MATH 559: BAYESIAN THEORY AND METHODS

METROPOLIS-HASTINGS FOR CONTINUOUS STATE SPACES

We wish to sample from the distribution

$$\pi(x) \propto x^{\gamma-1} e^{-x-1/x} \qquad x > 0$$

using the Metropolis-Hastings algorithm. This density is not the $Gamma(\gamma, 1)$ density, although it is somewhat similar. The normalizing constant is not readily available analytically, but can be estimated using Monte Carlo by noting that

$$\int_0^\infty x^{\gamma-1} e^{-x-1/x}\, dx = \Gamma(\gamma) \int_0^\infty e^{-1/x} \frac{1}{\Gamma(\gamma)} x^{\gamma-1} e^{-x}\, dx = \Gamma(\gamma)\mathbb{E}[e^{-1/X}]$$

with the expectation taken with respect to $X \sim Gamma(\gamma, 1)$.

```
set.seed(4532)
N<-100000
gam<-2.5
(M<-replicate(10,mean(gamma(gam)*exp(-1/rgamma(N,gam,1)))))

+  [1] 0.7803060 0.7786459 0.7786613 0.7799752 0.7799418 0.7782698 0.7804620
+  [8] 0.7804127 0.7808335 0.7798685

(const=1/mean(M))

+ [1] 1.282483
```

To implement the algorithm we need to

1. pick a starting value x_0
2. for steps $t = 0, 1, 2, \ldots, N, \ldots$
   (i) propose a new value z from some transition density $q(x_t, z)$, some conditional density given x_t,
   (ii) set $x_{t+1} = z$ with probability
   $$\alpha(x_t, z) = \min\left\{1, \frac{\pi(z)q(z, x_t)}{\pi(x_t)q(x_t, z)}\right\}$$
   otherwise set $x_{t+1} = x_t$.

The selected transition density should leave the resulting chain irreducible, aperiodic and positive recurrent, so that $\pi(x)$ is the stationary distribution of the chain. It is sufficient to achieve a $q(x, z)$ that proposes values on the support of $\pi(x)$. In the algorithms below, the proposal that sets

$$z = |x + \delta| \qquad \delta \sim Normal(0, \sigma_\gamma^2)$$

for some $\sigma_\gamma > 0$ is used. With this choice it follows that $q(x, z) = q(z, x)$, and

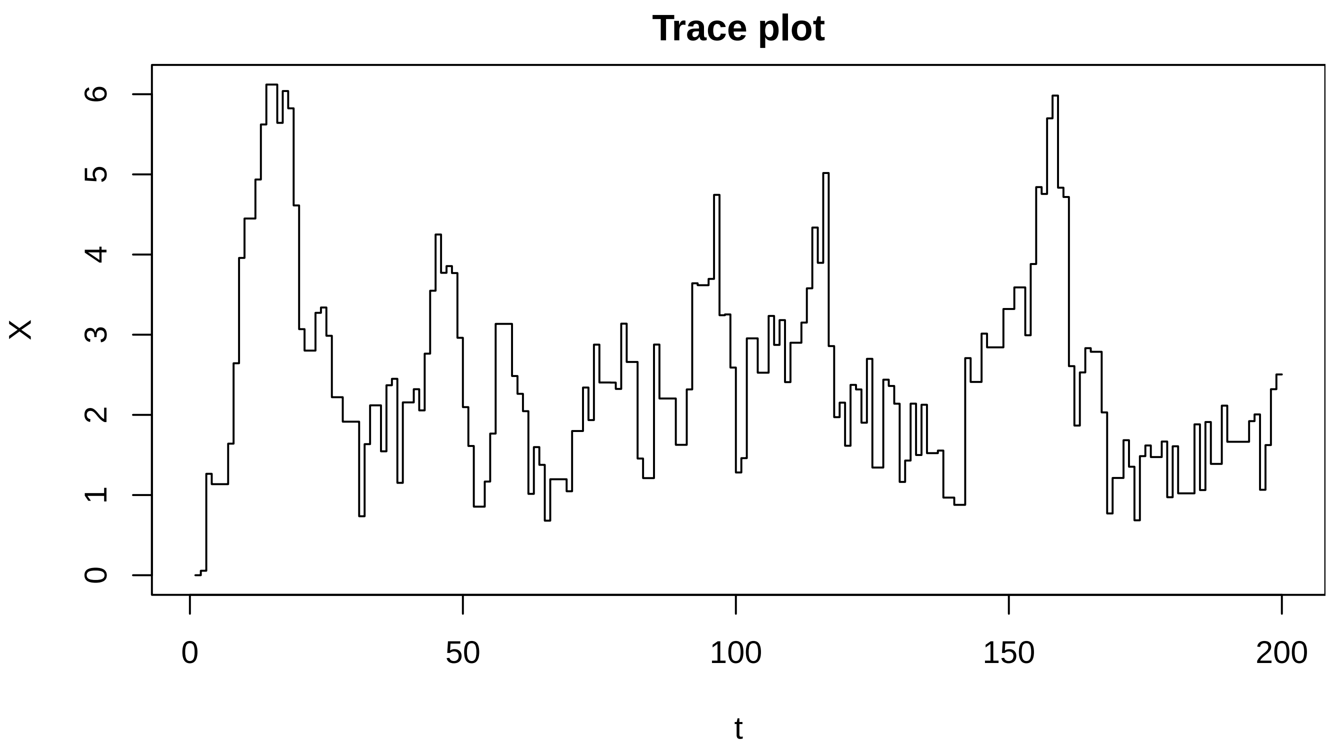
$$\alpha(x, z) = \min\left\{1, \frac{\pi(z)}{\pi(x)}\right\}.$$

Different results are obtained for different settings. Performance can be assessed using

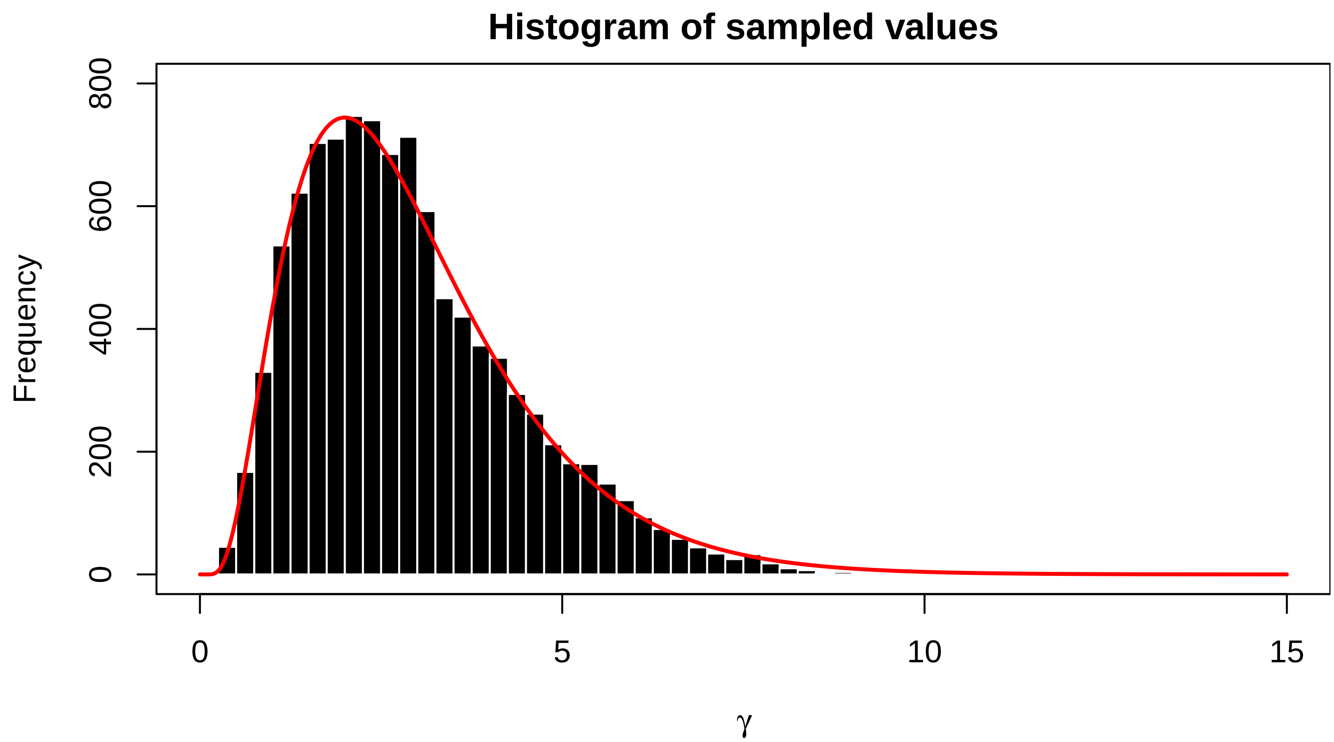
- trace plots of sampled $\{x_t\}$;
- the autocorrelation function (acf) of the sampled values;
- the computed effective sample size (ESS).

In these examples, $\gamma = 2.5$. First, a run starting from $x_0 = 0$ with $\sigma_\gamma = 1$.

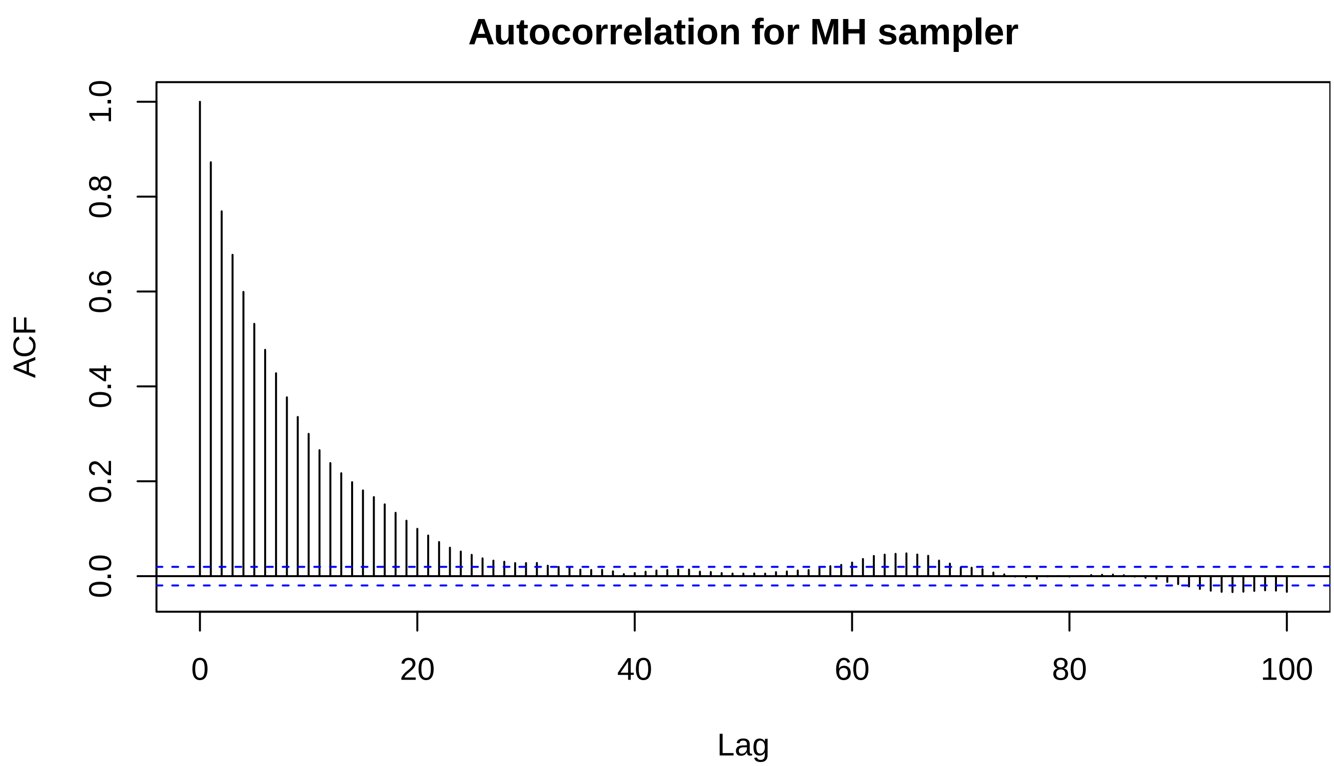
```
set.seed(4532)
new.dens<-function(x,gv){return(exp((gv-1)*log(x)-x-1/x))}
N<-10000
gam<-2.5
sigg<-1
X<-rep(0,N)
X[1]<-0
for(istep in 2:N){
        x<-X[istep-1]
        Z<-abs(rnorm(1,x,sigg))
        al<-min(1,new.dens(Z,gam)/new.dens(x,gam))
        u<-runif(1)
        if(u < al){
                X[istep]<-Z
        }else{
                X[istep]<-x
        }
}

par(mar=c(4,4,2,0))
plot(X[1:200],type="s",xlab="t",ylab="X",main='Trace plot')
```




```
par(mar=c(4,4,2,0))
hist(X,br=seq(0,15,by=0.25),xlab=expression(gamma),main="Histogram of sampled values",
    col="black",border="white",ylim=range(0,800));box()
xv<-seq(0,15,by=0.01);yv<-const*new.dens(xv,gam);lines(xv,yv*N*0.25,col="red",lwd=2)
```




```
par(mar=c(4,4,2,0))
acf(X,lag.max=100,main=' ');title("Autocorrelation for MH sampler",line=1)
```

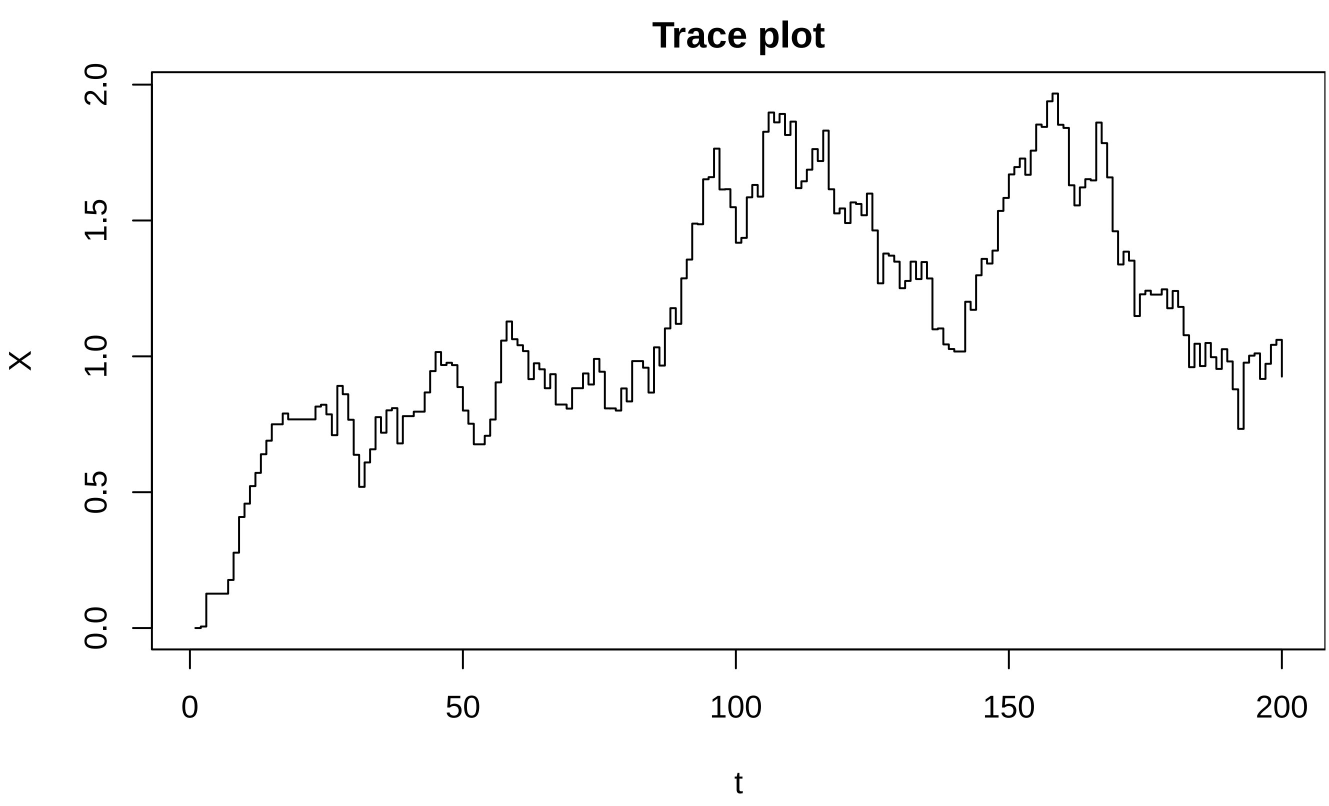



```
library(coda)
effectiveSize(X)
```

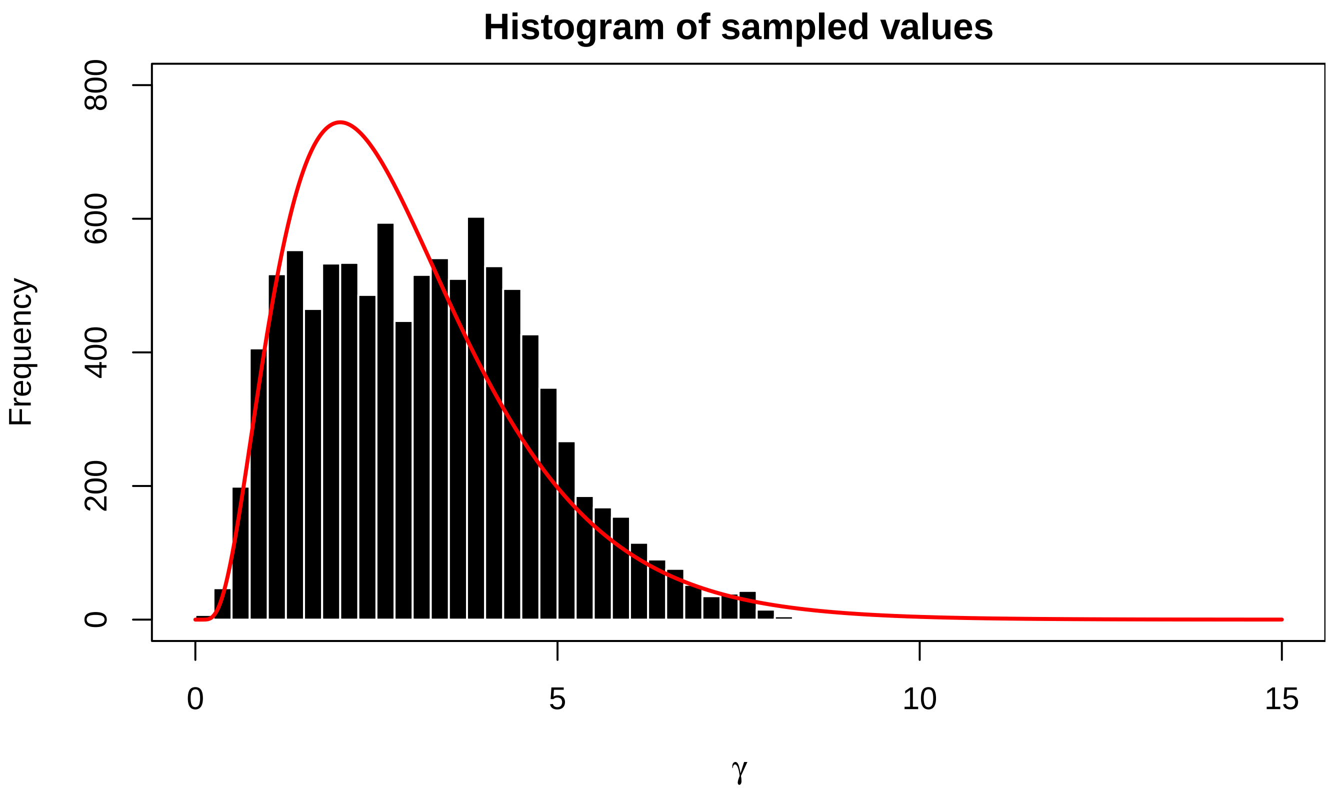
```
+    var1
+ 618.954
```

Here the effective sample size is 618.954003, which is quite low given the run length. We now run the algorithm again with $\sigma_\gamma = 0.1$

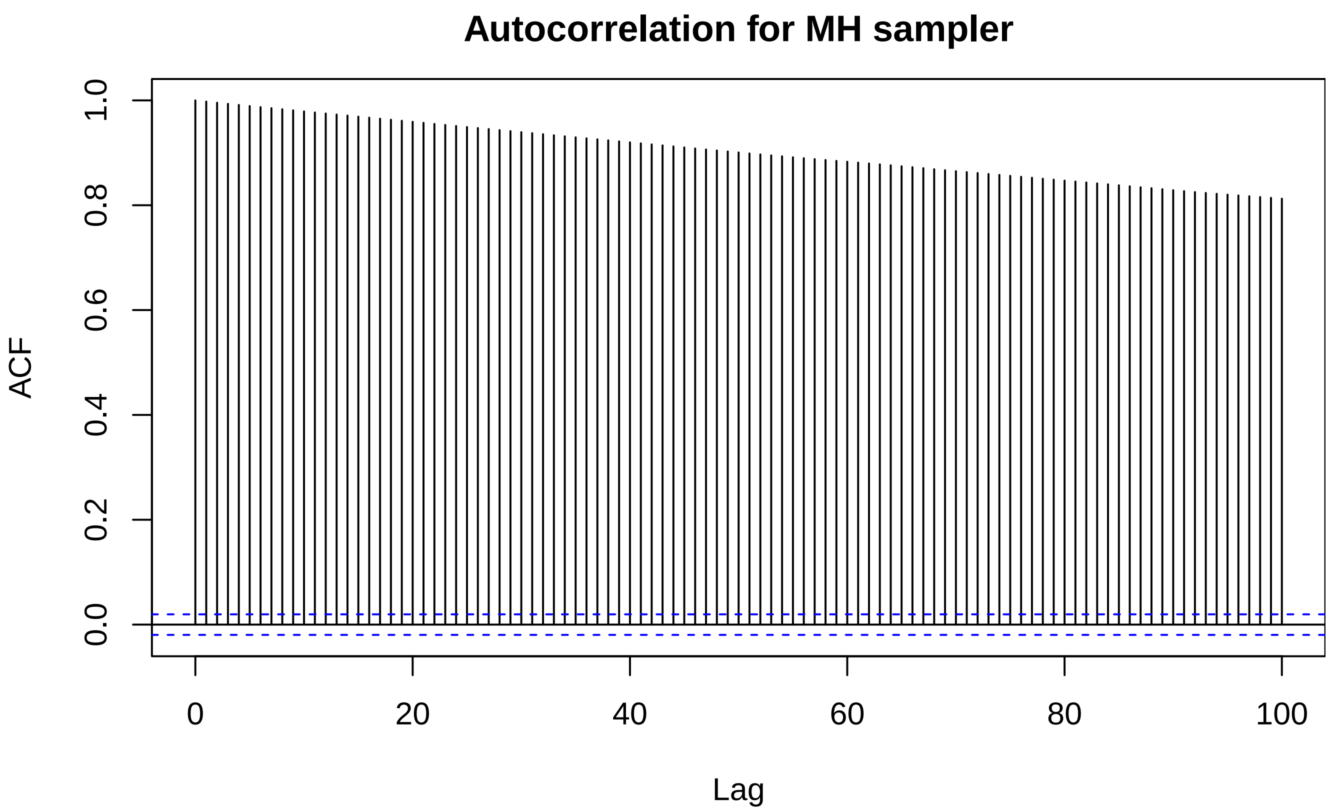
```
set.seed(4532)
N<-10000
gam<-2.5
sigg<-0.1
X<-rep(0,N)
X[1]<-0
for(istep in 2:N){
        x<-X[istep-1]
        Z<-abs(rnorm(1,x,sigg))
        al<-min(1,new.dens(Z,gam)/new.dens(x,gam))
        u<-runif(1)
        if(u < al){
                X[istep]<-Z
        }else{
                X[istep]<-x
        }
}
par(mar=c(4,4,2,0))
plot(X[1:200],type="s",xlab="t",ylab="X",main='Trace plot')
```




```
par(mar=c(4,4,2,0))
hist(X,br=seq(0,15,by=0.25),xlab=expression(gamma),main="Histogram of sampled values",
    col="black",border="white",ylim=range(0,800));box()
xv<-seq(0,15,by=0.01);yv<-const*new.dens(xv,gam)
lines(xv,yv*N*0.25,col="red",lwd=2)
```



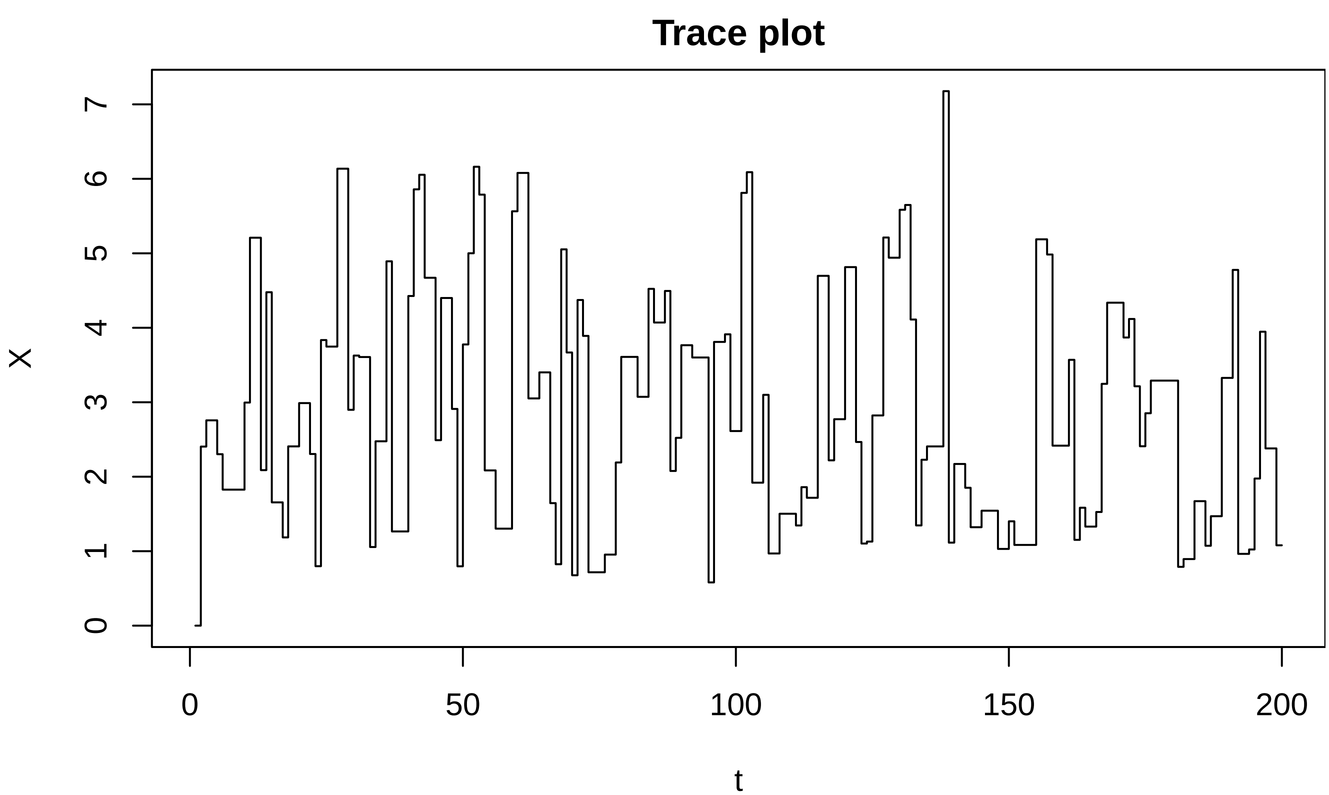

```
par(mar=c(4,4,2,0))
acf(X,lag.max=100,main=' ');title("Autocorrelation for MH sampler",line=1)
```



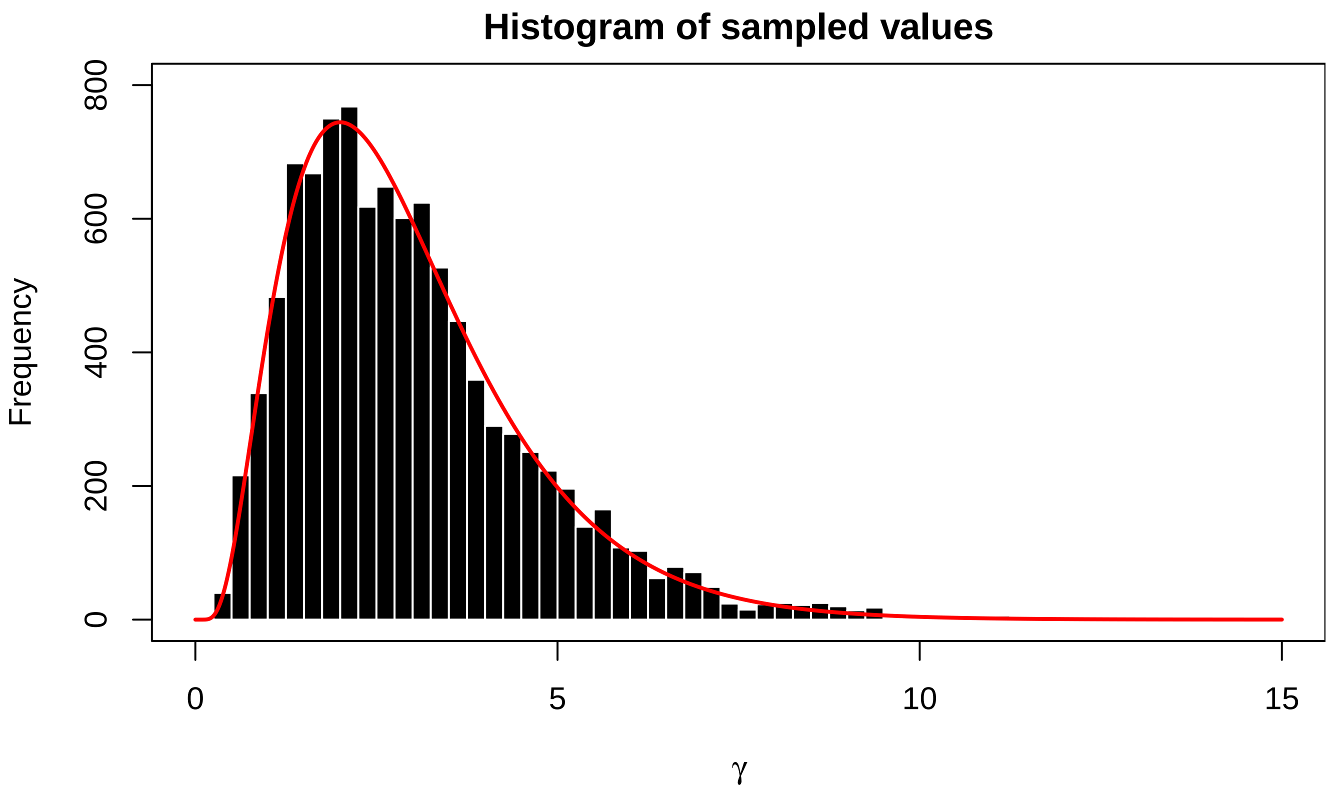
```
library(coda)
effectiveSize(X)

+      var1
+ 11.07978
```

Here the effective sample size is 11.079784, which is extremely poor. Now with $\sigma_\gamma = 3$.

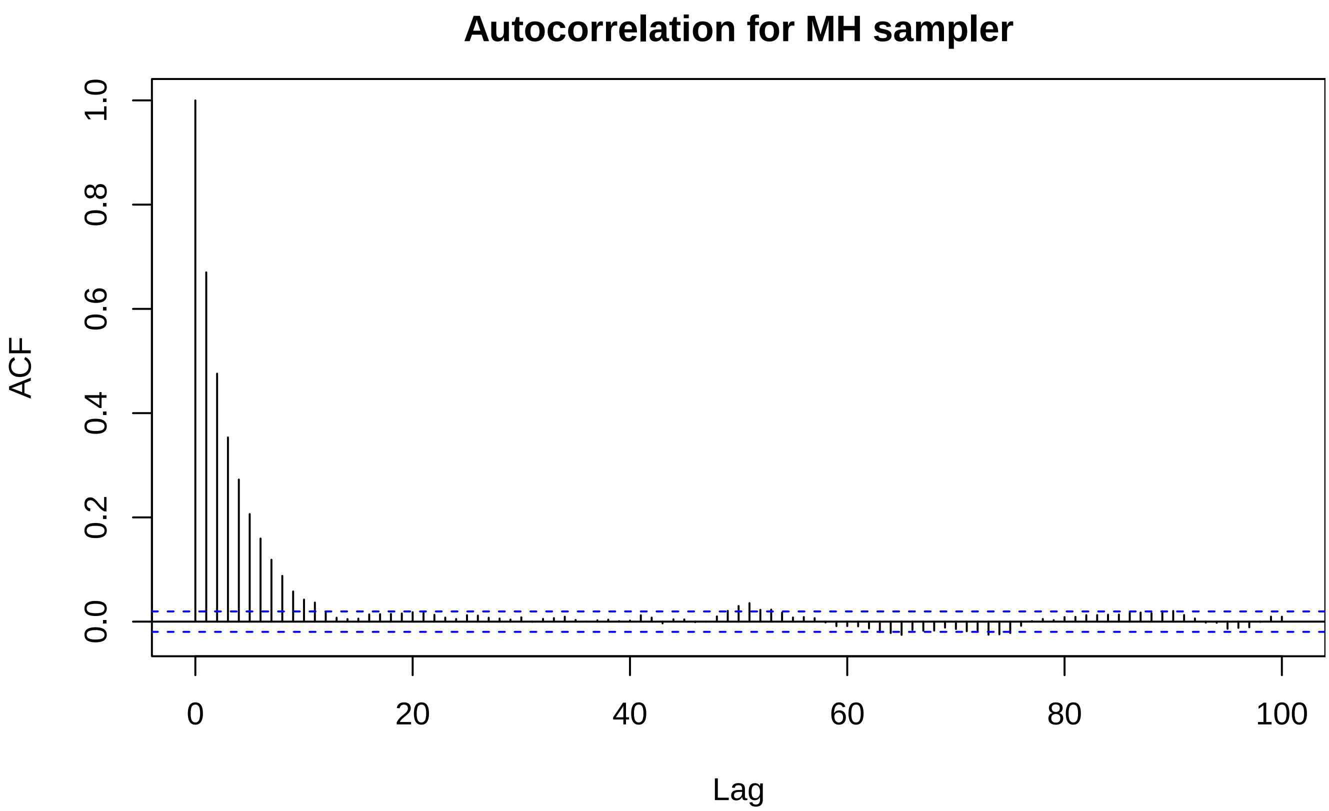
```
set.seed(48532)
N<-10000
gam<-2.5; sigg<-3
X<-rep(0,N);X[1]<-0
for(istep in 2:N){
        x<-X[istep-1]
        Z<-abs(rnorm(1,x,sigg))
        al<-min(1,new.dens(Z,gam)/new.dens(x,gam))
        u<-runif(1)
        if(u < al){
                X[istep]<-Z
        }else{
                X[istep]<-x
        }
}

par(mar=c(4,4,2,0))
plot(X[1:200],type="s",xlab="t",ylab="X",main='Trace plot')
```




```
par(mar=c(4,4,2,0))
hist(X,br=seq(0,15,by=0.25),xlab=expression(gamma),main="Histogram of sampled values",
    col="black",border="white",ylim=range(0,800));box()
xv<-seq(0,15,by=0.01);yv<-const*new.dens(xv,gam)
lines(xv,yv*N*0.25,col="red",lwd=2)
```




```r
par(mar=c(4,4,2,0))
acf(X,lag.max=100,main=' ');title("Autocorrelation for MH sampler",line=1)
```



```
effectiveSize(X)

+     var1
+ 1603.031
```

Here the effective sample size is 1603.030644, which is much better.

Finally we start the chain with $x_0 = 20$, and use $\sigma_\gamma = 0.1$. In the plots, the central 0.95 high probability region of the target distribution is marked by the horizontal red lines; the chain should stabilize to spend 95% of the time in that region.

```
set.seed(48532)
N<-20000
gam<-2.5; sigg<-0.1
X<-rep(0,N);X[1]<-20
for(istep in 2:N){
        x<-X[istep-1]
        Z<-abs(rnorm(1,x,sigg))
        al<-min(1,new.dens(Z,gam)/new.dens(x,gam))
        u<-runif(1)
        if(u < al){
                X[istep]<-Z
        }else{
                X[istep]<-x
        }
}
```

Eventually, the chain recovers from the extreme starting value. To regard the collected sample as a correlated sample from the target, we should discard the initial iterations. However, the effective sample sizes are very low due to the high sample autocorrelation in the sampled values.

```
effectiveSize(X[1:1000])  #out of 1000

+     var1
+ 2.270879

effectiveSize(X[1:10000]) #out of 10000

+     var1
+ 1.840849

effectiveSize(X[1:N])     #out of 20000

+     var1
+ 4.054752

effectiveSize(X[15001:N]) #out of 5000

+     var1
+ 5.511462
```