904971914_stats102c_hw2

Xiaoshu Luo

2020/10/26

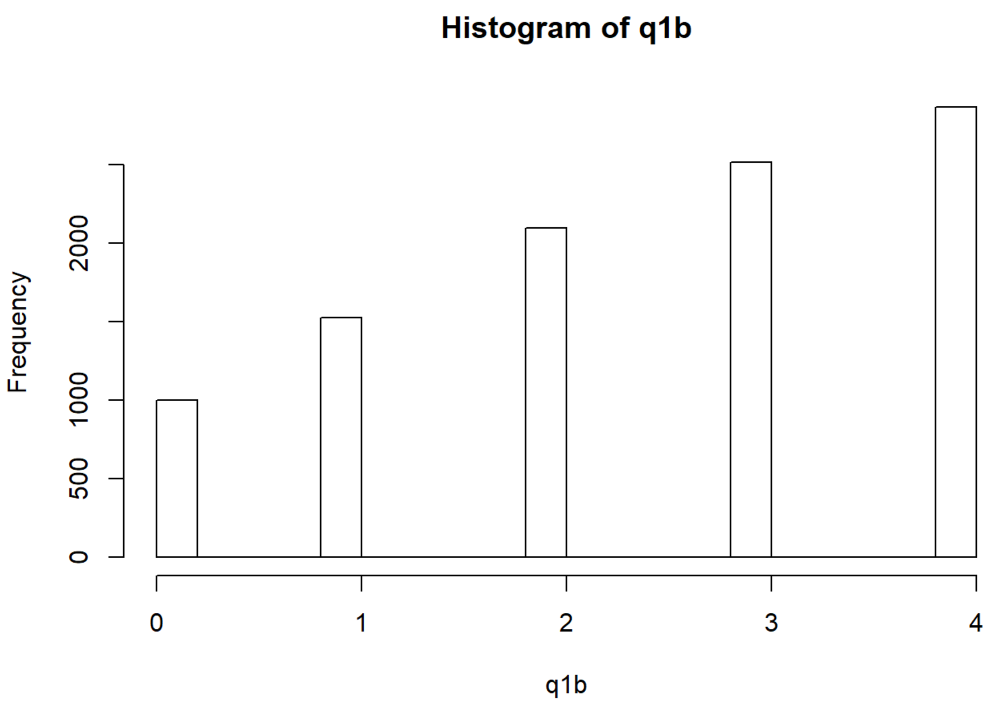
Question 1

(a)

```
set.seed(904971914)
q1a<-function(){
  #This function generates one value from the distribution X in problem 1
  #it has no arguments
  #it returns this value

  u<-runif(1)

  if(u<=0.1){
    return (0);
  }
  else if(u<=0.25){
    return (1);
  }
  else if (u<=0.45){
    return (2);
  }
  else if (u<=0.7){
    (3);
  }
  else {
    return (4);
  }
}
```

(b)

```
q1b<-c()
for(i in 1:10000){
  q1b<-c(q1b,q1a())
}
hist(q1b,freq=T)
```



(c)

```
sum(q1b==0)/10000
```

```
## [1] 0.0999
```

```
sum(q1b==1)/10000
```

```
## [1] 0.1524
```

```
sum(q1b==2)/10000
```

```
## [1] 0.2098
```

```
sum(q1b==3)/10000
```

```
## [1] 0.2515
```

```
sum(q1b==4)/10000
```

```
## [1] 0.2864
```

The sample relative frequencies are very close to the theoretical probabilities.

Question 2

(a)

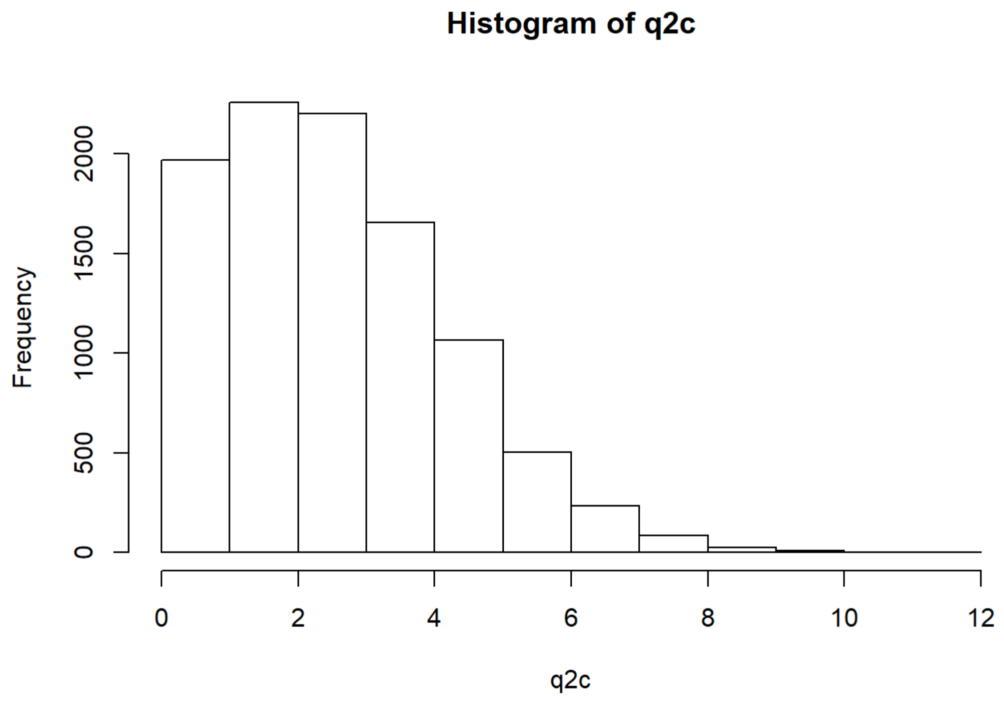
Pseudocode: 1 Generate u from uniform(0,1)
2 while loop:
if u >= F(x)
increment x by 1
generate p(x) from poisson
increment F(X) by adding p(x)
3. Stop and return x

(b)

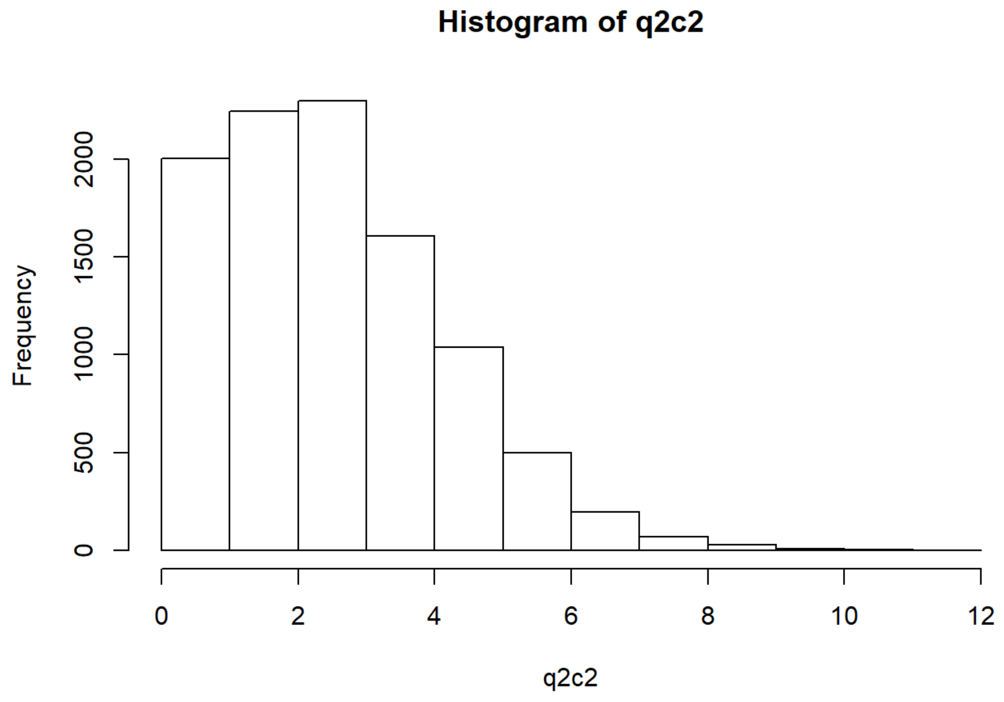
```
q2b<-function(lambda){
  #This function generates a value from a poisson distribution with lambda
  #the argument lambda gives the value of the parameter
  #the function returns the value generated
  u<-runif(1)
  x<-0
  cdf<-((lambda^x)*exp(-lambda))/factorial(x)
  while(u >=cdf){
    x<-x+1
    px<-((lambda^x)*exp(-lambda))/factorial(x)
    cdf<-cdf+px
  }
  return (x)



}
```

(c)

```
q2c<-c()
for(i in 1:10000){
  q2c<-c(q2c,q2b(3))
}

q2c2<-rpois(n=10000,lambda=3)
hist(q2c)
```




```
hist(q2c2)
```



The results from the two functions are very similar in terms of distribution.

Question 3

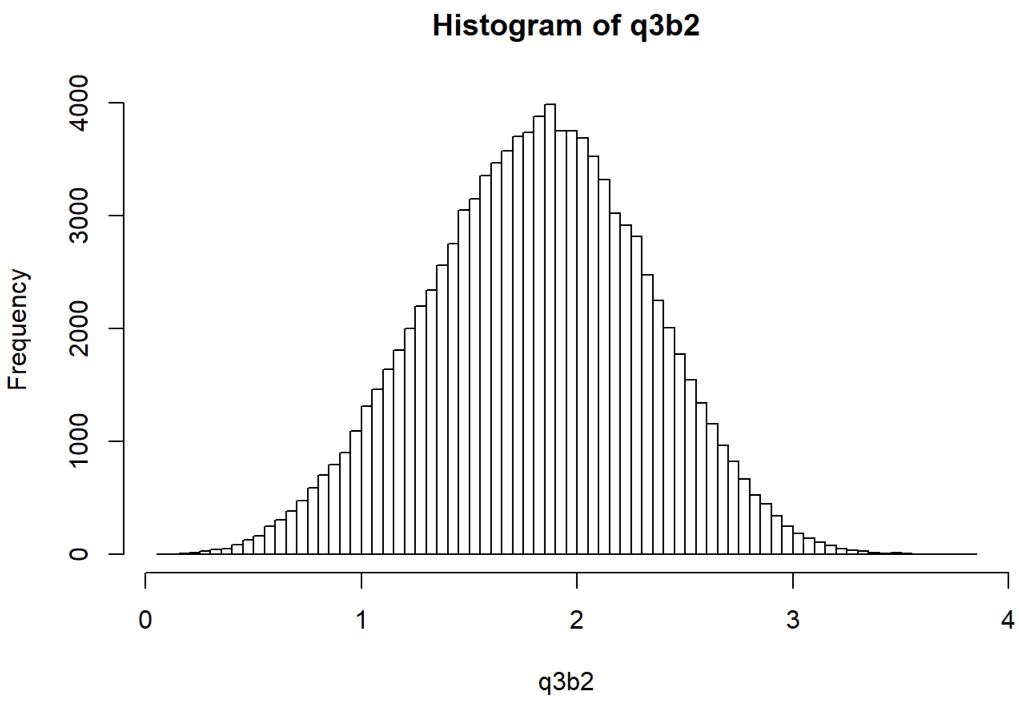
(a)

$$
\begin{aligned}
Y &= \left(\frac{X}{\alpha}\right)^{\beta} \\
P(Y \le y) &= P\left(\left(\frac{X}{\alpha}\right)^{\beta} \le y\right) \\
&= P\left(X \le \alpha y^{\frac{1}{\beta}}\right) \\
&= F_X\left(\alpha y^{\frac{1}{\beta}}\right) \\
&= 1 - e^{-\left(\frac{\alpha y^{\frac{1}{\beta}}}{\alpha}\right)^{\beta}} \\
&= 1 - e^{-y} \sim exp(1)
\end{aligned}
$$

So it follows an exponential distribution with lambda = 1.

(b)

```
q3b<-function (n){
  #This function generates n values from distribution of X in problem 3.
  #It takes one argument n which specifies the number of values to e generated.
  #It returns the values
  raw <-rexp(n)
  y <-(raw^(1/4))*2
  y


}

q3b2 <-q3b(100000)
hist(q3b2,breaks=100)
```



Question 4

Assume that g(x) is the envelope function and n is total number of points.

$$N = Total\ number\ of\ points\ in(x, x+\Delta x) = n \times g(x) \times \Delta x$$

$$p = Acceptance\ probability\ in(x, x+\Delta x) = \frac{f(x)}{Mg(x)}$$

$$Number\ of\ points\ remained\ in(x, x+\Delta x) = N \times p = \frac{n}{M} f(x)\Delta x$$

$$total\ points\ remained = \sum_{bin\ number} \frac{n}{M} f(x)\Delta x = \frac{n}{M}\sum_{bin\ number} f(x)\Delta x = \frac{n}{M}$$

$$P(x \in (x, x+\Delta x)) = \frac{\frac{n}{M} f(x)\Delta x}{\frac{n}{M}} = f(x)\Delta x$$

$$f(x) = \frac{P(x \in (x, x+\Delta x))}{\Delta x}$$

Question 5

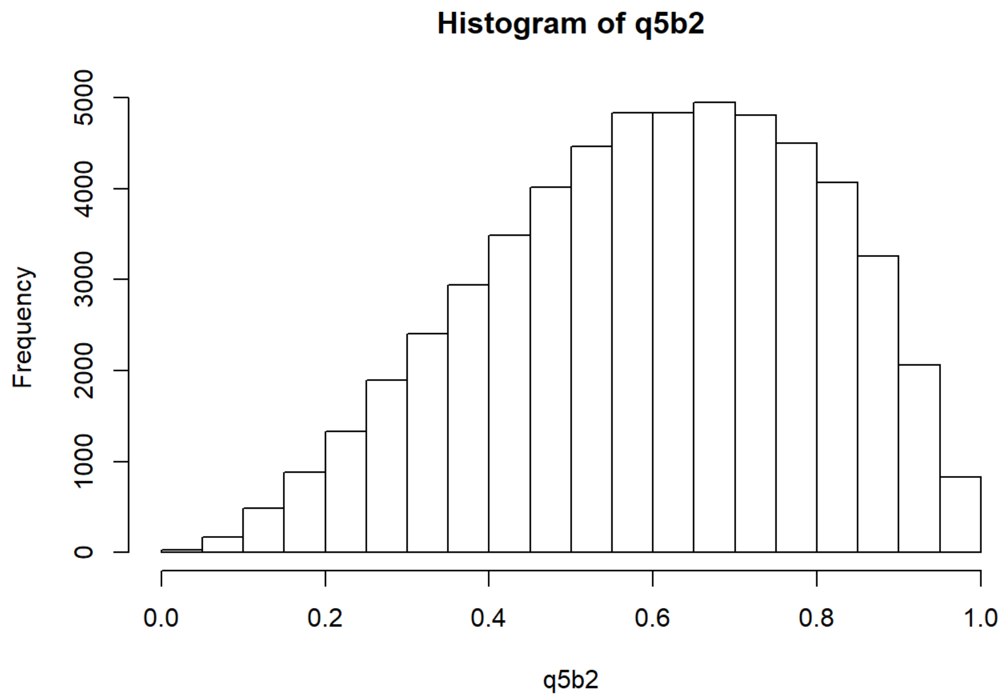
(a) Take g(x) as Unif(0,1), f(x) as Beta(3,2) [

$$x \in [0, 1]$$
$$g(x) = 1$$
$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\frac{2!*1!}{4!}}$$
$$f(x) = 12 \times x^2 \times (1-x)$$
$$f'(x) = 0$$
$$x = 0 \ \ or \ \ x = \frac{2}{3}$$
$$M = max(\frac{f(x)}{g(x)}) = \frac{f(\frac{2}{3})}{1} = \frac{16}{9}$$

]

(b)

```
my_beta<-function(x,a,b){
  #this function gives the pdf of a beta distribution
  #it takes in x as the subject, a and b as parameters
  #it returns the pdf at x
  y <-(x^(a-1))*((1-x)^(b-1))
  z<-(factorial(a-1))*(factorial(b-1))/factorial(a+b-1)
  y/z
}

q5b<-function(n){
  #This function generates a distribution from n sample points using the acceptance-rejection method
  #it takes in one argument n as the number of sample points
  #it returns a vector of accepted x
  M<-16/9
  result<-c()

for(i in 1:n){

x<-runif(1)
U<-runif(1)
if (U < my_beta(x,3,2)/M){
  result <-c(result,x)
}

}

  result
}
q5b2<-q5b(100000)
hist(q5b2)
```



(c)

```
length(q5b2)/100000
```

```
## [1] 0.56225
```

```
1/ (16/9)
```

```
## [1] 0.5625
```

The acceptance rate is very close to 1/M.

Question 6

(a)

[

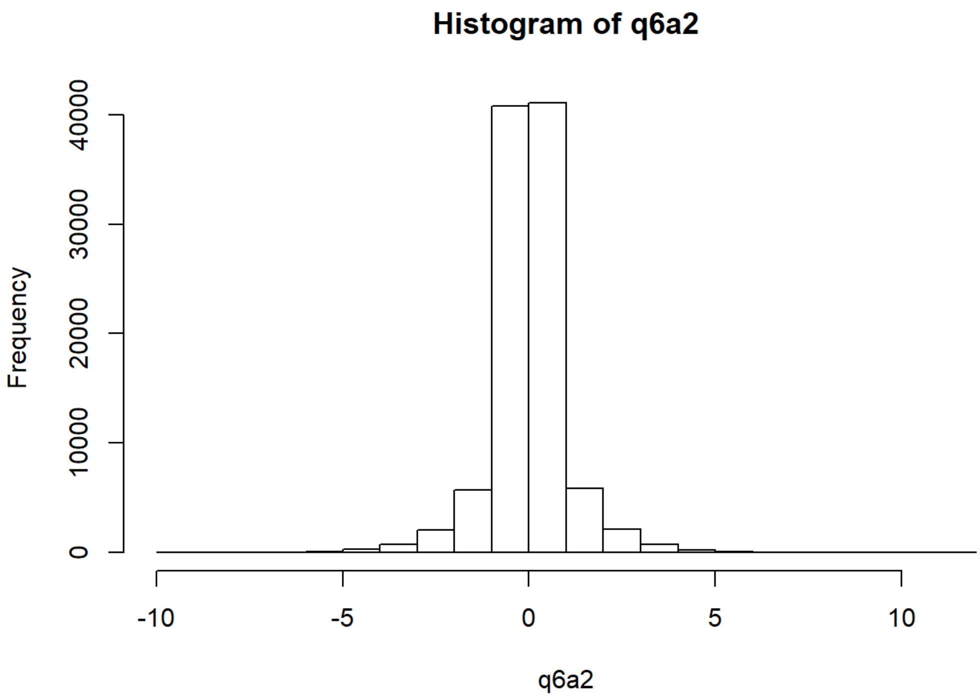
$$
\begin{aligned}
f(x) &= \frac{1}{2}e^{-|x|} \\
F(x) &= \int_{-\infty}^{x} f(x)dx \\
&= \int_{-\infty}^{x} \frac{1}{2}e^{-|x|}dx \\
if(x < 0), f(x) &= \frac{1}{2}e^{x} \\
if(x >= 0), f(x) &= \frac{1}{2}e^{-x} \\
if(x < 0), F(x) &= \frac{1}{2}e^{x} \\
if(x >= 0), F(x) &= 1 - \frac{1}{2}e^{-x} \\
if(x < 0), F^{-1}(u) &= ln(2u) \\
if(x >= 0), F^{-1}(u) &= -ln(2(1-u)) = -ln(2u), as\ u \sim Unif(0,1), 1-u \sim Unif(0,1)
\end{aligned}
$$

]

```r
q6a<-function(n){
  #This function generates n data points from the laplace distribution
  #it takes in one argument n as the number of samples to be generated
  #it returns a vector containing all the samples.
  u1 <-runif(n/2)
  result <- -log(2*u1)
  u2 <-runif(n/2)
  result<-c(result,log(2*u2))
  result

}
q6a2<-q6a(100000)
system.time(q6a(100000))
```

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

```r
hist(q6a2)
```



(b) Take g(x) as Norm(0,3).

[

$$max(\frac{f(x)}{g(x)}) = max(\frac{\frac{1}{2}e^{-|x|}}{\frac{1}{\sqrt{3}*\sqrt{2\pi}} * e^{-0.5\frac{x^2}{3}}})$$

$$if(x > 0), (\frac{f(x)}{g(x)})' = \frac{(\frac{x}{3} - 1) * e^{-x}}{e^{-\frac{x^2}{6}}}, take\ x = 3$$

$$if(x <= 0), (\frac{f(x)}{g(x)})' = \frac{(\frac{x}{3} + 1) * e^{-x}}{e^{-\frac{x^2}{6}}}, take\ x = -3$$

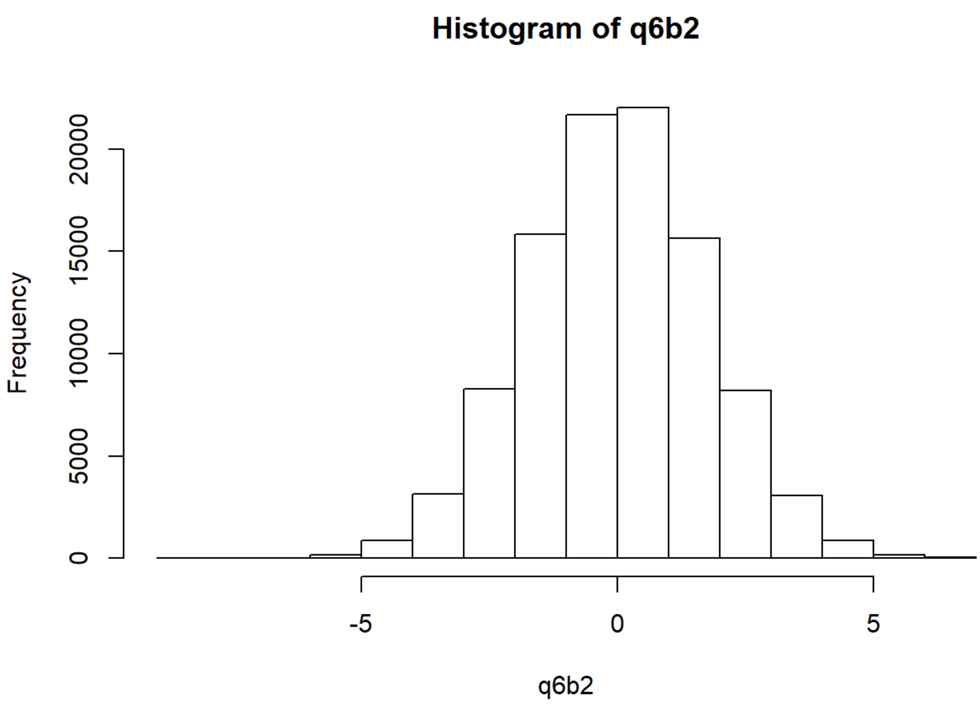
$$M = max(\frac{f(x)}{g(x)}) = max(\frac{f(3)}{g(3)}) = 0.4844$$

]

```r
q6b<-function(n){
  #This function generates n data points from the laplace distribution using acceptance-rejection method.
  #it takes in one argument n as the number of samples to be generated
  #it returns a vector containing all the samples accepted.
  M<-0.4844
  result<-c()

for(i in 1:n){
x<-rnorm(1,mean=0,sd=sqrt(3))
f<-0.5*exp(-abs(x))
g<-dnorm(x,mean=0,sd=sqrt(3))
U<-runif(1)
if (U < f/(g*M)){
  result <-c(result,x)
}


}
 result
}


q6b2<-q6b(100000)
system.time(q6b(100000))
```

```
##    user  system elapsed
##   11.26    0.06   11.60
```

```r
hist(q6b2)
```



(c) For inverse CDF, it is computationally efficient and thus takes less time. However, sometimes an explicit inverse cdf may not be available.

For acceptance-rejection, it does not requuire an explicit inverse cdf. However, it is sometimes hard to find an appropriate g(x) and M. Also, [illegible] mpuational cost than inverse CDF and thus takes more time.