STAT 428: Homework 2: Random Number Generation and Sampling

Du, Yuting(Iris), yutingd3 Collaborated with: Sun, Yifan, yifans8

Due Week 5, Saturday Feb 22 by 11:59 PM on Compass

Table of Contents

Please refer to the **detailed homework policy document** on Course Page for information about homework formatting, submission, and grading.

Exercise 1

(a) For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

```
vector1 <- c("8", "12", "7", "32")
min(vector1)
sort(vector1)
sum(vector1)
```

(b) For the next series of commands, either explain their results, or why they should produce errors.

```
vector2 <- c("5",7,12)
vector2[2] + vector2[3]

dataframe3 <- data.frame(z1="5",z2=5,z3=12)
dataframe3[1,2] + dataframe3[1,3]

list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]]
list4[2]+list4[4]
```

Solution 1

(a) The first command will execute the smallest first element in the vector. The second will follow the same rule. The third command will result in an error since the data type is not addable. **(b)** The output type is determined from the highest type of components in the vector.The first command will not execute since there is non-numeric element in the vector which could not be calculated. The second command will result in 17. Since only the numeric data are invovled in the operation. The first command in the thrid chuck will execute the summation of the second and the forth element in the given list because this operation uses the copies of elements. However, the following command gets the slices of the list.

Exercise 2

Working with functions and operators.

(a) The colon operator will create a sequence of integers in order. It is a special case of the function `seq()`. Using the help command `?seq` to learn about the function, design an expression that will give you the sequence of numbers from 1 to 20000 in increments of 582. Design another that will give you a sequence between 1 and 20000 that is exactly 52 numbers in length.

(b) The function `rep()` repeats a vector some number of times. Explain the difference between `rep(1:5, times=6)` and `rep(1:5, each=6)`.

Solution 2

```
seq(1, 20000, 582)
```

```
##  [1]     1   583  1165  1747  2329  2911  3493  4075  4657  5239
5821  6403
## [13]  6985  7567  8149  8731  9313  9895 10477 11059 11641 12223
12805 13387
```

```
## [25] 13969 14551 15133 15715 16297 16879 17461 18043 18625 19207
19789
```

```
seq(1, 20000, length.out = 52)
```

```
##  [1]     1.0   393.1   785.3  1177.4  1569.5  1961.7  2353.8  2746.0
3138.1
## [10]  3530.2  3922.4  4314.5  4706.6  5098.8  5490.9  5883.1  6275.2
6667.3
## [19]  7059.5  7451.6  7843.7  8235.9  8628.0  9020.2  9412.3  9804.4
10196.6
## [28] 10588.7 10980.8 11373.0 11765.1 12157.3 12549.4 12941.5 13333.7
13725.8
## [37] 14117.9 14510.1 14902.2 15294.4 15686.5 16078.6 16470.8 16862.9
17255.0
## [46] 17647.2 18039.3 18431.5 18823.6 19215.7 19607.9 20000.0
```

```
rep(1:5, times= 6)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
rep(1:5, each = 6)
```

```
##  [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5
```

`rep(1:5, times=6)` will repeat the range of numbers in order by 6 times. `rep(1:5, each=6)` will repeat the element each by 6 times in order.

Exercise 3

Writing R functions. Write an R function that accepts an $n \times n$ matrix A as an argument and returns the trace of A, $trace(A) = \sum_{i=1}^{n} a_{ii}$. Include code that verifies that the function's argument is indeed a square matrix. Also include comments that describe the function's purpose, argument and output. The following code generates a list of 25 matrices of different dimensions. Apply your trace function to each matrix in this list.

```
matrix.list <- lapply(1:25, FUN=function(x) {matrix(1:x^2, nrow=x,
ncol=x)})
#The given list of matrices

findTrace <- function(x){
    n <- dim(x)[1] # get the dimension of the matrix
    m <- dim(x)[2]

    # check for squared
    if(n == m){
      tr <- 0 # intialize the trace value
       #loop over, add the diagonal elements of the squared matrix to
the tr
    for(k in 1: n){
```

```
      ele <- x[k, k]
      tr <- ele + tr
    }
    return(tr[[1]])
    }
    else {
      message('Matrix is not square')
    }
}

lapply(matrix.list, FUN = findTrace)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 5
##
## [[3]]
## [1] 15
##
## [[4]]
## [1] 34
##
## [[5]]
## [1] 65
##
## [[6]]
## [1] 111
##
## [[7]]
## [1] 175
##
## [[8]]
## [1] 260
##
## [[9]]
## [1] 369
##
## [[10]]
## [1] 505
##
## [[11]]
## [1] 671
##
## [[12]]
## [1] 870
##
## [[13]]
## [1] 1105
```

```
##
## [[14]]
## [1] 1379
##
## [[15]]
## [1] 1695
##
## [[16]]
## [1] 2056
##
## [[17]]
## [1] 2465
##
## [[18]]
## [1] 2925
##
## [[19]]
## [1] 3439
##
## [[20]]
## [1] 4010
##
## [[21]]
## [1] 4641
##
## [[22]]
## [1] 5335
##
## [[23]]
## [1] 6095
##
## [[24]]
## [1] 6924
##
## [[25]]
## [1] 7825
```

Exercise 4

Simulating Beta(3, 4) Random Variables: Which method is better?

In this exercise, you will simulate Beta(3,4) random variables with target density

$$f(x) \propto x^2\ (1-x)^3.$$

using the Accept/Reject Algorithm with two different instrumental densities.

- **[Method I]** Use uniform RV's with instrumental density

$$g_1(x) = 1, \quad 0 < x < 1.$$

  - you may use $M \approx 0.0346$.
  - to simulate from g_1 use the command `x <- runif(1)`.
- **[Method II]** Use the average of two uniform RV's with instrumental density

$$g_2(x) = 2 - |4x - 2|, \quad 0 < x < 1.$$

  - you may use $M \approx 0.0264$.
  - the R function for absolute value is `abs`.
  - to simulate from g_2 use the command `x <- mean(runif(2))`.
- **[Question]** Which method is better? Answer this by responding to the following parts.

(a) Write a program that simulates 10,000 Beta(3,4) random variables using both methods.

(b) Compare the acceptance rates and decide which one is the better algorithm.

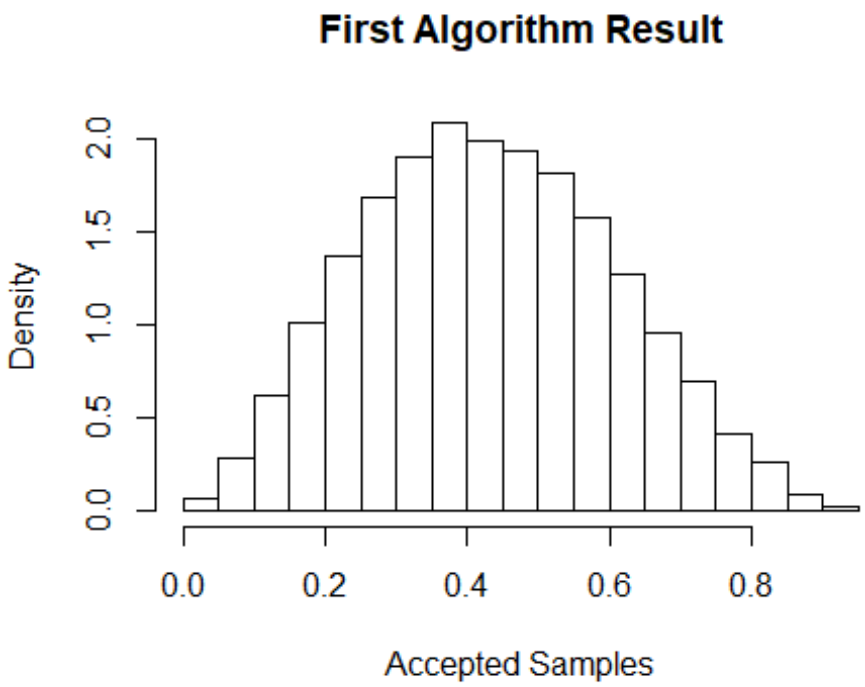
(c) Is it correct to use theoretical acceptance rate $\frac{1}{M}$? Why or why not?

(d) Graph histograms of your results.

(e) Summarizes your findings.

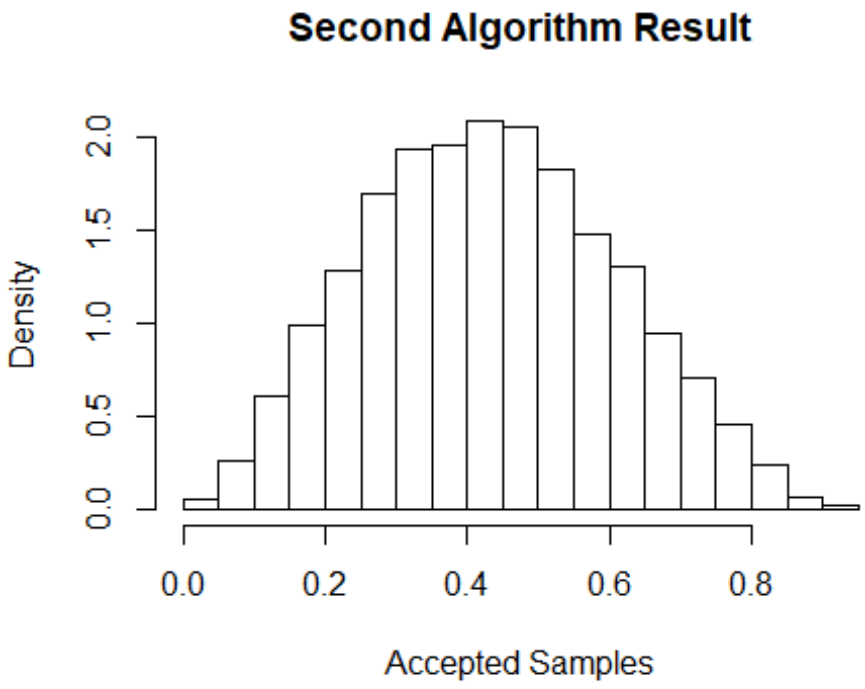
Solution 4

```
algorithm <- function(n){
  M <- 0.0346
  accept <- 0; num <- 0
  z <- rep(0, times = n)
  while(accept < n){
      x <- runif(1)
      f <- (x^2) * (1 - x)^3
      u <- runif(1)
      g <- 1
      if (u <= f/M/g){
        accept <- accept + 1
        z[accept] <- x
      }
      num <- num + 1
    }
list(z = z, accept = n/num)
}
method1 <- algorithm(10000)
method1$accept

## [1] 0.4774

hist(method1$z, freq = FALSE, breaks = 30, main = "First Algorithm
Result", xlab = "Accepted Samples")
```




```
# nsim = 10000
# k = 0 #counter for accepted samples
# j = 0 #No of iterations required to get desired sample size
# y = numeric(nsim)    #Storing the sample
# M = 0.0246
#
# while(k < nsim){
#   u = runif(1)
#   j = j + 1
#   x = mean(runif(2))
#   g = 2- abs(4*x-2)#random variate from reference distribution g(x)
#   if((g^2)*(1-g)^3 > M*u){
#     #Condition of accepting x in our sample
#     k = k + 1
#     y[k] = g
#   }
# }
#
# hist(y, prob = TRUE)
algorithm <- function(n){
  M <- 0.0246
  accept <- 0; num <- 0
  z <- rep(0, times = n)
  while(accept < n){
      x <- mean(runif(2))
      f <- (x^2) * (1 - x)^3
      u <- runif(1)
```

```
      g <- 2 - abs(4 * x - 2)
      if (u <= f/M/g){
        accept <- accept + 1
        z[accept] <- x
      }
      num <- num + 1
    }
list(z = z, accept = n/num)
}
method2 <- algorithm(10000)
method2$accept

## [1] 0.6667

hist(method2$z, freq = FALSE, breaks = 30, main = "Second Algorithm
Result", xlab = "Accepted Samples")
```



(b) The second method has higher acceptance rate. The second one is better. **(c)** No. Since f(x) is only proportional to $x^2(1-x)^3$, We could not have the constant and get upper bound as M', then the $P(acceptance)$ is no longer $\frac{1}{M'}$. **(e)** From the plot, both methods have a good performance on simulation, though they may differ in acceptance rate.

Exercise 5

Simulating a mixture distribution.

Consider a density function of the form

$$f(x) = \sum_{k=1}^{K} \pi_k \, f_k(x)$$

where $\pi_k > 0$ for each k and $\sum_{k=1}^{K} \pi_k = 1$. Also assume that each f_k is a probability density function for $k = 1,2,\ldots,K$.

Then it's not hard to see that $f(x)$ is a probability density function. Furthermore, a random variable X with pdf $f(x)$ is said to arise from a mixture of K distributions. The constants $\pi_k > 0$ are called mixing probabilities or mixing weights.

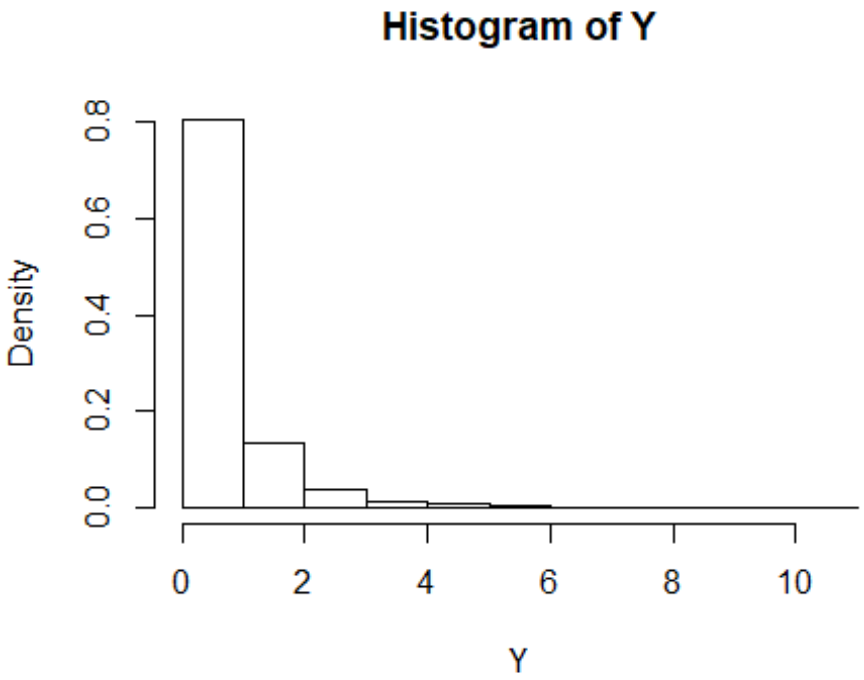
To draw observations from this mixture distribution, we first choose a distribution, say ith distribution $f_i(x)$ to sample from based on the mixing probabilities and then generate from that i-th density like you normally would. The resulting variates would be a mixture of the K distributions.

Now, do problem 3.12.

Solution 5

3.12 Simulate a continuous Exponential-Gamma mixture. Suppose that the rate parameter Λ has Gamma(r, β) distribution and Y has Exp(Λ) distribution. That is, $(Y|\Lambda = \lambda) \sim fY(y|\lambda) = \lambda e^{-}\lambda y$. Generate 1000 random observations from this mixture with r = 4 and β = 2.

```
# Λ~gamma(r, β)
#Y~exp(Λ)

r = 4
s = 2

set.seed(123)
u = runif(1000)
Y = s*(1/((1-u)^(1/r))-1)


hist(Y, prob= TRUE)
```



Exercise 6

χ^2 **distribution** There are many ways to generate a $\chi^2(v)$ distribution. Consider the following methods. For each method, plot a historgram of the resulting variables: do they all seem equally effective?

(a) Simulate 10000 pairs of random variables (Z_1, Z_2), both from $\mathcal{N}(0,1)$ using the transformation method 4 under "3.4: Transformation methods" in the book. Use these pairs to generate 10000 $\chi^2(2)$ random variables.

(b) How does the exponential distribution relate to a $\chi^2(2)$ distribution? (Do a little research!) Generate 10000 random variables from a useful exponential distribution using CDF inversion for this case. Use them to calculate 10000 random variables from a $\chi^2(2)$ distribution.

(c) Generate 10000 random variables from a gamma distribution such that the resulting variables will be $\chi^2(2)$. (You can use an r function to generate the gamma variables.)

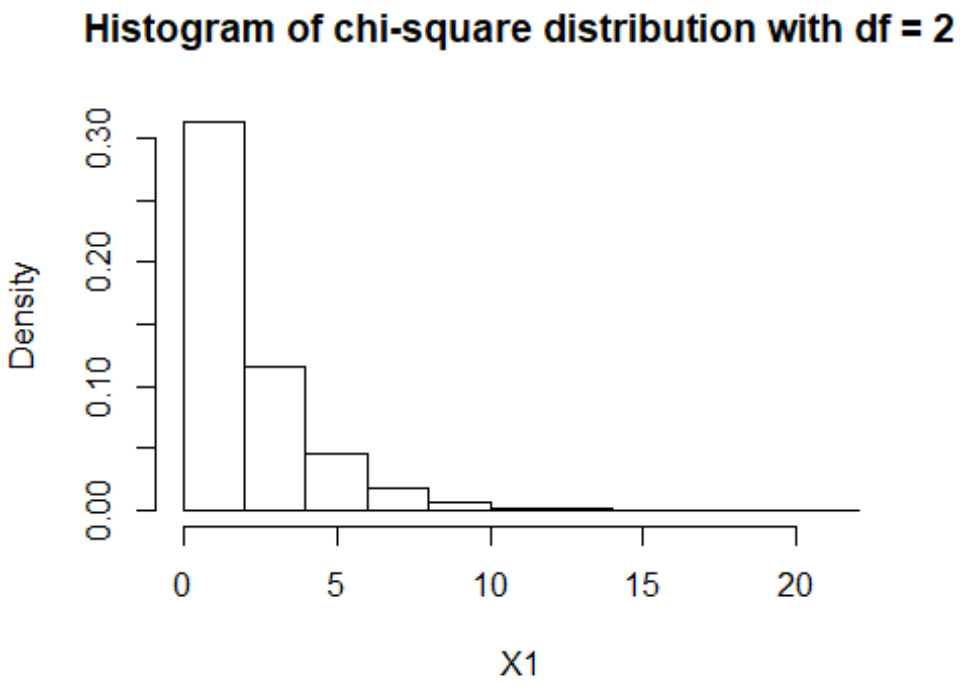
Solution 6

```
# simulation from normal distribution
# set.seed(123)
# Z1 = rnorm(10000)
# set.seed(124)
# Z2 = rnorm(10000)
```

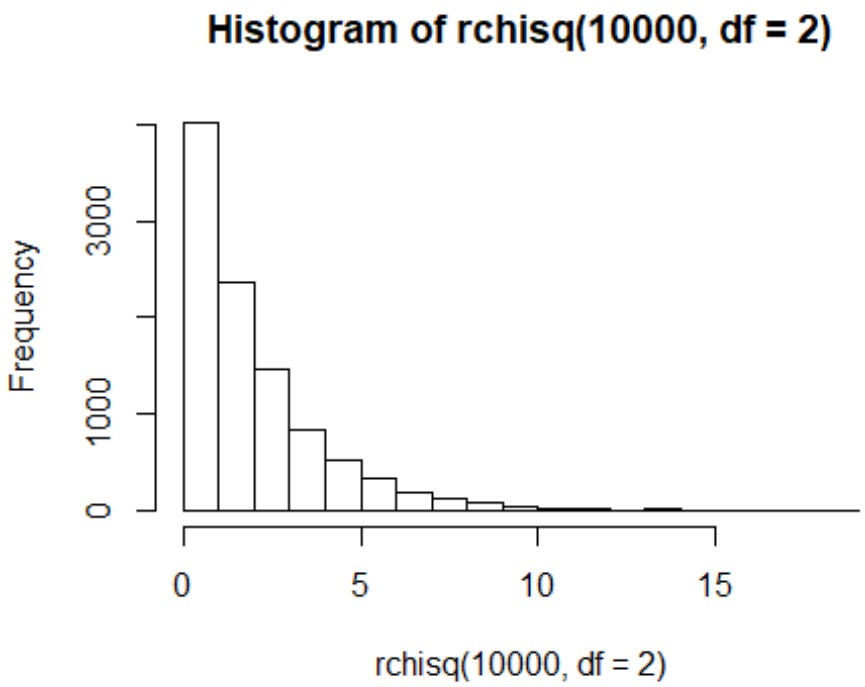
```r
#  using the transformation method 4 under "3.4: Transformation
methods" in the book
u = runif(10000)
v = runif(10000)
z1 = sqrt(-2*log(u))*cos(2*pi*v)
z2 = sqrt(-2*log(v))*sin(2*pi*u)

X1 = z1^2 + z2^2
mean(X1)
```

```
## [1] 2.018
```

```r
hist(X1, prob= TRUE, main= "Histogram of chi-square distribution with
df = 2")
```



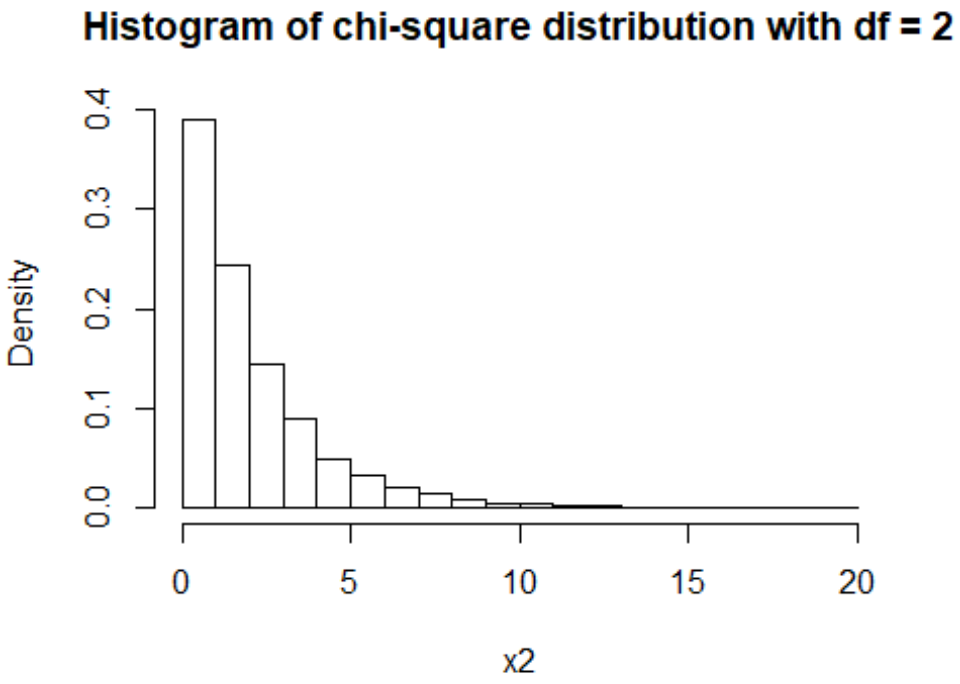

```r
#compare
hist(rchisq(10000,df=2))
```



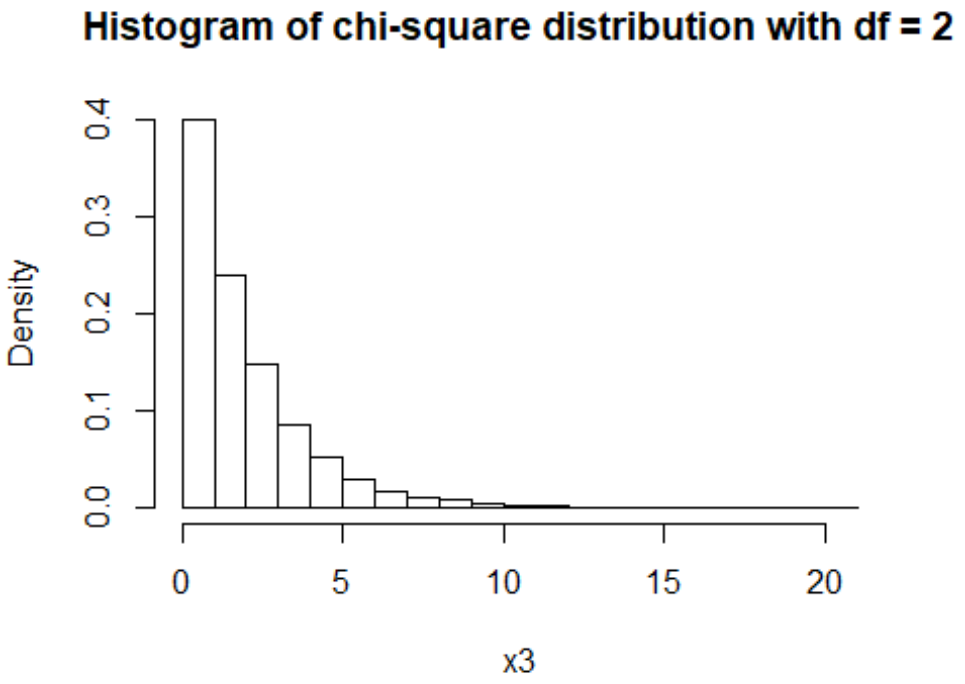

```r
mean(rchisq(10000,df=2))
```

```
## [1] 1.996
```

```r
#from expontenial distribution??
# A chi-squared distribution with 2 degrees of freedom is an
exponential distribution with mean 2 and vice versa.
#use inverse CDF
set.seed(123)
u = runif(10000)
x2 = -2*log(u)
hist(x2, prob=TRUE, main = "Histogram of chi-square distribution with
df = 2")
```




```
mean(x2)

## [1] 2.008

#from gamma distribution
set.seed(123)
df = 2
x3 = rgamma(10000, df/2, 1/2)
hist(x3, prob=TRUE, main = "Histogram of chi-square distribution with
df = 2")
```




```
mean(x3)
```

```
## [1] 1.968
```

For Exerise 7-10, do the following problems from the book 3.2, 3.3, 3.5, 3.9.

Exerise 7 (3.2)

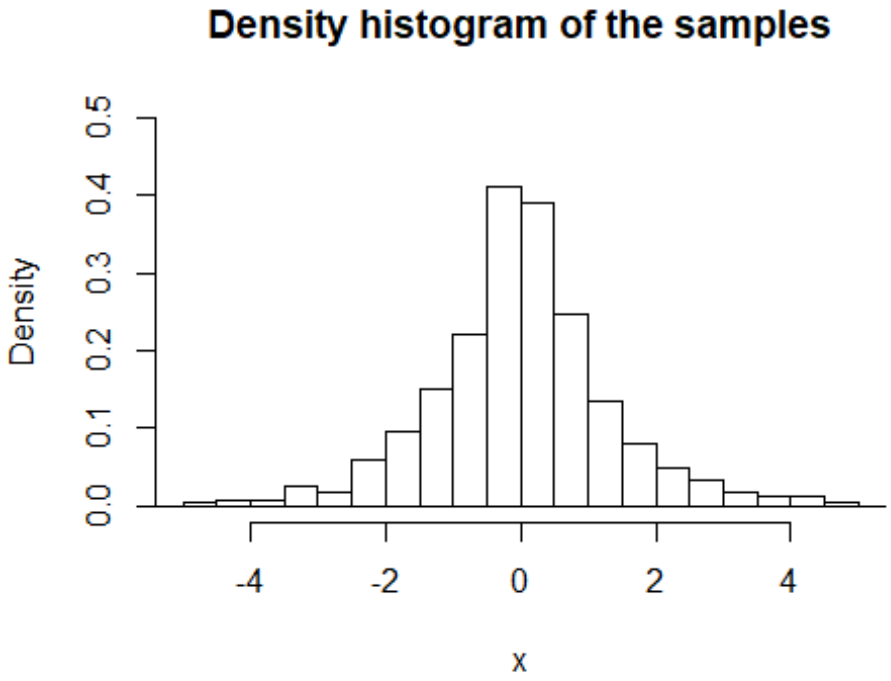
The standard Laplace distribution has density $f(x) = 1/2 * e^{-}|x|$, x ∈ R . Use the inverse transform method to generate a random sample of size 1000 from this distribution. Use one of the methods shown in this chapter to compare the generated sample to the target distribution.

Solution 7

```
# x ∈ R??
set.seed(123)
u = runif(1000)

indicator = numeric()
for(i in 1:1000){
  if(u[i] > 0.5) indicator[i] = 1
  else indicator[i] = -1
}

x = -indicator * log(1-2*abs(u-0.5))
mean(x)
```

```
## [1] -0.01599
```

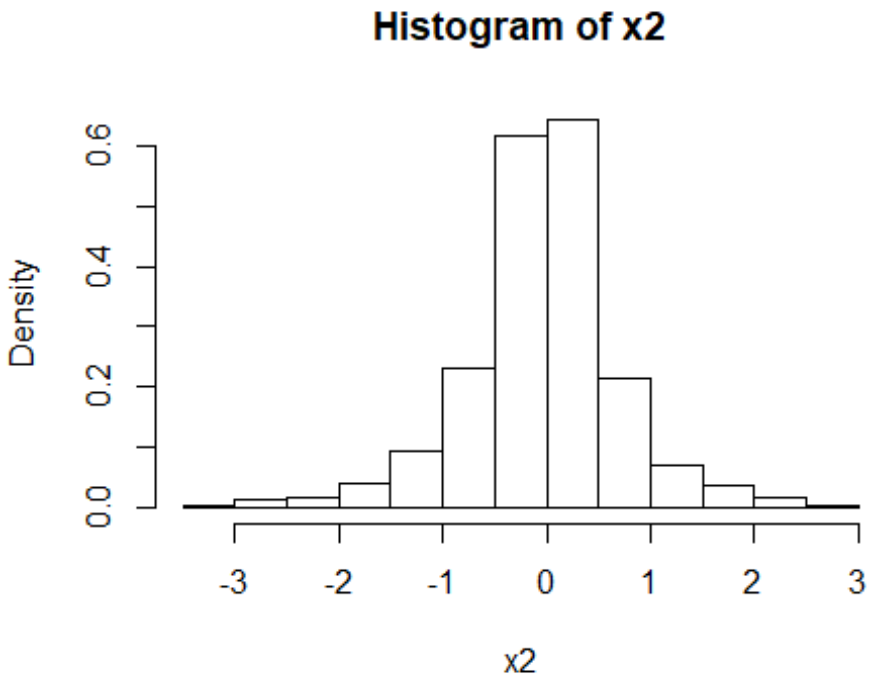
```
hist(x, probability = T, xlim=c(-5,5), ylim=c(0,0.5), breaks = 40,
main="Density histogram of the samples", xlab = "x")
```




```
#Use one of the methods shown in this chapter to compare the
#generated sample to the target distribution
x2 = 0.5*indicator*rexp(n = 1000, rate = 1)
mean(x2)
```

```
## [1] -0.03766
```

```
hist(x2, probability = TRUE)
```

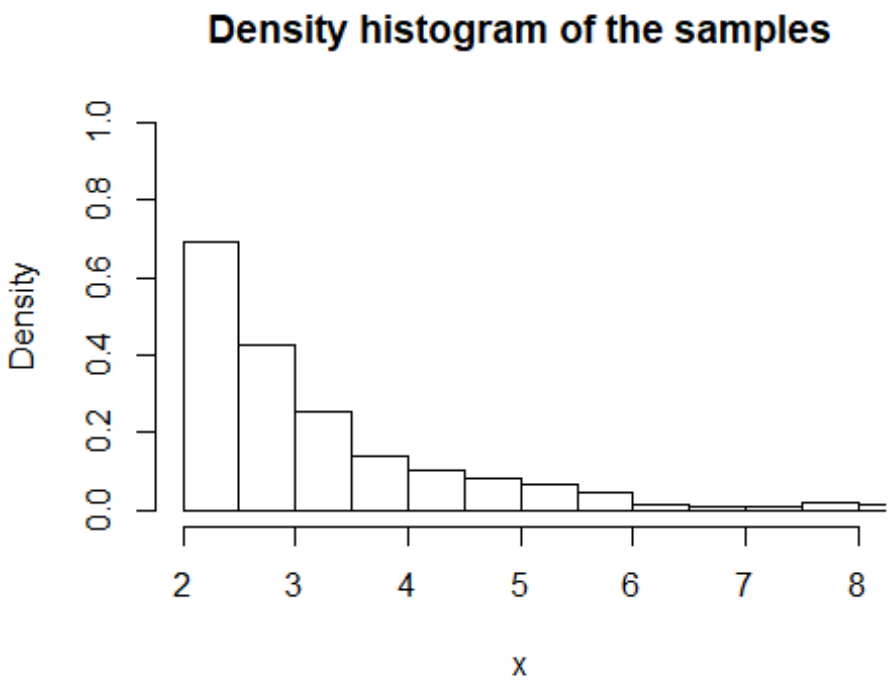



```
#similar distribution
```

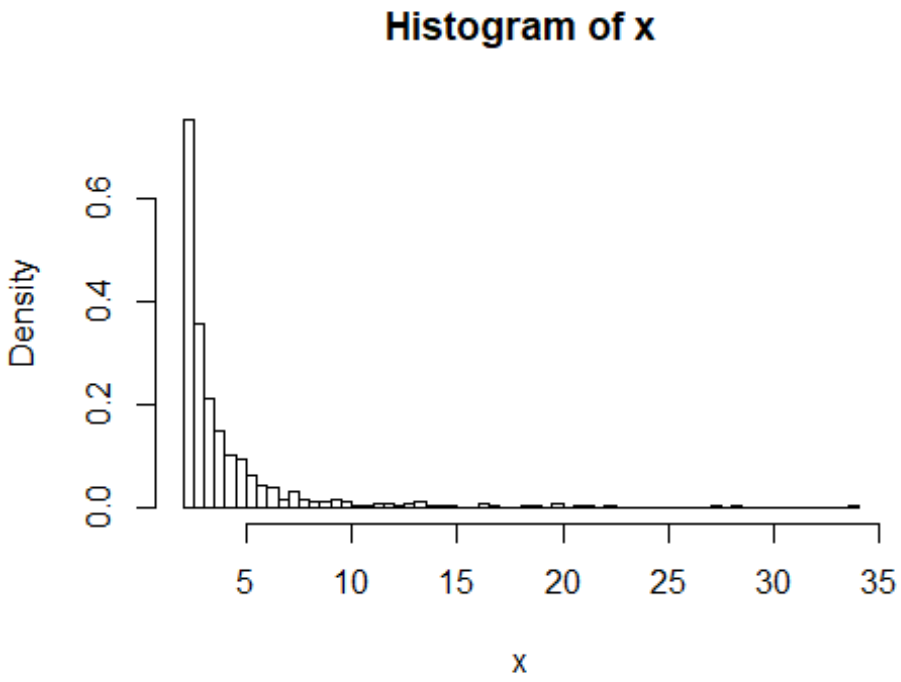
Exerise 8 (3.3)

Solution 8

```
u <- runif(1000)
f <- 2/sqrt(1-u)
hist(f, probability = T, breaks=200, xlim=c(2,8), ylim=c(0,1),
main="Density histogram of the samples", xlab = "x")
```




```r
# #for comparsion
library(extraDistr)
x <- rpareto(1000, 2, 2)
hist(x, 100, probability = T)
```



Exerise 9 (3.5)

```r
set.seed(123)
x <- c(0, 1, 2, 3, 4)
prob <- c(0.1, 0.2, 0.2, 0.2, 0.3)

sample <- sample(x, size=1000, prob = prob, replace = TRUE)
table(sample)
```

```
## sample
##   0   1   2   3   4
##  92 203 211 198 296
```

```r
Empirical_Prob = table(sample)/1000
data.frame(Empirical_Prob,prob)
```

```
##   sample  Freq prob
## 1      0 0.092  0.1
## 2      1 0.203  0.2
## 3      2 0.211  0.2
## 4      3 0.198  0.2
## 5      4 0.296  0.3
```

```r
#the sample result is similar to the empirical probability
```

Exercise 10 (3.9)

```
u1 = runif(5000, -1, 1)
u2 = runif(5000, -1, 1)
u3 = runif(5000, -1, 1)
data = ifelse(abs(u3) >= abs(u2) & abs(u3) >= abs(u1), u2, u3)
hist(data, freq = F, main="Density histogram of the samples", xlab =
"x", breaks = seq(-1, 1, by = 0.05))
```