Homework 5

Rebecca C. Steorts

February 21, 2018

1.

```
h = function(x){
  y= exp(-x^4)
  return(y)
}
set.seed(1)
mySample = 1000
x = rnorm(mySample)
m = mean(h(x)/dnorm(x))
se = sd(h(x)/dnorm(x))/sqrt(mySample)
mySample = 100000
x = rnorm(mySample)
m2 = mean(h(x)/dnorm(x))
se2 = sd(h(x)/dnorm(x))/sqrt(mySample)
mySample = 1000
x = runif(mySample, min = -6, max = 6)
m3 = mean(h(x)/dunif(x, min = -6, max = 6))
se3 = sd(h(x)/dunif(x, -6, 6))/sqrt(mySample)
mySample = 100000
x = runif(mySample, min = -6, max = 6)
m4 = mean(h(x)/dunif(x, min = -6, max = 6))
se4 = sd(h(x)/dunif(x, -6, 6))/sqrt(mySample)
```

When the number of random samples for generating the classical Monte Carlo estimate is smaller (n=1000), one finds an estimate for the solution of this integral to be 1.7854208 with a standard error of 0.0330458, using a Normal(0,1) distribution to generate the sample. When one generates a larger number of random samples (n= 100,000), the estimate for the solution to this integral is similar, 1.8090321, however the standard error is much smaller, namely, 0.0032588.

When, one uses a Uniform(-6,6) distribution to generate a sample, with a smaller number of samples (n= 1000), the estimate to the solution to the integral is 1.7501799 with a standard error of 0.1226675. Using a larger number of random samples (n=100000) results in a similar estimate when using the Uniform(-6,6) distribution to generate the sample, at 1.8143103, but with a lower standard error of 0.0122671.

One problem with the aforementioned procedures is that there is a greater standard error when using smaller sample sizes, indicating that estimated solution is less exact or precise. Another issue is that the standard errors are different when using different procedures, with a Normal(0,1) distribution for samples versus a Uniform(-6,6) distribution for samples.

2.

Let

$$F(x \mid c, \beta) = \exp\{-e^{-(y-c)/\beta}\}$$

Let

$$x = \exp\{-e^{-(y-c)/\beta}$$

$$\ln(x) = -e^{-(y-c)/\beta} \tag{1}$$
$$-\ln(x) = e^{-(y-c)/\beta} \tag{2}$$
$$\ln(-\ln(x)) = -(y-c)/\beta \tag{3}$$
$$\beta \ln(-\ln(x)) = -(y-c) \tag{4}$$
$$y = c - \beta \ln(-\ln(x)) \tag{5}$$

Then

$$G(x) = c - \beta \ln(-\ln(x)).$$

This implies that

$$G(U) = c - \beta \ln(-\ln(U)),$$

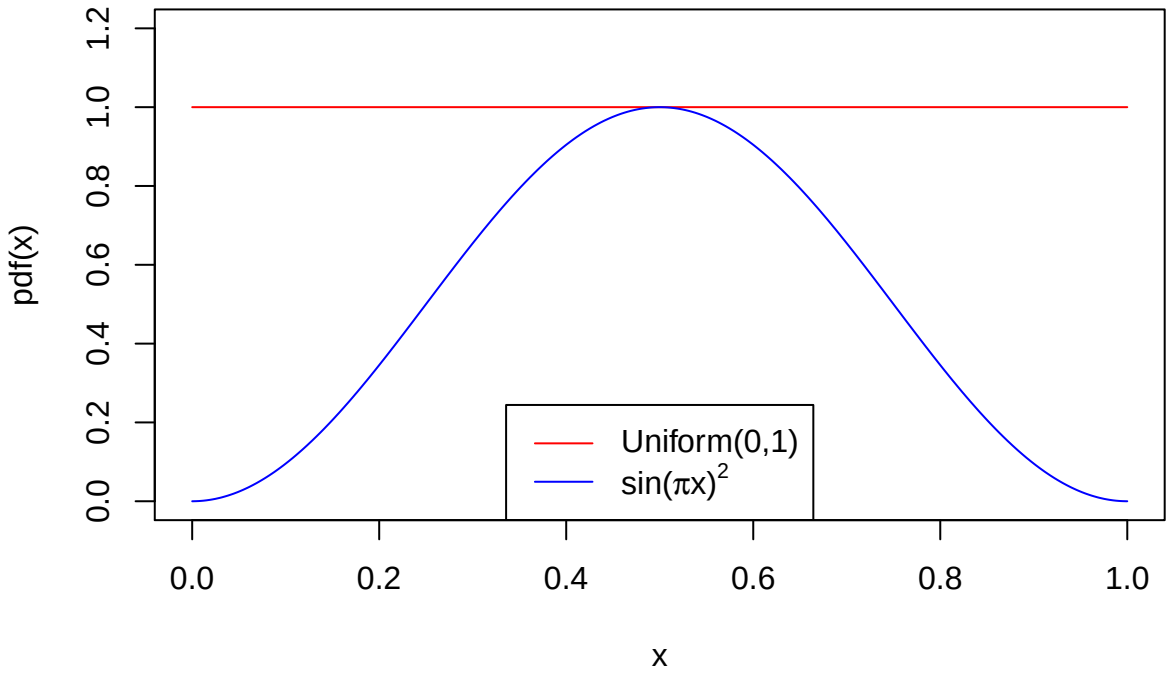
where $U \sim Unif(0,1)$.

Thus, if you generate uniform random variables and subsitute these random varaibles in above, you will generate a Gumbel(c, β) random variable.

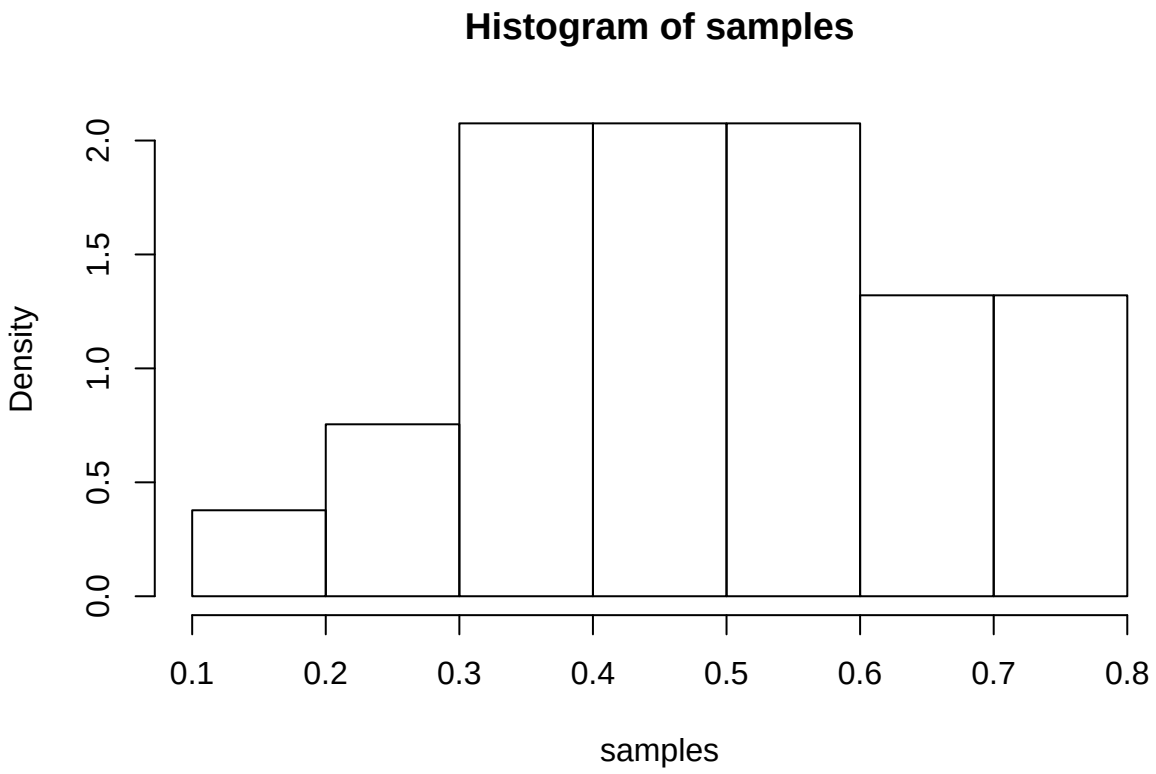
4.

Task 3

```
set.seed(123)
densFun = function(x) {
  return(sin(pi*x)^2)
}
x = seq(0, 1, length.out = 1000)
envelope = dunif(x)
f = densFun(x)
plot(x, envelope, type = "l", col = "red", xlab = "x",
     ylab = "pdf(x)", main = "Density Comparison", ylim = c(0,1.2))
lines(x, f, type = "l", col = "blue",
      xlab = "", ylab = "", ylim = c(0,1.2))
legend("bottom", legend = c("Uniform(0,1)",
                            expression(paste("sin(",pi,"x)"^"2"))),
       col = c("red", "blue"), lty = c(1,1))
```

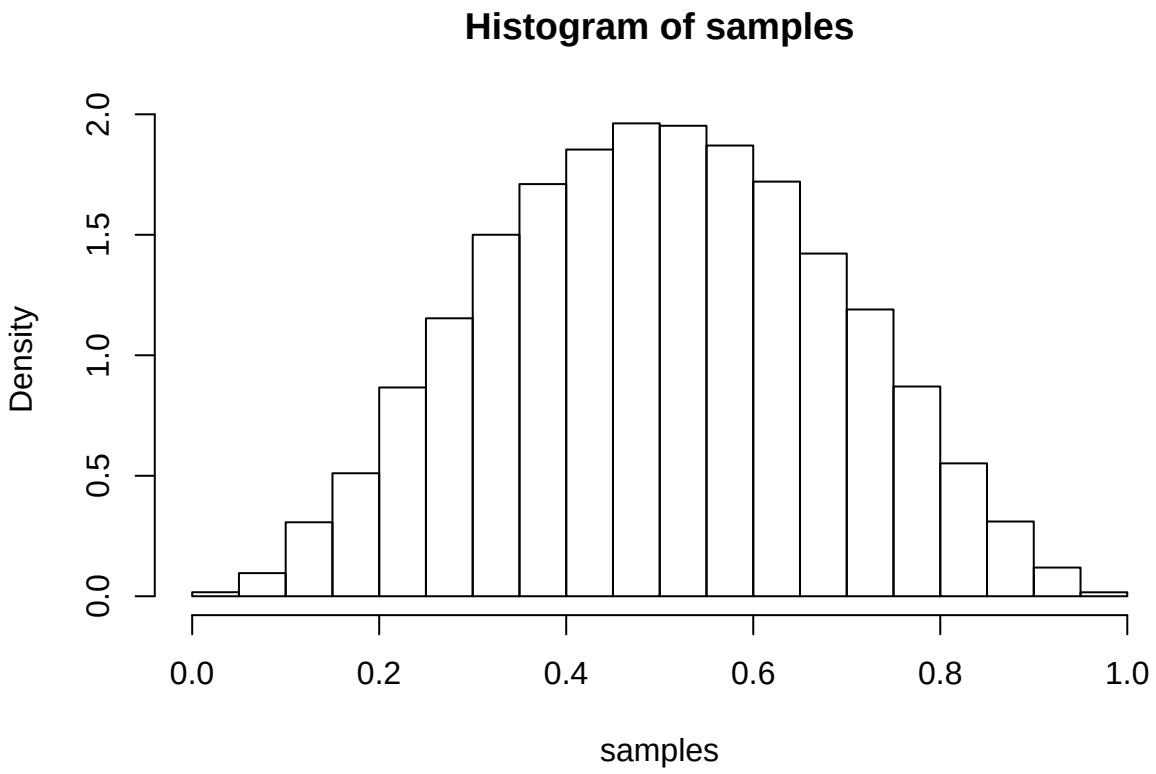



```
numSim=10^2
samples = NULL
for (i in 1:numSim) {
  #get a uniform proposal
  proposal = runif(1)
  #calculate the ratio
  densRat = densFun(proposal)/dunif(proposal)
  #accept the sample with p = densRat
  if (runif(1) <densRat) {
    #fill our vector with accepted samples
    samples = c(samples, proposal)
  }
}
hist(samples, freq = FALSE)
```




```
ar1 = length(samples)/numSim

numSim=10^5
samples = NULL
for (i in 1:numSim) {
  #get a uniform proposal
  proposal = runif(1)
  #calculate the ratio
  densRat = densFun(proposal)/dunif(proposal)
  #accept the sample with p = densRat
  if (runif(1) <densRat) {
    #fill our vector with accepted samples
    samples = c(samples, proposal)
  }
}
hist(samples, freq = FALSE)
```

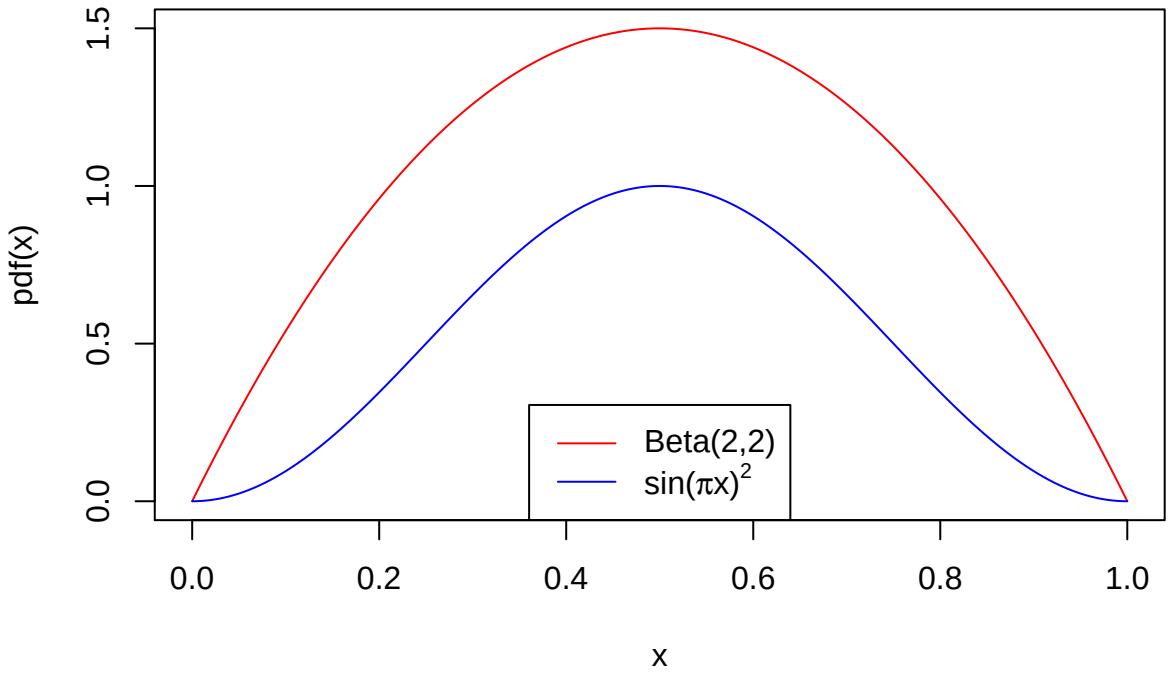



```
ar2 = length(samples)/numSim
```

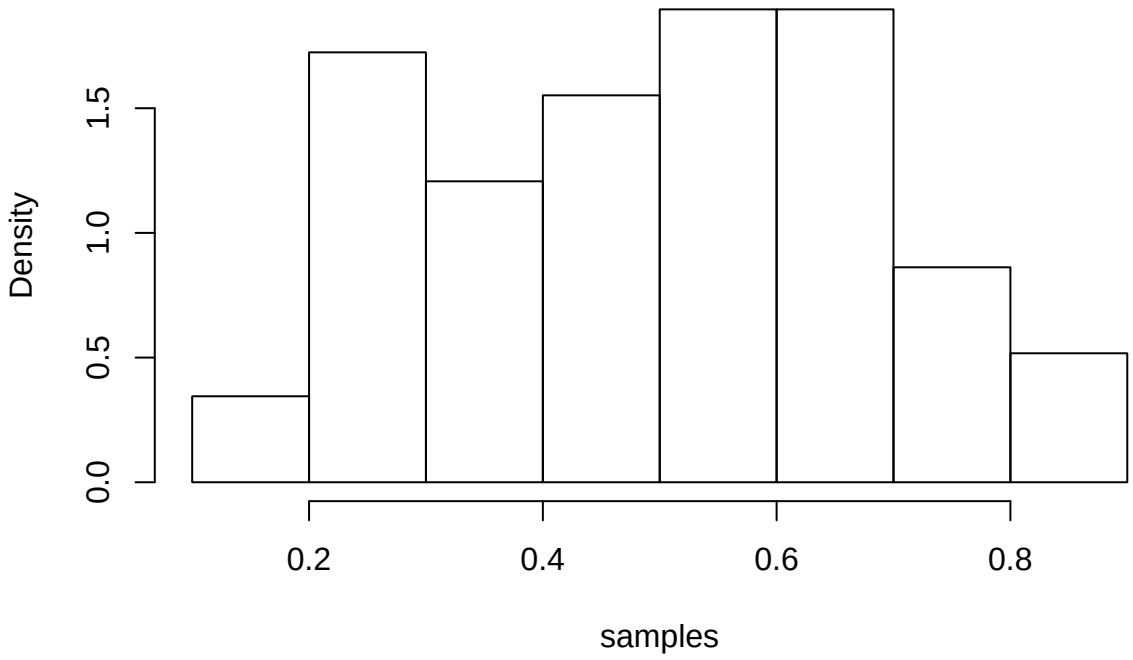
When comparing the output of this rejection sampling for 100 versus 100,000 simulations, I found that the acceptance ratio was different for the smaller number of simulations, at 0.53 for 100 simulations versus 0.50233 for 100,000 simulations. Since there were more simulations for the acceptance ratio estimate of 0.50233, this is probably closer to the true acceptance ratio for the Uniform(0,1) enveloping function. Comparing the histograms for 100 versus 100,000 simulations, the histogram for the 100,000 simulations looks closer in shape to the distribution of the kernel of the density function we are estimating, $sin^2(\pi x)$, than the histogram for 100 simulations.

Task 4

```
x = seq(0, 1, length.out = 1000)
envelope = dbeta(x, 2,2)
f = densFun(x)
plot(x, envelope, type = "l", col = "red", xlab = "x",
     ylab = "pdf(x)", main = "Density Comparison")
lines(x, f, type = "l", col = "blue",
      xlab = "", ylab = "")
legend("bottom", legend = c("Beta(2,2)",
                            expression(paste("sin(",pi,"x)"^"2"))),
       col = c("red", "blue"), lty = c(1,1))
```

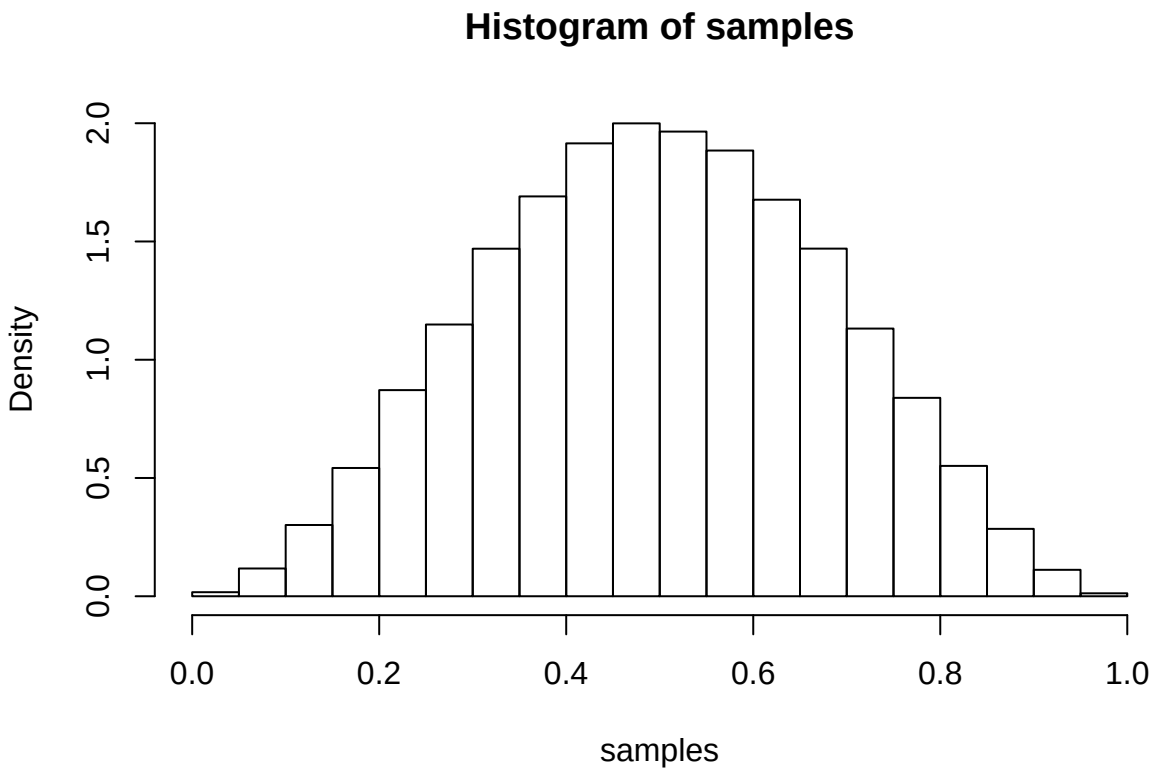



```
set.seed(123)
numSim=10^2
samples = NULL
for (i in 1:numSim) {
  #get a uniform proposal
  proposal = rbeta(1, shape1 = 2, shape2 = 2)
  #calculate the ratio
  densRat = densFun(proposal)/dbeta(proposal,
                                    shape1 = 2, shape2 = 2)
  #accept the sample with p = densRat
  if (runif(1) <densRat) {
    #fill our vector with accepted samples
    samples = c(samples, proposal)
  }
}
hist(samples, freq = FALSE)
```




```
ar3 = length(samples)/numSim

numSim=10^5
samples = NULL
for (i in 1:numSim) {
  #get a uniform proposal
  proposal = rbeta(1, shape1 = 2, shape2 = 2)
  #calculate the ratio
  densRat = densFun(proposal)/dbeta(proposal,
                                    shape1 = 2, shape2 = 2)
  #accept the sample with p = densRat
  if (runif(1) <densRat) {
    #fill our vector with accepted samples
    samples = c(samples, proposal)
  }
}
hist(samples, freq = FALSE)
```




```
ar4 = length(samples)/numSim
```

For the enveloping function of Beta(2,2), the acceptance ratio for 100 simulations is 0.58, different than the acceptance ratio for 100,000 simulations, 0.50113. Again, because of the higher number of simulations, the acceptance ratio for 100,000 simulations, 0.50113, is probably closer to the truth for this enveloping function. The histogram for 100,000 simulations is closer in shape to the distribution of the kernel of the density function we are estimating, $sin^2(\pi x)$, than the histogram for 100 simulations.

Task 5

Since the Beta(2,2) enveloping function more closely follows the kernel of the distribution we are estimating than the Uniform(0,1) enveloping function, in theory the Beta(2,2) should produce more accurate estimations of the normalized density of the function f. This is because around 0.5 for the Uniform(0,1) enveloping function, almost 100% of the proposals will be accepted while around 0 or 1, almost none of the proposals will be accepted, due to the nature of the space between the enveloping Uniform(0,1) function and the density function $sin^2(\pi x)$. However, in practice, the histograms of the samples look approximately the same for the higher number of simulations, 100000, for the Uniform(0,1) and Beta(2,2) enveloping functions, indicating that they have similar accuracies at this number of simulations. For lower numbers of simulations, the histograms for both enveloping functions are essentially random and do not reflect the true distribution of the density function. The acceptance ratios for the respective number of simulations for the enveloping functions are also similar, with 0.53 and 0.58 for the Uniform(0,1) and Beta(2,2), respectively, for the 100 simulations, and 0.50233 and 0.50113 for the Uniform(0,1) and the Beta(2,2), respectively, for the 100000 simulations. At lower numbers of simulations the acceptance ratios for the two enveloping functions are similar but farther apart than at higher numbers of simulations, because at higher number of simulations, both procedures are more accurate. Therefore, at higher numbers of simulations, both procedures are essentially the same even with different enveloping functions.