Statistical Computing Ch.3 Homework

Wei-Chen Chang r12227118

Due: 2024-03-13

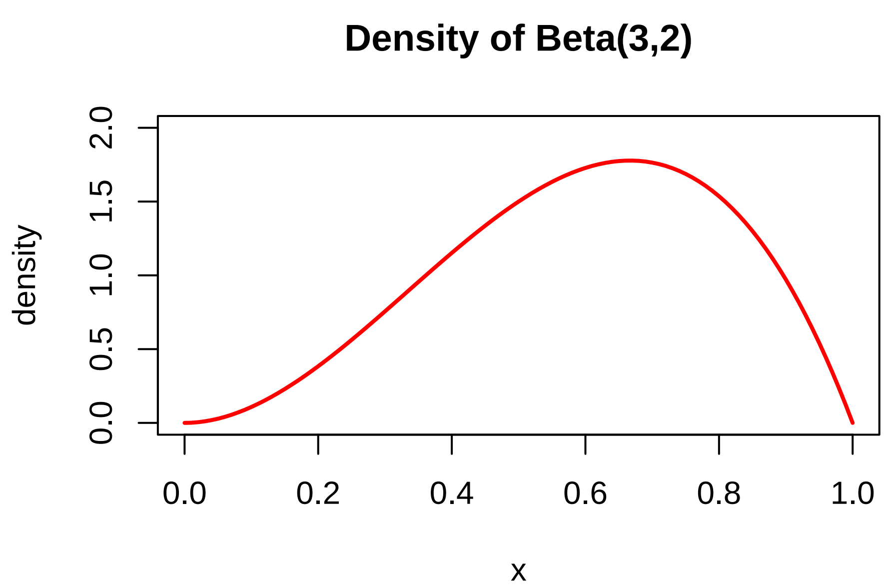
Q1

Generate a random sample of size 1000 from the $Beta(3,2)$ distribution using the acceptance-rejection method. Graph the histogram of the sample with the theoretical $Beta(3,2)$ density superimposed as the Figure below.

Ans:

Fist take a look into the target distribution:

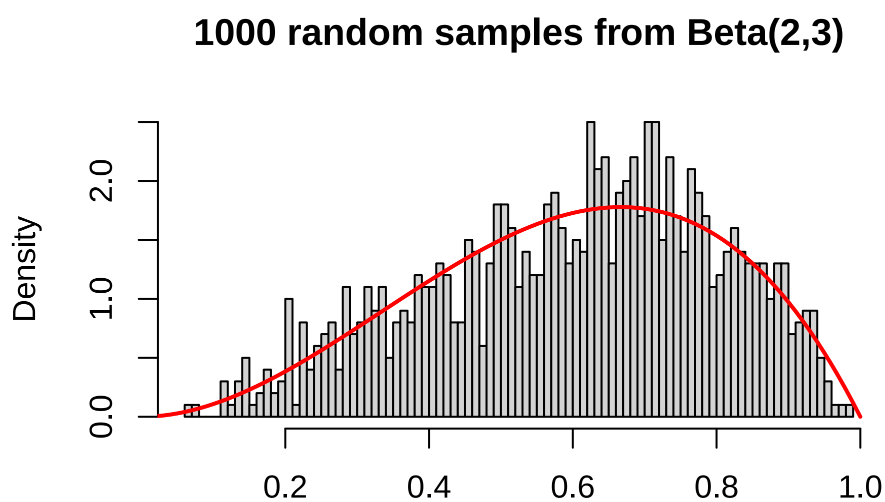
```
curve(dbeta(x, 3,2), from = 0, to = 1, lwd =2, col="red",
      ylab = "density",
      ylim = c(0,2),
      main = "Density of Beta(3,2)")
```



As the beta distribution has the support on $[0,1]$, and from the graph we can see its maximum is lesser than 2. Therefore, $Unif(0,1)$ was selected as the envelope distribution, and set $C = 2$ for the A-R method.

```r
set.seed(87)
generate_beta <- c()
c <-  2 # criterion

while (length(generate_beta) < 1000){
  y <- runif(1) # envelope
  u <- runif(1) # random uniform
  if (u < dbeta(y,3,2)/c){
    generate_beta = c(generate_beta , y)
  }
}
# cat(length(generate_beta))
generate_beta %>% hist(freq = FALSE, breaks = 100,main = "1000 random samples from Beta(2,3)")
curve(dbeta(x, 3,2), from = 0, to = 1, lwd =2, col="red",add = TRUE)
```



.

Q2

Generate a random sample of size 1000 from the pdf:

$$f(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2},\ -R \le x \le R$$

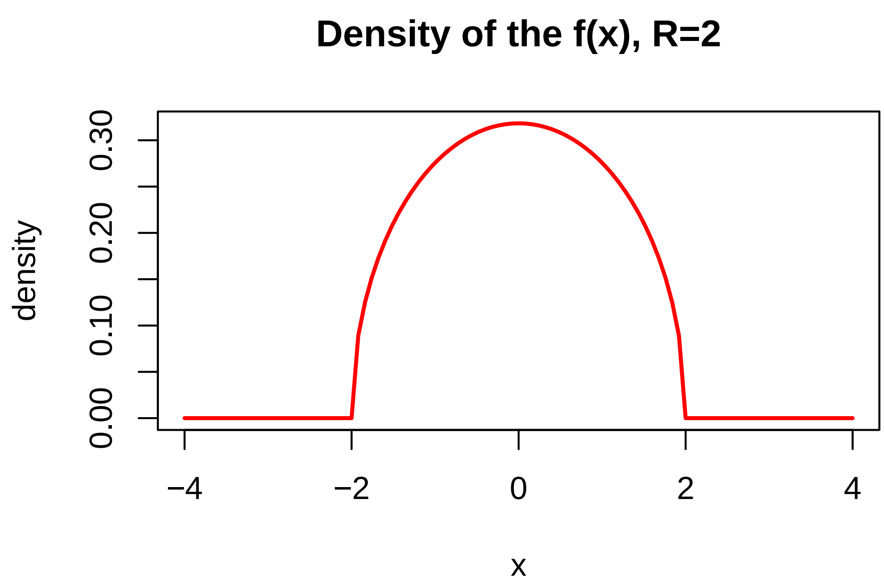
using the acceptance-rejection method. Graph the histogram of the sample with the theoretical density superimposed.

Ans:

First take a look at the target distribution:

```r
dhemisph <-  function(x,r){
  prob_density <- ifelse(abs(x)<r, 2/(pi*r^2)*sqrt(r^2-x^2), 0)
  return(prob_density)
}

curve(dhemisph(x, r=2), from= -4, to=4,
      lwd=2, col="red", ylab ="density",
      main = "Density of the f(x), R=2")
```



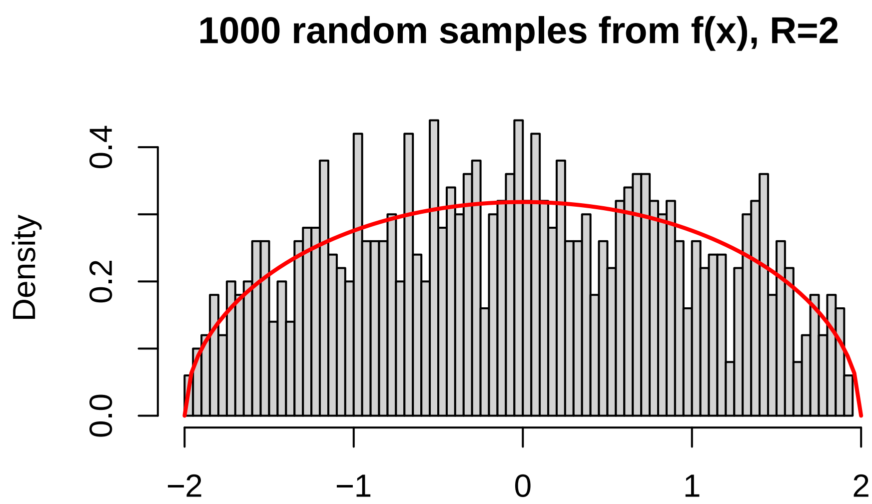
Observation:

From its pdf or the graph above, the density has the maximum at $x = 0$, which equals to $\frac{2}{\pi R}$. Also, here we select $Unif(-R, R)$ has the density $\frac{1}{R-(-R)} = \frac{1}{2R}$. To let the A-R method efficient, we set $c = \frac{\frac{2}{\pi R}}{\frac{1}{2R}} = \frac{4}{\pi}$ such that c times $Unif(-R, R)$ just covered the peak of the target distribution.

Set R=2 to show the random sampling as below:

```r
set.seed(87)
generate_hemisphere <- c()
c <-  4/pi # criterion
R <- 2 # model parameter R
while(length(generate_hemisphere) < 1000){
  y <- runif(1, -R, R) # envelope
  u <- runif(1) # random uniform sample
  if (u < dhemisph(y,r=R)/c){
    generate_hemisphere = c(generate_hemisphere , y)
  }
}
# plot
generate_hemisphere %>% hist(freq = FALSE, breaks = 100,
```

```
                                main = paste0("1000 random samples from f(x), R=",R))
curve(dhemisph(x, r=R), from = -R, to = R, lwd =2, col="red",add = TRUE)
```



.

Q3

The continuous random variable X with positive support is said to have the Pareto distribution if its probability density function is given by

$$f(x) = \frac{\beta\alpha^{\beta}}{(x+\alpha)^{\beta+1}},\ x > 0$$

Generate a random sample of size 1000 from the Pareto distribution with $\alpha = 2$ and $\beta = 4$ using "inverse transformation method". Compare the empirical and theoretical distributions by graphing the histogram of the sample and superimposing the Pareto density curve.

Ans:

First, integrate $f(x)$ to derive its cdf $F(x)$ then find its inverse function $F^{-1}(x)$, as below:

$$F(x) = 1 - (\frac{\alpha}{x+\alpha})^{\beta}, F^{-1}(x) = \frac{\alpha}{(1-x)^{\frac{1}{\beta}}} - \alpha$$

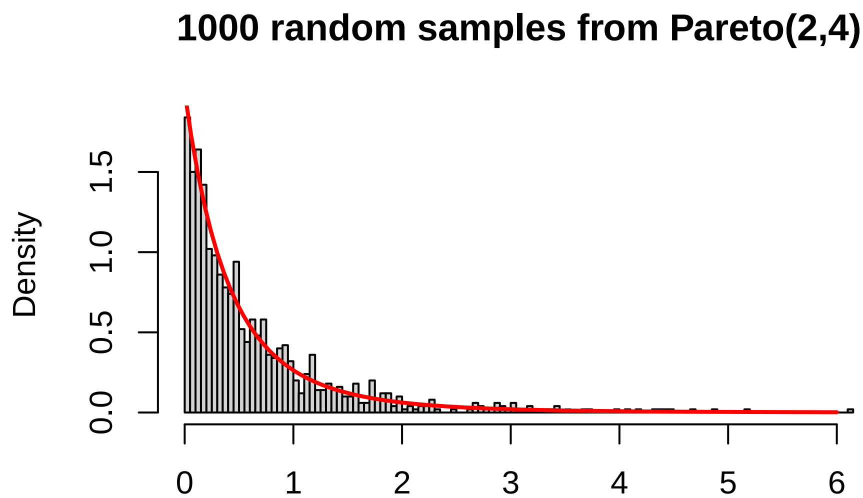
Then defining these functions and apply inverse transformation method to produce random samples:

```
# density
dpareto <- function(x, alpha=2, beta=4){
  return(beta*alpha^beta/(x+alpha)^(beta+1))
}

# quantile (inverse)
```

```
qpareto <- function(x, alpha=2, beta=4){
  return(alpha/((1-x)^(1/beta))-alpha)
}

# Inverse transformation method
set.seed(87)
u <- runif(1000) # 1000 random uniform distribution
qpareto(u) %>% hist(breaks = 100,freq = FALSE,
                    main="1000 random samples from Pareto(2,4)")
curve(dpareto(x), from=0, to=6,
      lwd=2, col="red", add= TRUE)
```



.

Q4

Please compare the efficiencies of two methods for generating the Wishart Distribution with n = 100 and covariance $\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$

Ans:

Method 1:

1. Decompose $\Sigma = \mathbf{QQ^T}$ via spectral decomposition or Cholesky decomposition.
2. Generate 100 samples from two independent standard normal distribution: $\mathbf{Z} = \begin{pmatrix} Z_1 & Z_2 \end{pmatrix}^T \sim N(0, I_2)$ and multiply it by $\mathbf{Q}$ to create the bivariate normal distributed samples $\mathbf{X_{2\times 100}} = \mathbf{QZ} \sim N(\mathbf{0}, \mathbf{\Sigma})$.
3. Lastly, compute $\mathbf{W} = \mathbf{XX^T}$, and $\mathbf{W} \sim Wishart(100, \Sigma)$

Method 2:

1. Generate a lower-triangular matrix $\mathbf{A}$ where its diagonal position A_{ii} follows a $\sqrt{\chi^2(n-i+1)}$ and its lower part $A_{ij} \sim^{iid} N(0,1)$
2. Decompose $\Sigma = \mathbf{LL^T}$ via Cholesky decomposition, where $\mathbf{L}$ is lower-triangular.
3. Compute $\mathbf{W} = \mathbf{LAA^TL^L}$, and $\mathbf{W} \sim Wishart(100, \Sigma)$

Functions

```r
# Method 1
wishart_method1 <- function(n, covMtx){
  if(is.matrix(covMtx)==F){
    break
  }
  # spec
  # .a <- eigen(covMtx) # attain eigenvalues/vectors
  # Q <-.a$vectors%*%diag(sqrt(.a$values)) #
  Q <- chol(covMtx) %>% t() #lower triangular by Cholesky decomp.
  # generate Zs and Xs
  z1 <-  rnorm(n)
  z2 <-  rnorm(n)
  zMtx <-  matrix(c(z1,z2), nrow=2, byrow = TRUE)
  X <- Q %*% zMtx #2x100
  # cov(X[1,],X[2,]) # Check if the simulation reasonable
  wishart <-  X %*% t(X)  # 2*2
  return(wishart)
}

# Method 2
wishart_method2 <- function(n, covMtx){
  if(is.matrix(covMtx)==F){
    break
  }
  A <- matrix(c(sqrt(rchisq(1,n)), 0,rnorm(1),sqrt(rchisq(1,n-1))),
              ncol=2, byrow = TRUE) ##
  U <- chol(covMtx)# upper triangle

  wishart <- t(U) %*% A %*% t(A) %*% U # i.e., L A A^T L^T
  return(wishart)
}
```

Run time test:

To compare the performance, both method were implemented for 10000 times.

```r
covMtx <- matrix(c(1, 0.5, 0.5, 2),ncol=2, byrow = T)
cat("Method 1 for random Wishart: \n")
system.time(for(i in 1:10000){wishart_method1(100, covMtx = covMtx)})
cat("\n Method 2 for random Wishart: \n")
system.time(for(i in 1:10000){wishart_method2(100, covMtx = covMtx)})
```

```
## Method 1 for random Wishart:
##    user  system elapsed
##    0.30    0.06    0.35
##
##  Method 2 for random Wishart:
##    user  system elapsed
##    0.20    0.05    0.23
```

One can see Method 2 is more efficient.

"'