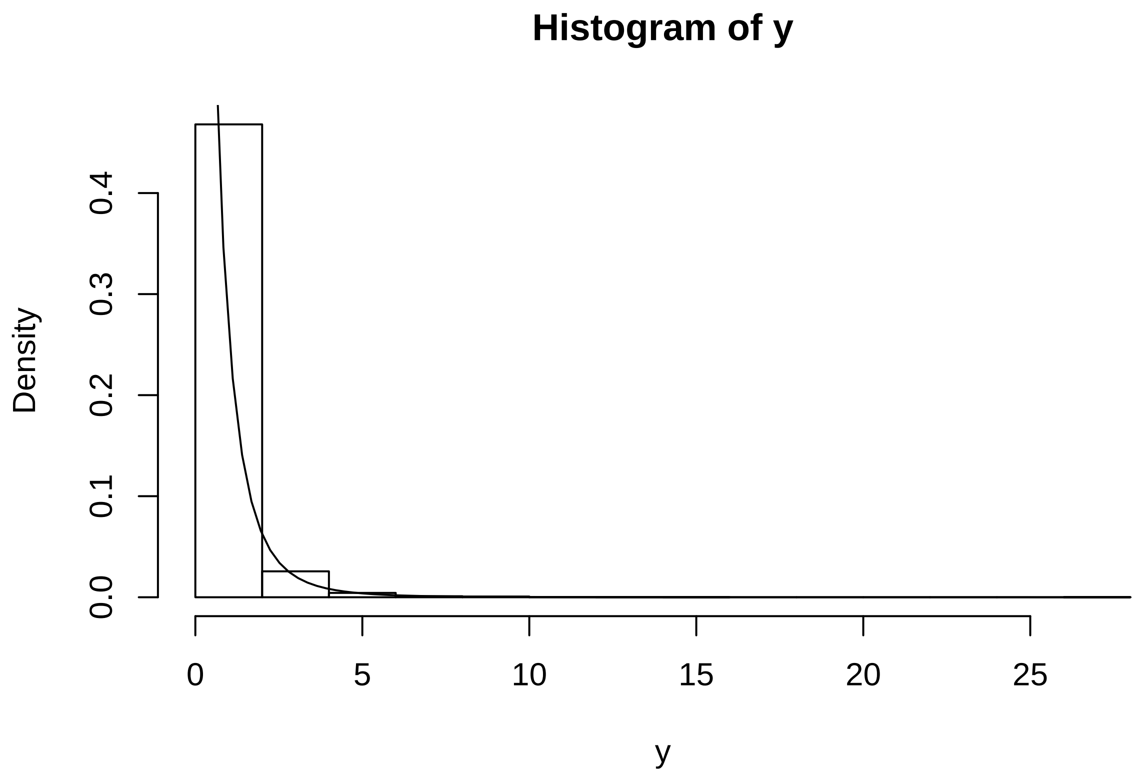
(b)

```
n=10^4
w=rgamma(n, 4, 2)
x=rexp(n, w)
hist(x, freq=F)
```

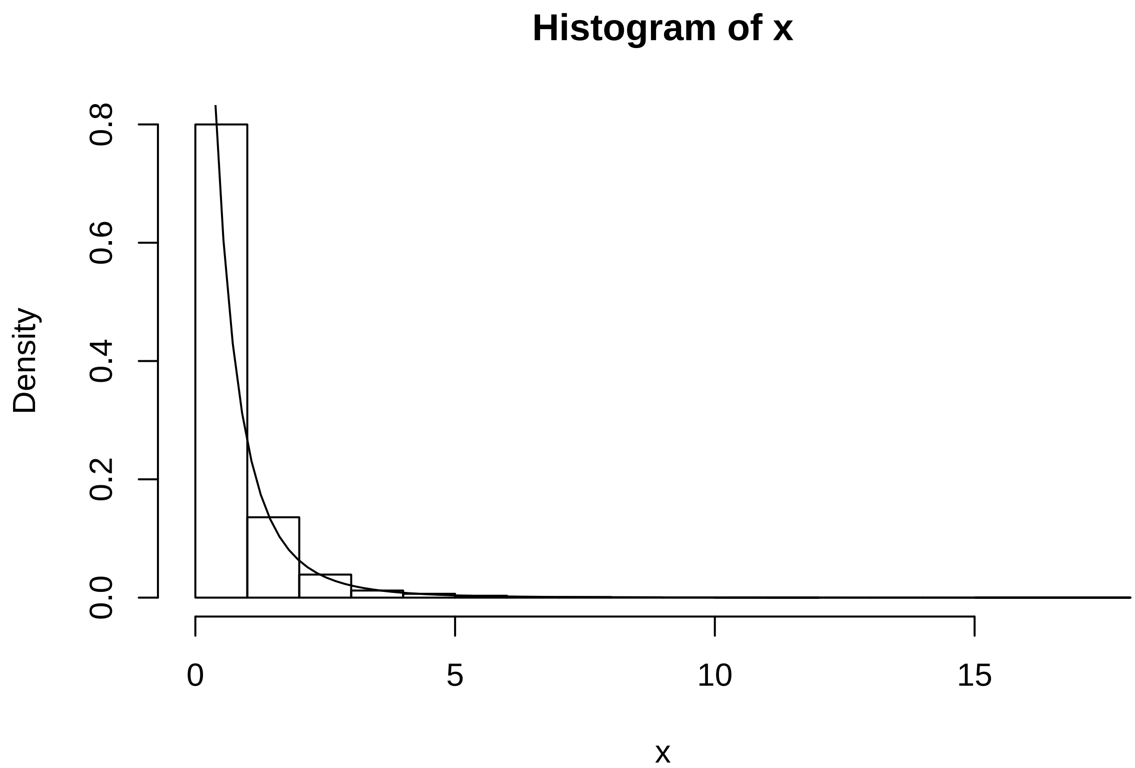


(c)

```
hist(y, freq = F)
curve(64/(2+x)^5, add=T)
```




```r
hist(x, freq=F)
curve(64/(2+x)^5, add=T)
```




```r
mean(y)
```

```
## [1] 0.6777903
```

```r
var(y)
```

```
## [1] 0.9236392
```

```r
mean(x)
```

```
## [1] 0.6651359
```

```r
var(x)
```

```
## [1] 0.8426813
```

From the above results we can assume that X and Y have the same distribution.

4

(a)

```r
simF=function(y){
    v1=rchisq(1, 3)
    v2=rchisq(1, 4)
```

```r
    f=(v1/3)/(v2/4)
    return(f)
}

randF=replicate(10^4, simF())

mean(randF>40)
```

```
## [1] 0.0018
```

```r
pf(40, 3, 4, lower.tail = F)
```

```
## [1] 0.001929985
```

(b)

```r
n=10^4

isF= function(z,n){
  u=runif(n)
  gx=function(x) {200/x^3}
  X=sqrt(100/(1-u))
  w = df(X, 3, 4)/gx(X)
  mean(w*(X>z))
}

isF(40, n)
```

```
## [1] 0.002011414
```

(c)

```r
system.time({
  simF=function(y){
      v1=rchisq(1, 3)
      v2=rchisq(1, 4)
      f=(v1/3)/(v2/4)
      return(f)
  }

  randF=replicate(10^4, simF())

  mean(randF>40)
})
```

```
##    user  system elapsed
##    0.11    0.02    0.14
```

```r
system.time({
  n=10^4

isF= function(z,n){
  u=runif(n)
  gx=function(x) {200/x^3}
  X=sqrt(100/(1-u))
  w = df(X, 3, 4)/gx(X)
  mean(w*(X>z))
}

  isF(40, n)

})
```

```
##    user  system elapsed
##    0.01    0.00    0.02
```

The second method is more efficient compared to the first one.

5

(a)

```r
n=10^4
fx=function(x) {sqrt(2/pi)*exp(-x^2/2)*x^2}

simN=function(a, b){
  fx=function(x)  {sqrt(2/pi)*exp(-x^2/2)}
  gx=function(x) {1/2*exp(-abs(x))}
  M=optimize(f=function(x){fx(x)/gx(x)}, maximum = T,
             interval=c(0, 100))$objective
  while(T){
    u=runif(1)
    X=log(2*u)*(u<=1/2)-log(2*(1-u))*(u>1/2)
    Y=runif(1, 0, M*gx(X))
    if(Y<fx(X)) return(b*(2*rbinom(1, 1, 1/2)-1)*X+a)
  }
}


randN1=replicate(n, simN(0, 1))
mean((randN1>1)*randN1^2)
```

```
## [1] 0.4042714
```

```r
# true value
randN0=rnorm(n)
```

```r
mean((randN0>1)*randN0^2)
```

```
## [1] 0.4053364
```

(b)

```r
n=10^4

rnormt=function(n, range) {
  F.a=pnorm(min(range))
  F.b=pnorm(max(range))

  u=runif(n, min = F.a, max = F.b)

  qnorm(u)

}

isN1=function(z,n){
  u=runif(n)
  gx=function(x) {1/(sqrt(2*pi)*(1-pnorm(1)))*exp(-0.5*x^2)}
  X=rnormt(n, c(1, Inf))
  w = dnorm(X)/gx(X)
  mean(w*(X>z)*X^2)
}

isN1(1, n)
```

```
## [1] 0.4000806
```

```r
# true value
randN0=rnorm(n)
mean((randN0>1)*randN0^2)
```

```
## [1] 0.4100074
```

(c)

```r
n=10^4

isN2= function(z,n){
  u=runif(n)
  hx=function(x) {x*exp((1-x^2)/2)}
  X=sqrt(1-2*log(1-u))
  w = dnorm(X)/hx(X)
  mean(w*(X>z)*X^2)
}
```

```r
isN2(1, n)
```

```
## [1] 0.4032079
```

```r
# true value
randN0=rnorm(n)
mean((randN0>1)*randN0^2)
```

```
## [1] 0.4054365
```

(d)

```r
set.seed(1)

# (a)
mean((randN1>1)*randN1^2)
```

```
## [1] 0.4042714
```

```r
sd((randN1>1)*randN1^2)
```

```
## [1] 1.121329
```

```r
# (b)
n=10^4
z=1

u=runif(n)
gx=function(x) {1/(sqrt(2*pi)*(1-pnorm(1)))*exp(-0.5*x^2)}
randN2=rnormt(n, c(1, Inf))
w=dnorm(randN2)/gx(randN2)

mean(w*(randN2>z)*randN2^2)
```

```
## [1] 0.4003726
```

```r
sd(w*(randN2>z)*randN2^2)
```

```
## [1] 0.2598597
```

```r
# (c)
n=10^4
z=1

u=runif(n)
hx=function(x) {x*exp((1-x^2)/2)}
randN3=sqrt(1-2*log(1-u))
w = dnorm(randN3)/hx(randN3)

mean(w*(randN3>z)*randN3^2)
```

```
## [1] 0.3987094
```

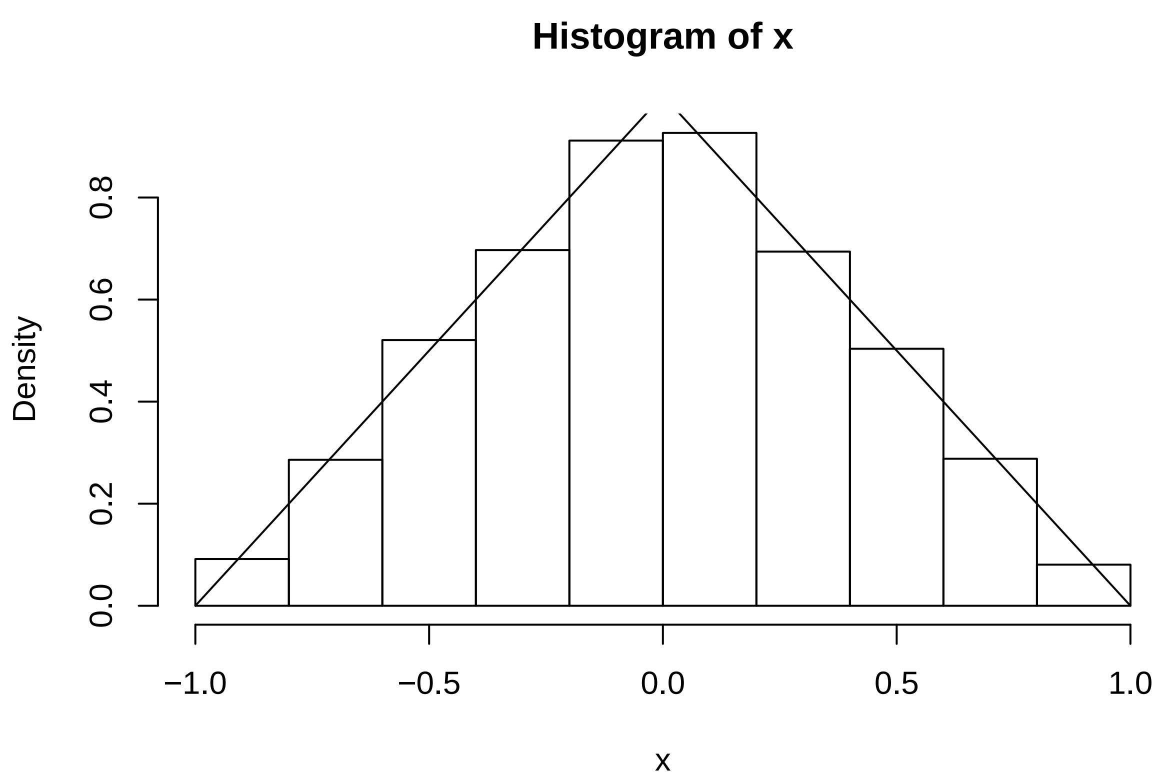
```r
sd(w*(randN3>z)*randN3^2)
```

```
## [1] 0.1218829
```

The estimate of θ from (a), (b), (c) is shown above. All three estimates are close to the true estimate calculated with rnorm(). The standard error of each simulation differs though. The estimate sampled from rejection sampling shown in (a) has the largest standard error. The estimate sampled from importance sampling with the truncated normal distribtuion has the second largest standard error. Estimate from (c) has the smallest standard error.

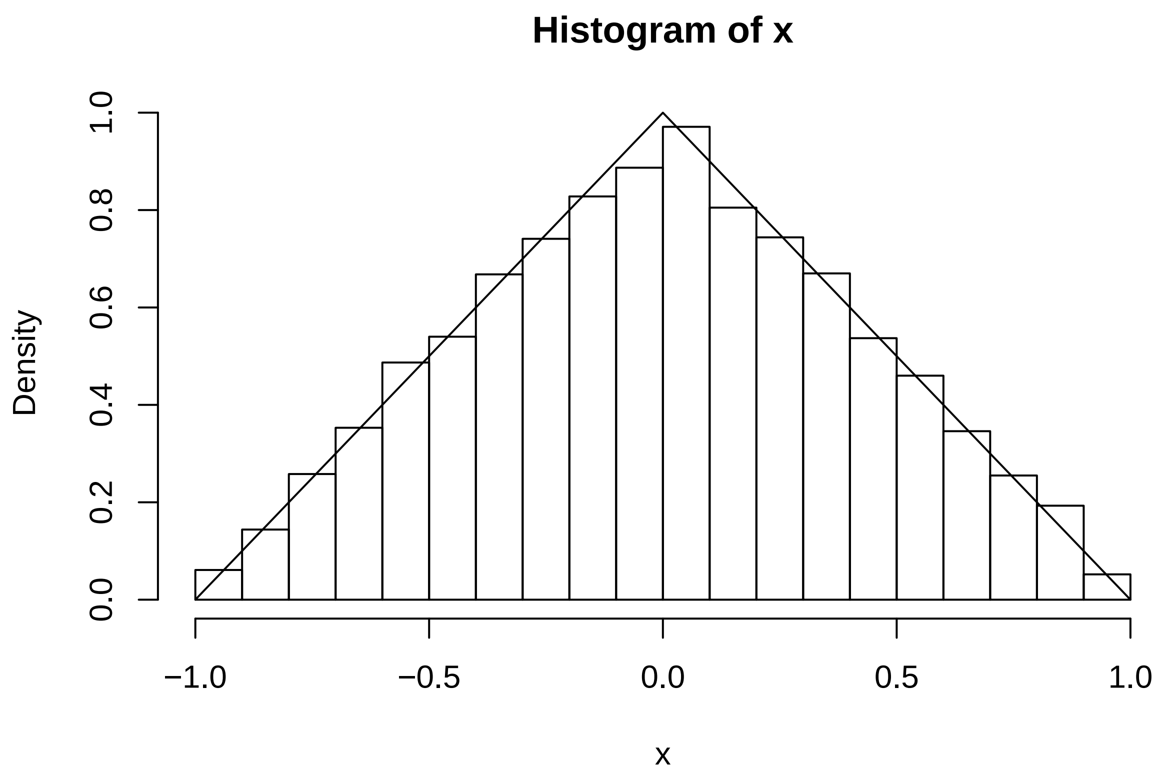
6

```r
# rejection method
u1=runif(10^4, -1, 1)
u2=runif(10^4)
x=u1[u2<(1-abs(u1))]
hist(x, freq=F)
curve(1-abs(x), add=T)
```




```r
# inverse transform method
n=10^4
u=runif(n)
```

```
x=(-1+sqrt(2*u))*(u<1/2)+(1-sqrt(2*(1-u)))*(u>1/2)
hist(x, freq=F)
curve(1-abs(x), add=T)
```



The first method is simulating the random variable with rejection sampling. First simulate a point (u1, u1) uniformly in the rectangle. If (u1, u2) is located within the triangle region, accept u1 as a random variable x. Else, reject it and return to the first step.

The second method is simulating by inverse transform method. From the triangular pdf provided we can calculate the cdf $F(x)$.

$$F(x) = \begin{cases} \frac{(x+1)^2}{2}, & -1 < x < 0 \\ 1 - \frac{(x-1)^2}{2}, & 0 \leq x \leq 1 \end{cases}$$

Equating the cdf to u which follows a uniform distribution, yields an inverse cdf $F(x)^{-1}$

$$F(u)^{-1} = \begin{cases} -1 + \sqrt{2u}\,, & 0 < u < \frac{1}{2} \\ 1 - \sqrt{(1-u)^2}, & \frac{1}{2} \leq u < 1 \end{cases}$$