Lab 5

Andrea Aveni

2023-02-06

Rejection sampling

The problem: We want to sample from a distribution on $\mathbb{R}$ given either:

1. its probability density function $p(\theta)$; or
2. some function $f(\theta) \propto p(\theta)$.

For instance, in a Bayesian framework, we want to sample from the posterior $p(\theta \mid x) \propto p(x \mid \theta)p(\theta)$.

- Rejection sampling is *one* way to address this.

The area under the graph of a function f is the set of points (x, y) such that $0 \leq y \leq f(x)$.

Fundamental lemma of rejection sampling: Let f be a positive and integrable function. If (X, Y) is uniformly distributed under the graph of f, then the marginal probability density function of X is proportional to f.

Rejection sampling Algorithm

Input:

- An integrable function $f \geq 0$ and an enveloppe $g \geq f$ which you can sample from.

Output:

- A sample X distributed following the density proportional to f.

Algorithm:

1. Sample $X \sim g$ and $Y \mid X \sim \text{unif}(0, g(X))$.
2. If $Y < f(X)$, then return X. Otherwise go back to step 1.

Rejection sampling

Note:

- The function g is called the *envelop* function, and the corresponding distribution is the *proposal* distribution, or the *instrumental* distribution.
- The function f is called the *target*.

Lab 5

```
library(ggplot2)
library(tictoc)
```

We want to sample from a random variable with the density

$$f(x) = 2\sin^2(\pi x)\mathbf{1}_{[0,1]}(x),$$

using rejection sampling. (PS can you verify that this is a density?)

We start considering the proposal $\text{Unif}(0,1) = \text{Beta}(1,1)$, so that its density is $g(x) = \mathbf{1}_{[0,1]}(x)$.

Before doing anything else we rescale f so that αf is always below the proposal g. In order to find the right constant, we need to solve the problem

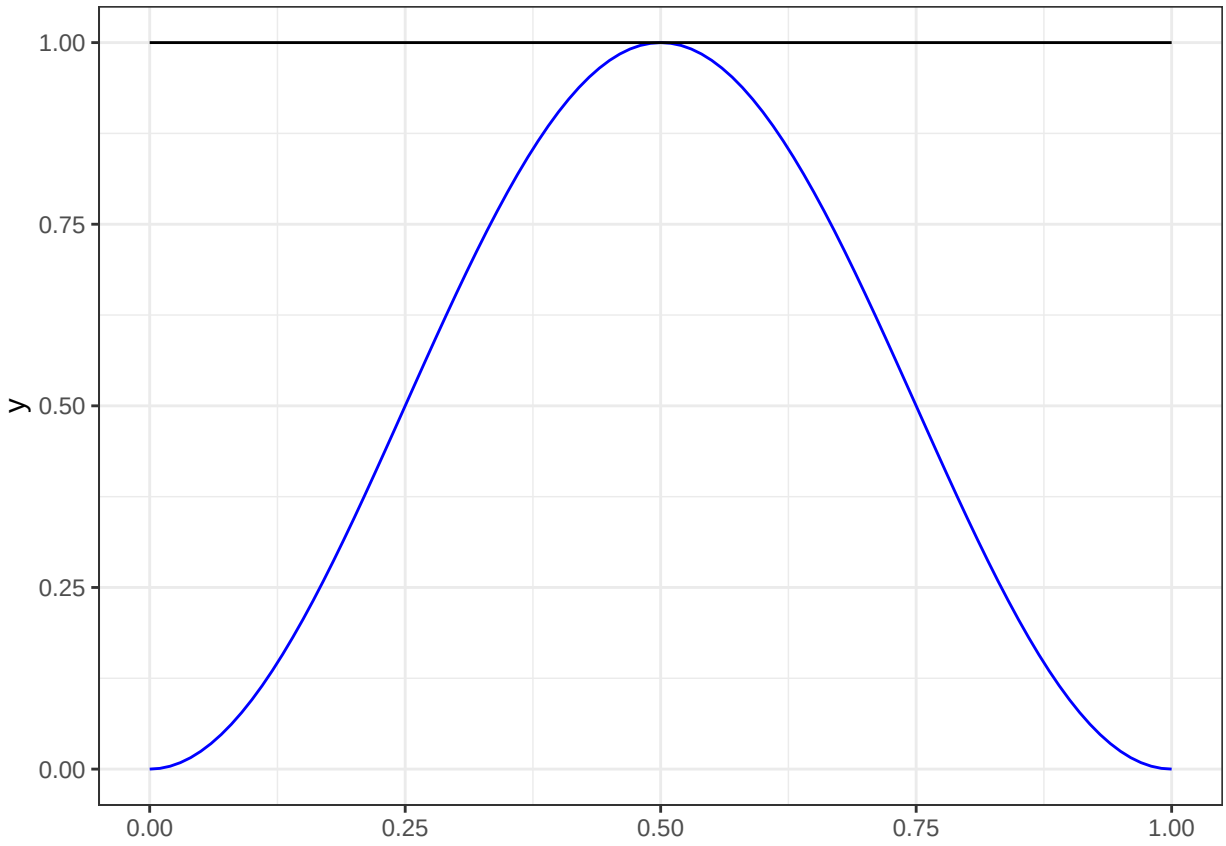
$$\forall x \in [0,1],\ \alpha f(x) \le g(x), \quad \Rightarrow \quad \alpha \le \min_{x\in[0,1]} \frac{g(x)}{f(x)}$$

And since a larger alpha will increase the acceptance rate, we will take

$$\alpha = \min_{x\in[0,1]} \frac{g(x)}{f(x)}$$

```
f=function(x){ return(2*sin(pi*x)^2) }
g1=function(x){ return(dunif(x,0,1)) }
rg1=function(n){ return(rbeta(n,1,1)) }

N=10^5
xx=(1:(N-1))/N
a1=min(g1(xx)/f(xx))

f1=function(x){ return(a1*f(x)) }
```

```
ggplot()+
  geom_function(fun=f1,col="blue")+
  geom_function(fun=g1,col="black")+
  xlim(0,1)+theme_bw()
```



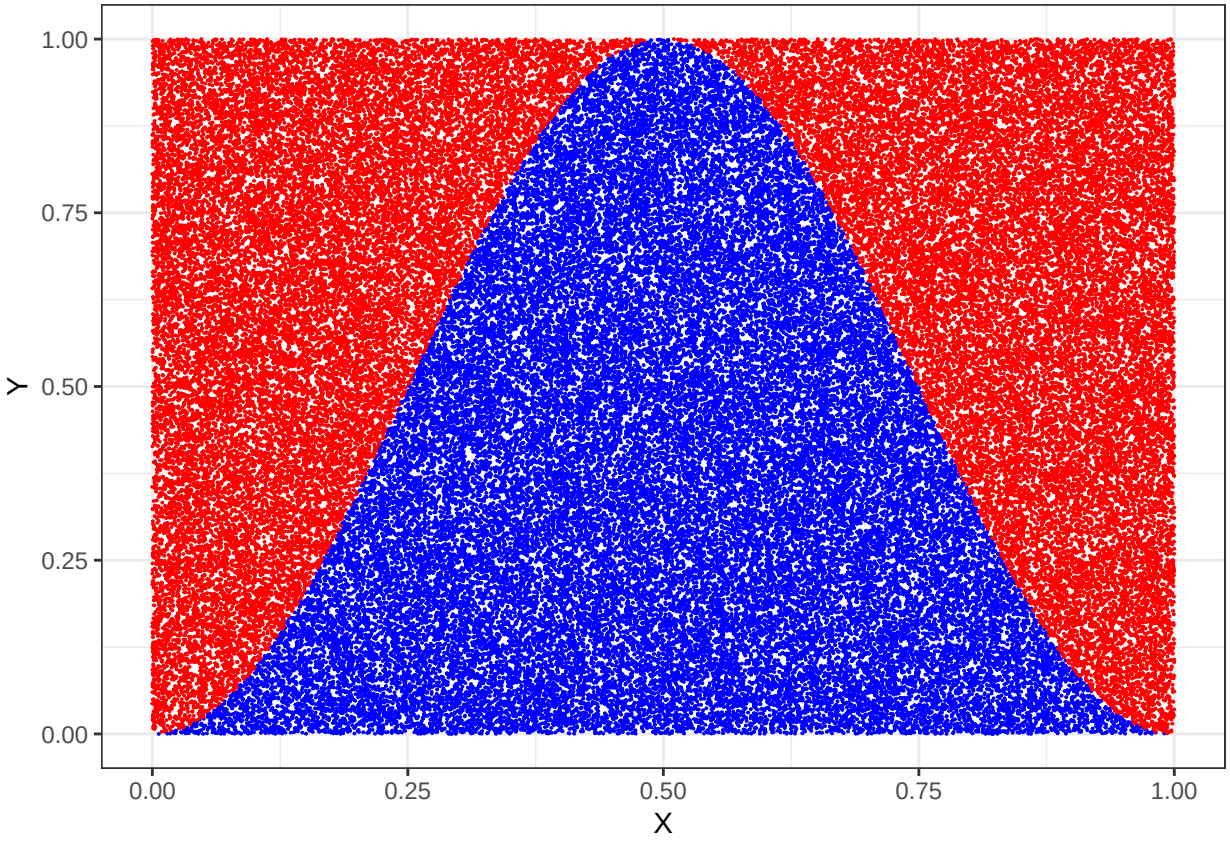
We now illustrate the algorithm

```
N=10^5
tic()
Xaccepted=Xrejected=Yaccepted=Yrejected=c()
for(j in 1:N){

  x=rg1(1)
  y=runif(1,min=0,max=g1(x))

  if(y<f1(x)){
    Xaccepted=c(Xaccepted,x)
    Yaccepted=c(Yaccepted,y)
  }
  else{
    Xrejected=c(Xrejected,x)
    Yrejected=c(Yrejected,y)
  }

}
fortime=toc()
```

```
## 29.869 sec elapsed
```

```
ggplot()+
  geom_point(aes(x=Xaccepted,y=Yaccepted),col="blue",size=0.01)+
  geom_point(aes(x=Xrejected,y=Yrejected),col="red",size=0.01)+
  xlab("X")+ylab("Y")+
```

```
  theme_bw()
```

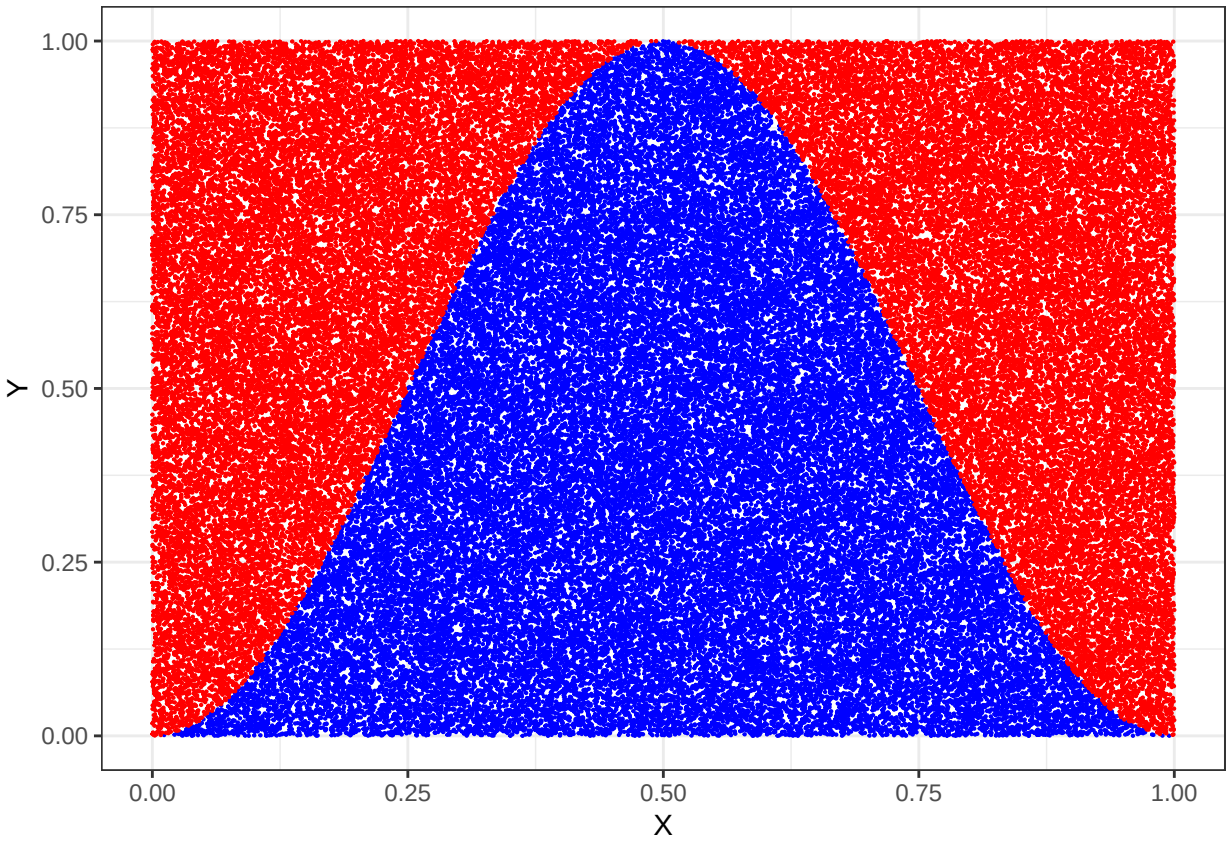


We can vectorize this for-loop and speed up computations in the following way

```
tic()
X = rg1(N)
Y = runif(N, min=0, max=g1(X))
acc=(Y<f1(X))
Xaccepted=X[acc]
Yaccepted=Y[acc]
Xrejected=X[!acc]
Yrejected=Y[!acc]
vectorizedtime=toc()
```

```
## 0.029 sec elapsed
```

```
ggplot()+
  geom_point(aes(x=Xaccepted,y=Yaccepted),col="blue",size=0.1)+
  geom_point(aes(x=Xrejected,y=Yrejected),col="red",size=0.1)+
  xlab("X")+ylab("Y")+
  theme_bw()
```



As you can see the result is the same but, by vectorizing, we reduced the time by a huge factor of

```
as.numeric((fortime$toc-fortime$tic)/(vectorizedtime$toc-vectorizedtime$tic))
```

```
## [1] 1029.966
```

We now provide a general function for rejection sampling

```
rej_sampl=function(N,f,g,rg){
  X = rg(N)
  Y = runif(N, min=0, max=g(X))
  acc=(Y<f(X))
  Xaccepted=X[acc]
  return(Xaccepted)
}

head(rej_sampl(1000,f1,g1,rg1))
```

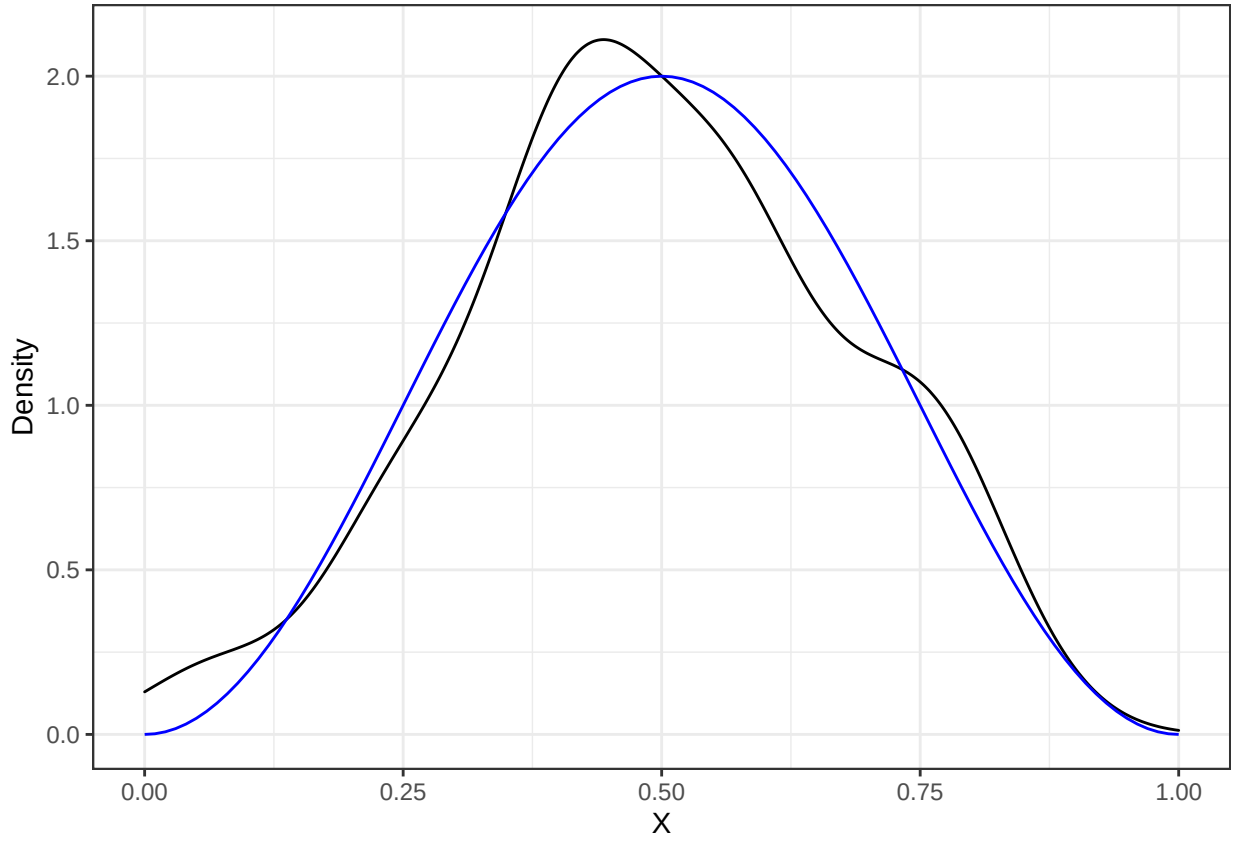
```
## [1] 0.4838756 0.3474047 0.5174293 0.3893938 0.5917014 0.3700326
```

We plot a histogram of the points that fall in the acceptance region for different simulation sizes and we report our acceptance ratio. We compare the ratios and histograms.

```
N=10^2
S=rej_sampl(N,f1,g1,rg1)

length(S)
```

```
## [1] 56
```

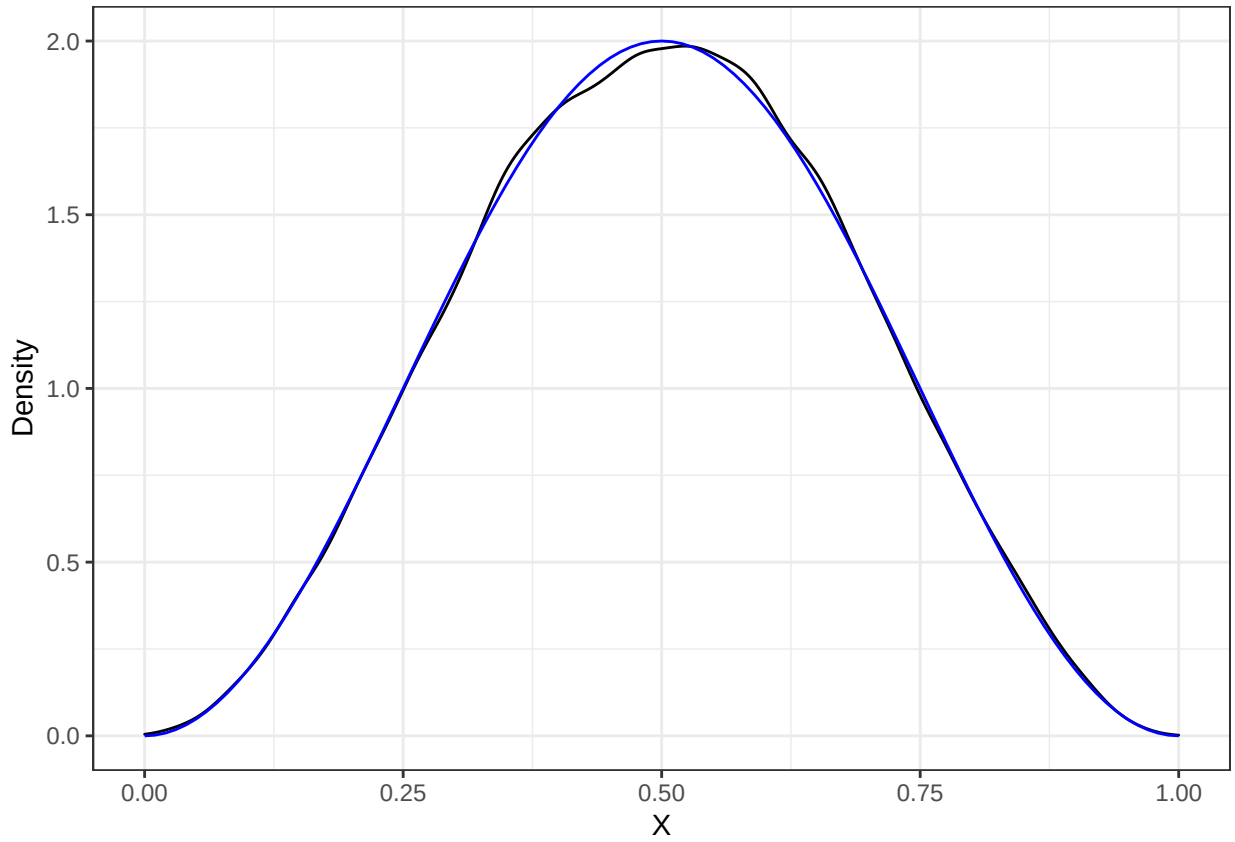
```
ggplot()+
  geom_density(aes(x=S))+
  geom_function(fun=f,col="blue")+
  xlim(0,1)+xlab("X")+ylab("Density")+
  theme_bw()
```




```
N=10^5
S=rej_sampl(N,f1,g1,rg1)

length(S)
```

```
## [1] 50300
```

```
ggplot()+
  geom_density(aes(x=S))+
  geom_function(fun=f,col="blue")+
  xlim(0,1)+xlab("X")+ylab("Density")+
  theme_bw()
```



Thus in this case the rejection sampling is 1/2. And this is not surprisingly, since the acceptance rate is equal to the area under the rescaled f. Thus

$$\text{rejection rate} = \int_0^1 \alpha f(x) \mathrm{d}x = \alpha$$

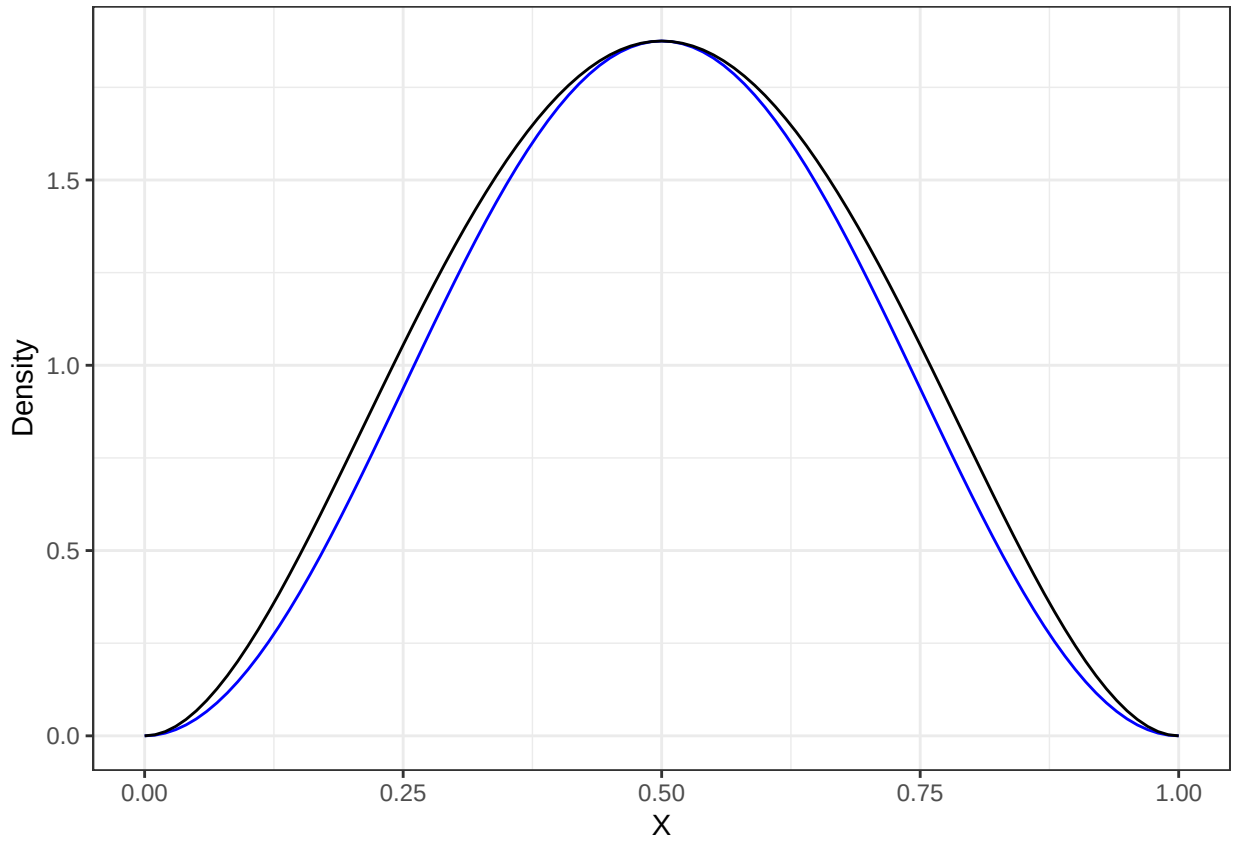
And, indeed, using the uniform proposal we got $\alpha = 1/2$.

Now we consider a different proposal that will lead to better rejection rates. We consider Beta$(3, 3)$ density.

```
g2=function(x){ return(dbeta(x,3,3)) }
rg2=function(n){ return(rbeta(n,3,3)) }

N=10^5
xx=(1:(N-1))/N
a2=min(g2(xx)/f(xx))

f2=function(x){ return(a2*f(x)) }
```

In this case the acceptance rate that we get is much higher (15/16), let's visualize this

```
ggplot()+
  geom_function(fun=f2,col="blue")+
  geom_function(fun=g2,col="black")+
  xlab("X")+ylab("Density")+xlim(0,1)+
  theme_bw()
```



And now let's proceed with the rejection sampling

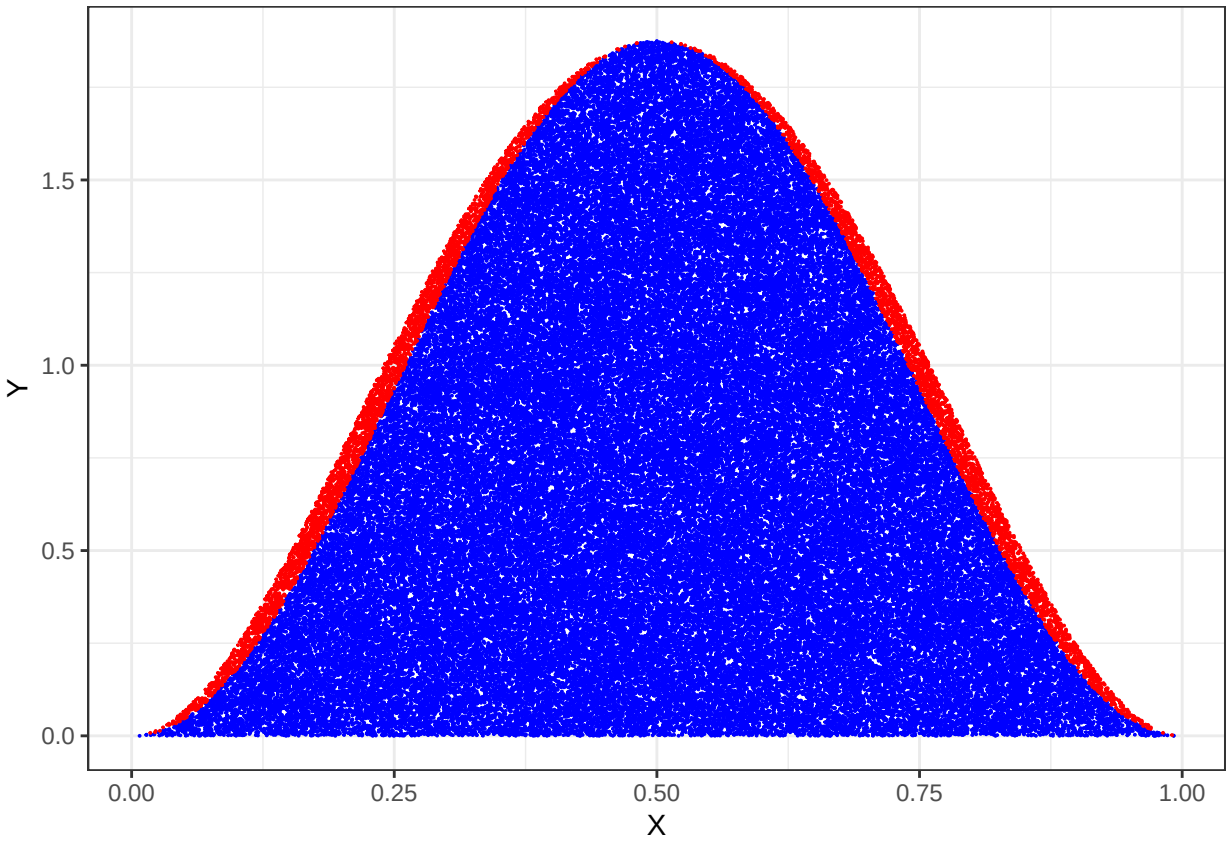
```
tic()
X = rg2(N)
Y = runif(N, min=0, max=g2(X))
acc=(Y<f2(X))
Xaccepted=X[acc]
Yaccepted=Y[acc]
Xrejected=X[!acc]
Yrejected=Y[!acc]
vectorizedtime=toc()
```

```
## 0.038 sec elapsed
```

```
sum(acc)/N
```

```
## [1] 0.93763
```

```
ggplot()+
  geom_point(aes(x=Xaccepted,y=Yaccepted),col="blue",size=0.01)+
  geom_point(aes(x=Xrejected,y=Yrejected),col="red",size=0.01)+
  xlab("X")+ylab("Y")+
  theme_bw()
```

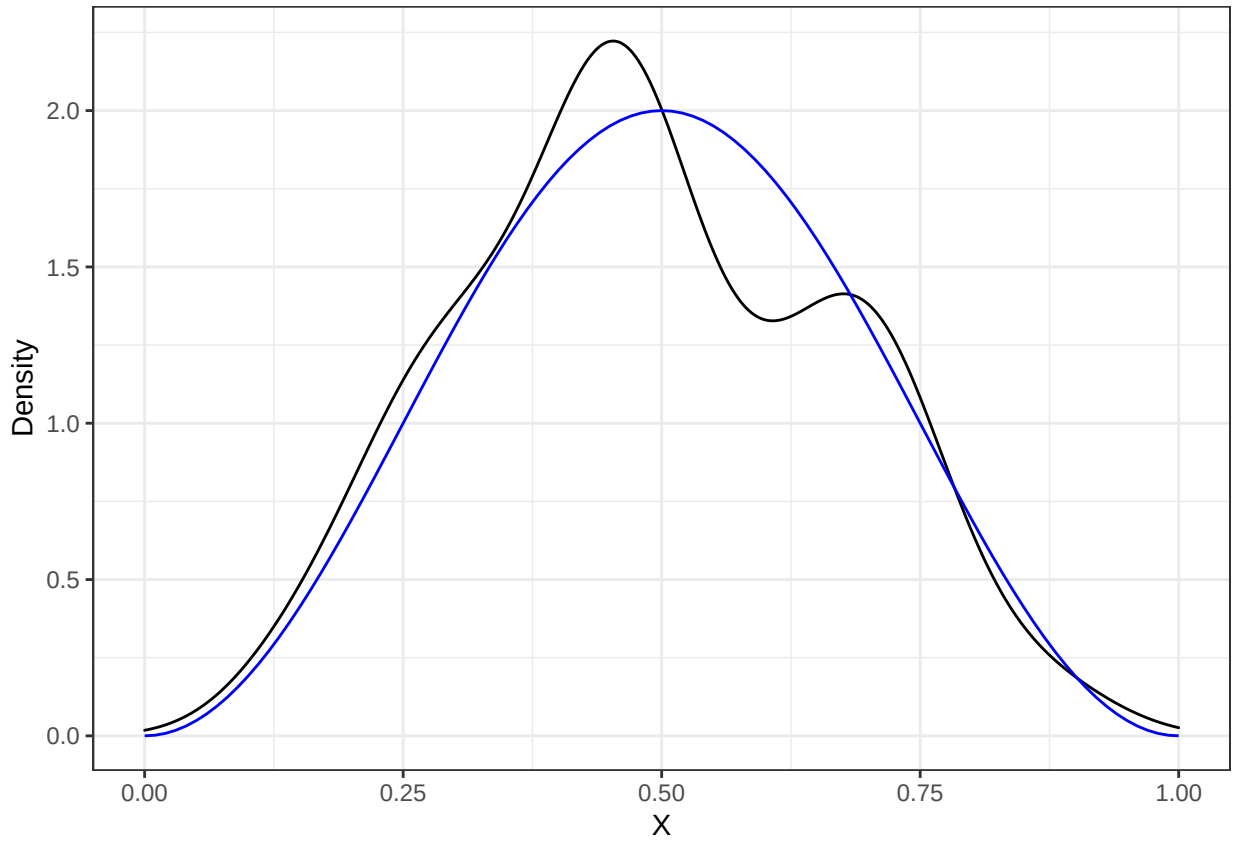


Plot a histogram of the points that fall in the acceptance region. Do this for a simulation size of 10^2 and 10^5 and report your acceptance ratio. Compare the ratios and histograms.

```
N=10^2
S=rej_sampl(N,f2,g2,rg2)

length(S)/N
```

```
## [1] 0.96
```

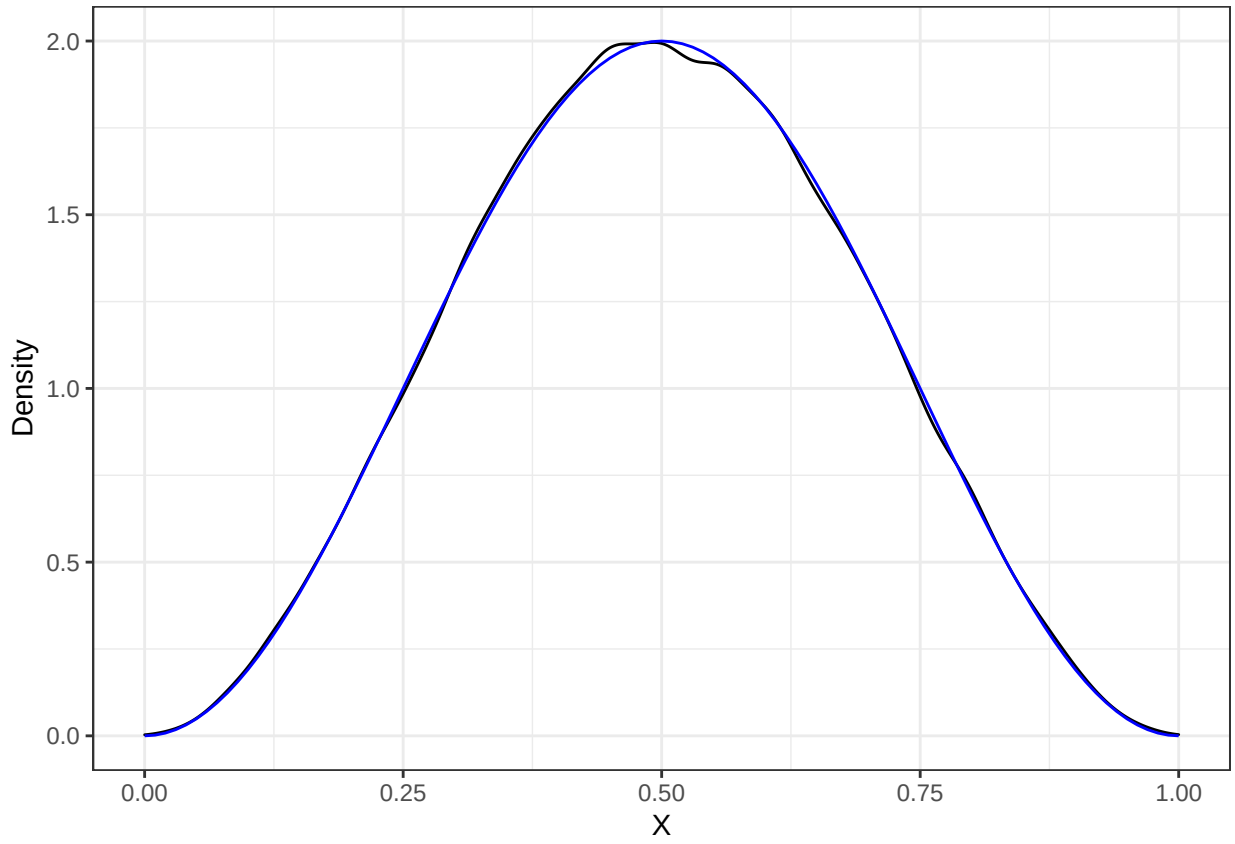
```
ggplot()+
  geom_density(aes(x=S))+
  geom_function(fun=f,col="blue")+
  xlim(0,1)+xlab("X")+ylab("Density")+
  theme_bw()
```




```
N=10^5
S=rej_sampl(N,f2,g2,rg2)

length(S)/N
```

```
## [1] 0.93637
```

```
ggplot()+
  geom_density(aes(x=S))+
  geom_function(fun=f,col="blue")+
  xlim(0,1)+xlab("X")+ylab("Density")+
  theme_bw()
```


2.0
1.5
1.0
0.5
0.0
Density
0.00
0.25
0.50
0.75
1.00
X