MATH 559 - ASSIGNMENT 4 - SOLUTIONS

The continuous univariate pdf

$$f(x) = cx^3 \exp\{-x/2 - 1/\sqrt{x}\} \qquad x > 0$$

for constant $c > 0$ is a non-standard form.

Using Monte Carlo methods based on

(a) *Rejection sampling* 8 MARKS

(b) *The Metropolis-Hastings Algorithm* 8 MARKS

estimate

$$\mathbb{E}_f[X]$$

and compare the variances of your Monte Carlo estimators over replicated estimation runs. 4 MARKS

First, for comparison, we can attempt to compute c and the expectation using numerical integration: this is not required for the solution, but will be useful for assessing the success of the algorithms. We can write

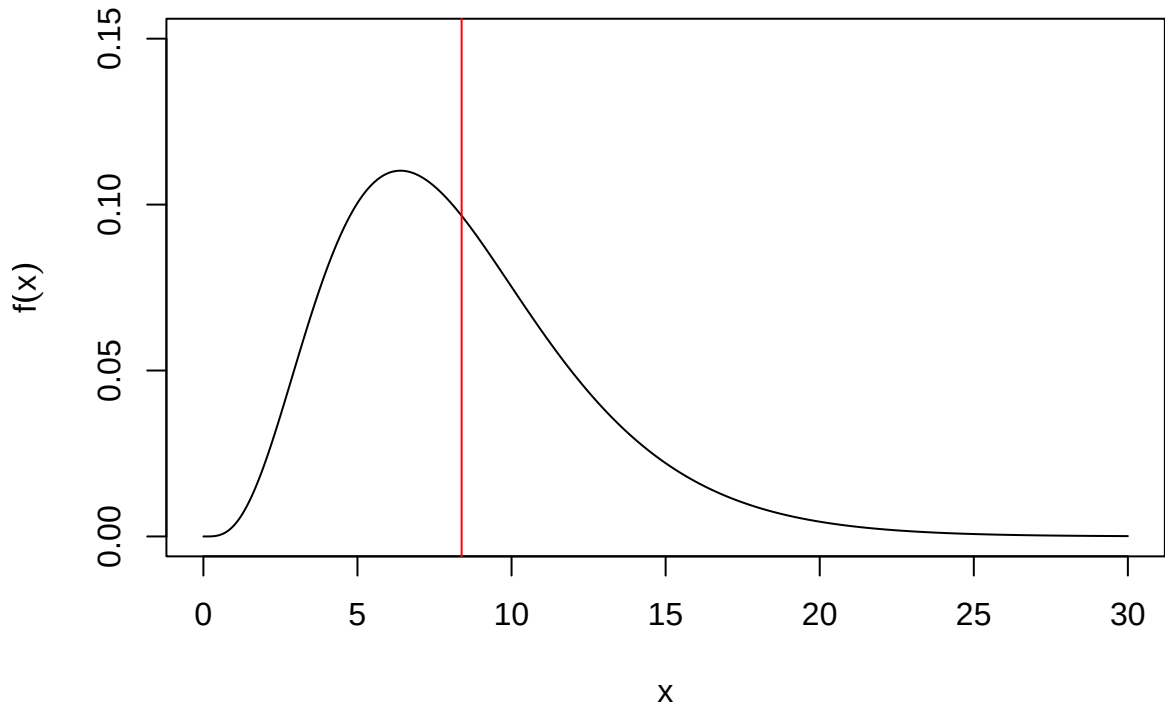
$$I_a = \int_0^\infty x^{3+a} \exp\{-x/2 - 1/\sqrt{x}\}\, dx$$

and then note that $I_0 = 1/c$ and $I_1 = \mathbb{E}_f[X]/c$

```
dens.func<-function(x,av=3,log=FALSE){

    val<-av*log(x)-x/2-1/sqrt(x)
    if(!log) val<-exp(val)
    return(val)

}
I0<-integrate(dens.func,lower=0,upper=Inf)
I1<-integrate(dens.func,av=4,lower=0,upper=Inf)
cval<-1/I0$value;cval

+ [1] 0.01531501

true.val<-cval*I1$value;true.val

+ [1] 8.379553
```

implying that $c = 0.01532$ and $\mathbb{E}_f[X] = 8.37955$.

```
xv<-seq(0,30,by=0.01)
yv<-cval*dens.func(xv)
par(mar=c(4,4,2,0))
plot(xv,yv,type='l',xlab='x',ylab=expression(f(x)),ylim=range(0,0.15))
abline(v=true.val,col='red')
```



(a) **Rejection sampling**: The pdf $f(x)$ quite closely resembles a $Gamma(4, 1/2)$ and so we take that as the proposal $f_0(x)$. Then

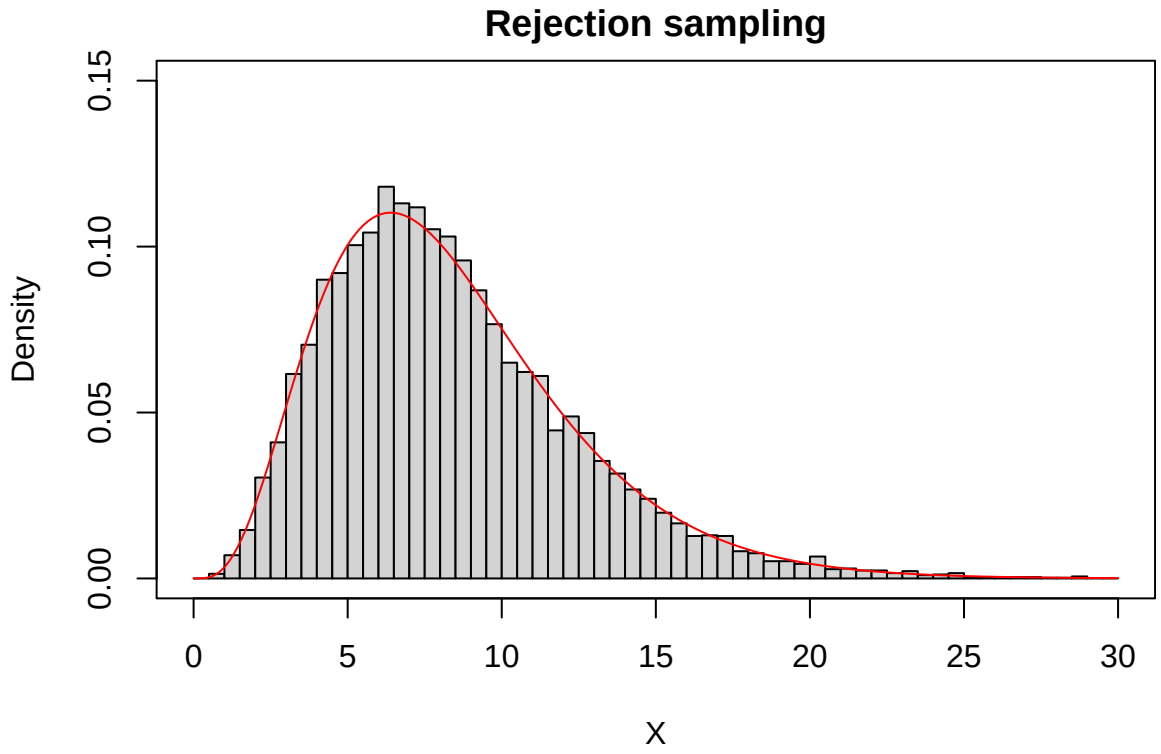
$$\frac{f(x)}{f_0(x)} = c\exp\{-1/\sqrt{x}\} < c$$

so the ratio is certainly bounded, and we may take $M = 1$ for the un-normalized version. The algorithm is therefore simple

```
nsamp<-10000
ico<-itot<-0
samp.X<-numeric()
while(ico < nsamp){
    X<-rgamma(nsamp,4,0.5)
    logu<-log(runif(nsamp))
    frat<--1/sqrt(X)
    ival<-logu < frat
    ico<-ico+sum(ival)
    samp.X<-c(samp.X,X[ival])
    itot<-itot+nsamp
}
c(ico,itot,ico/itot)

+ [1] 13611.00000 20000.00000     0.68055

samp.X<-samp.X[1:nsamp]
```

The acceptance rate is therefore 0.68055.

```
par(mar=c(4,4,2,0))
inc<-samp.X < 30
hist(samp.X[inc],breaks=seq(0,30,by=0.5),freq=FALSE,ylim=range(0,0.15),
            main='Rejection sampling',xlab='X')
box()
lines(xv,yv,col='red')
```




```
mean(samp.X)

+ [1] 8.368559
```

8 MARKS

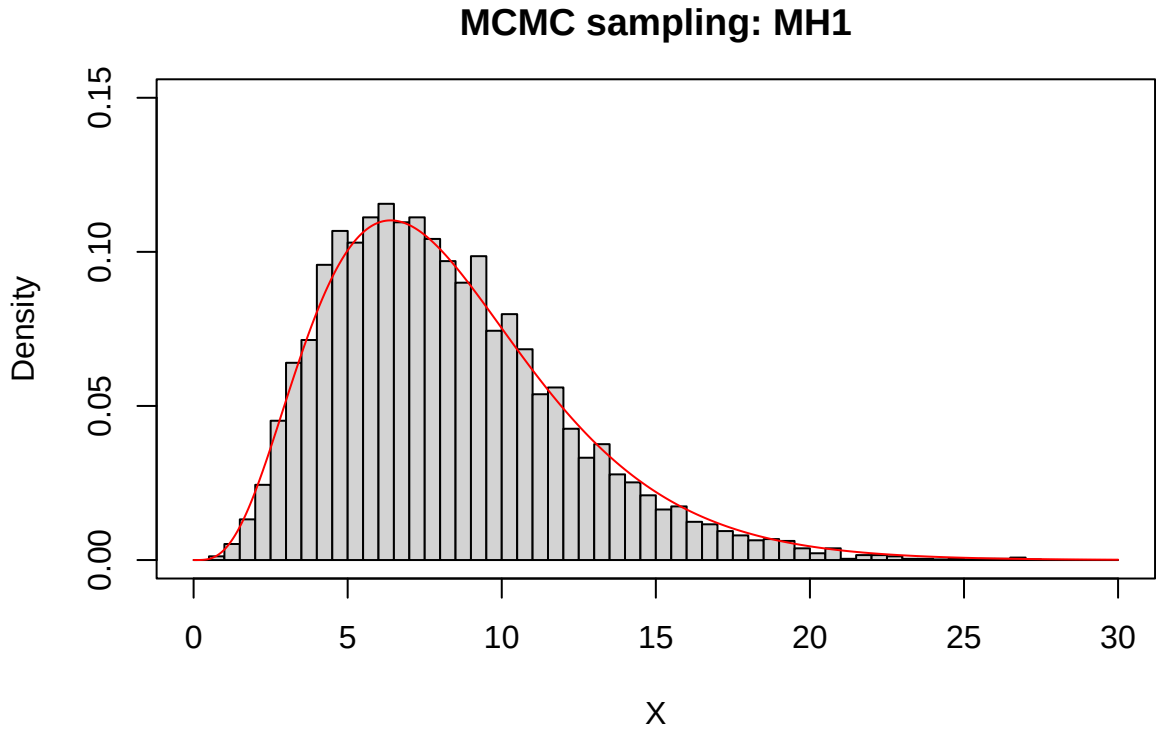
(b) **The Metropolis-Hastings Algorithm** For the Metropolis-Hastings algorithm we may select first the reflected Normal proposal with variance $\sigma_q^2 = 2^2$:

```
nburn<-20
nthin<-1
nits<-nburn+nsamp*nthin
old.x<-2
old.lfx<-dens.func(old.x,log=TRUE)
sigq<-2
MH.samp1<-rep(0,nsamp)
ico<-acc.count<-0
for(iter in 1:nits){
    new.x<-abs(old.x+rnorm(1)*sigq)
    new.lfx<-dens.func(new.x,log=TRUE)
    if(log(runif(1)) < new.lfx - old.lfx){
        old.x<-new.x
        old.lfx<-new.lfx
        acc.count<-acc.count+1
    }
    if(iter > nburn & iter %% nthin == 0){
        ico<-ico+1
        MH.samp1[ico]<-old.x
    }
}
```

The acceptance rate of MH proposals is 0.83583.

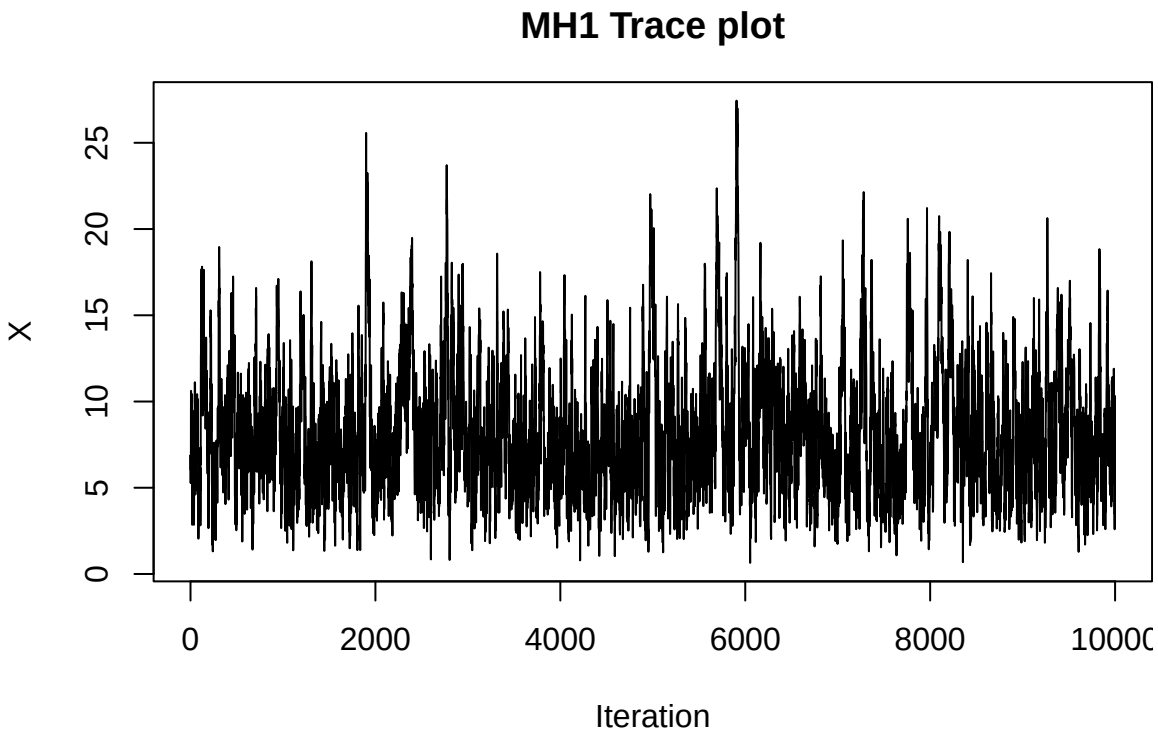
```
library(coda)
ESS1<-effectiveSize(MH.samp1);ESS1

+     var1
+ 447.7063
```

```
par(mar=c(4,4,3,0))
inc<-MH.samp1 < 30
hist(MH.samp1[inc],breaks=seq(0,30,by=0.5),freq=FALSE,ylim=range(0,0.15),
            main='MCMC sampling: MH1',xlab='X');box()
lines(xv,yv,col='red')
```

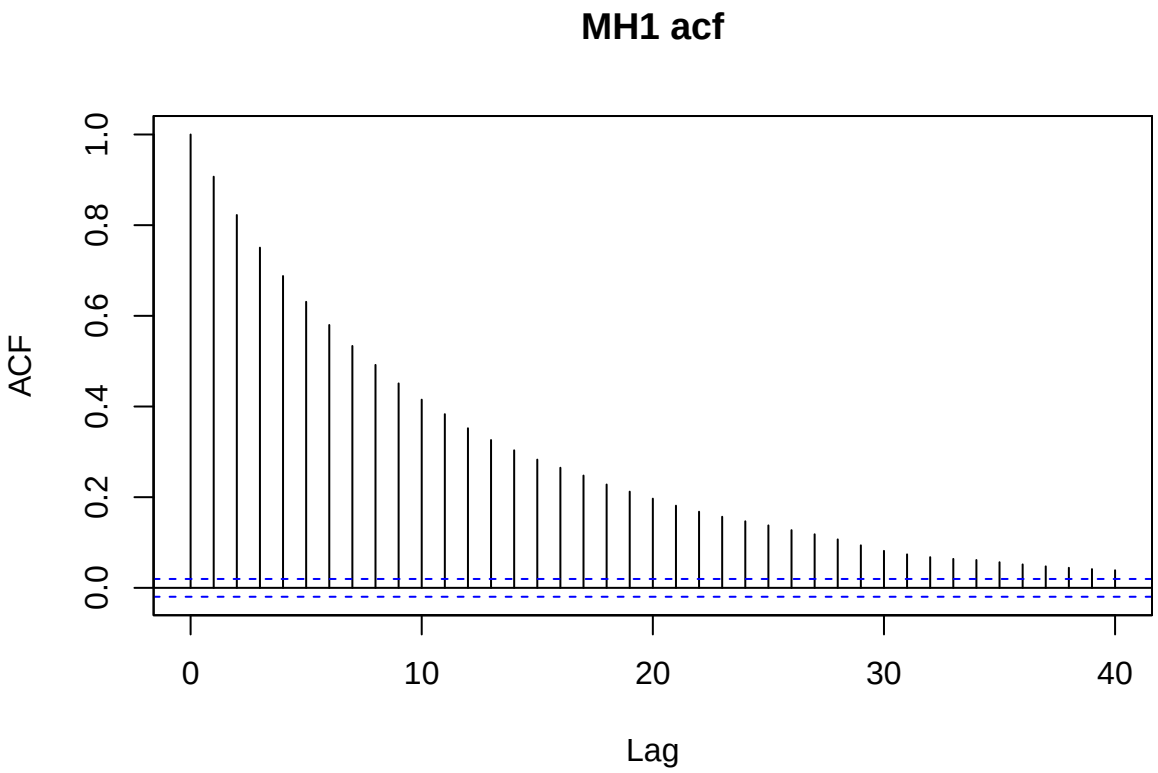



```
mean(MH.samp1)

+ [1] 8.18698
```

```
par(mar=c(4,4,3,0))
plot(MH.samp1,type='l',xlab='Iteration',ylab='X',main='MH1 Trace plot')
```




```
acf(MH.samp1,main='MH1 acf');box()
```



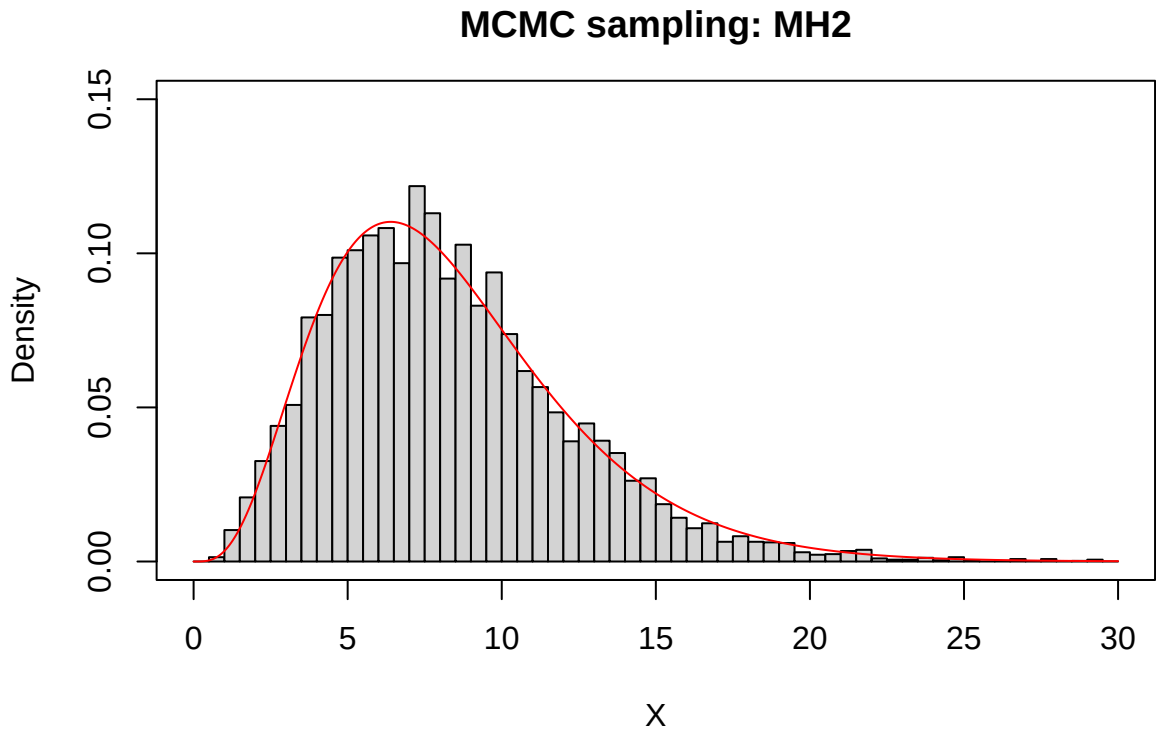
Performance seems ok, but the acceptance rate is rather high, and the effective sample size quite low at 447.70627. Therefore we repeat with $\sigma_q = 8$.

```
old.x<-2
old.lfx<-dens.func(old.x,log=TRUE)
sigq<-8
MH.samp2<-rep(0,nsamp)
ico<-acc.count<-0
for(iter in 1:nits){
    new.x<-abs(old.x+rnorm(1)*sigq)
    new.lfx<-dens.func(new.x,log=TRUE)
    if(log(runif(1)) < new.lfx - old.lfx){
        old.x<-new.x
        old.lfx<-new.lfx
        acc.count<-acc.count+1
    }
    if(iter > nburn & iter %% nthin == 0){
        ico<-ico+1
        MH.samp2[ico]<-old.x
    }
}
```

The acceptance rate of MH proposals is now 0.57405.

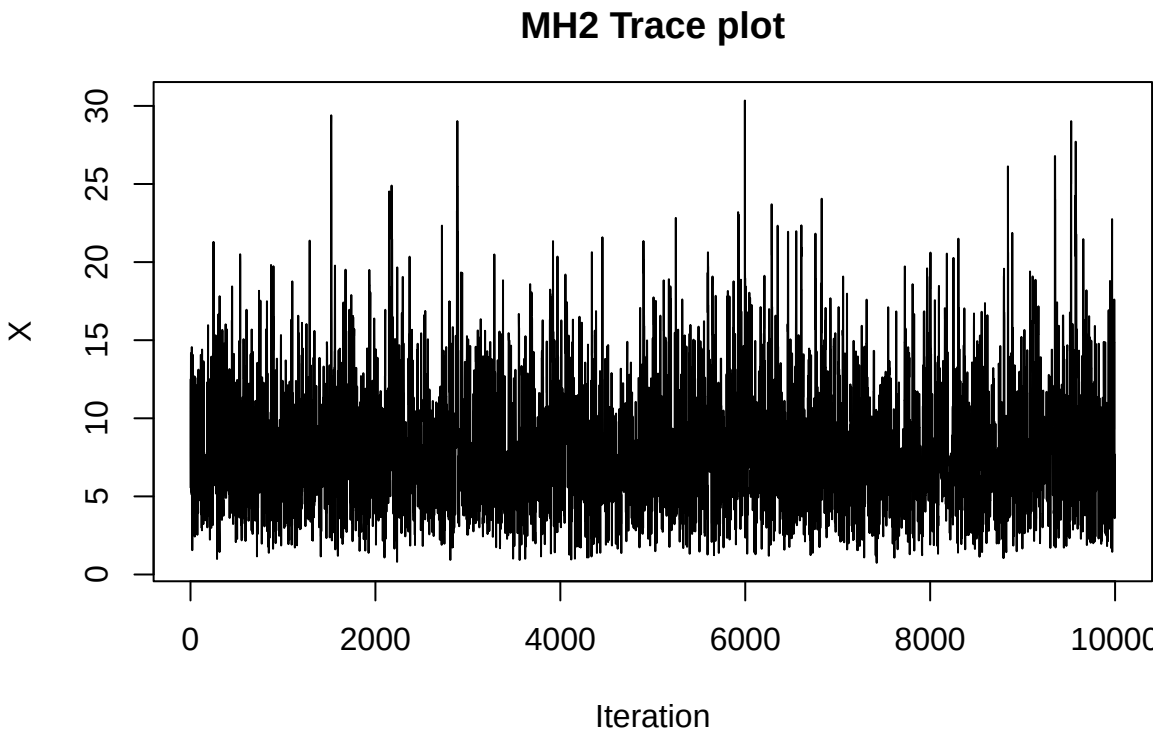
```
ESS2<-effectiveSize(MH.samp2);ESS2

+     var1
+ 2216.461
```

```
par(mar=c(4,4,3,0))
inc<-MH.samp2 < 30
hist(MH.samp2[inc],breaks=seq(0,30,by=0.5),freq=FALSE,ylim=range(0,0.15),
            main='MCMC sampling: MH2',xlab='X');box()
lines(xv,yv,col='red')
```

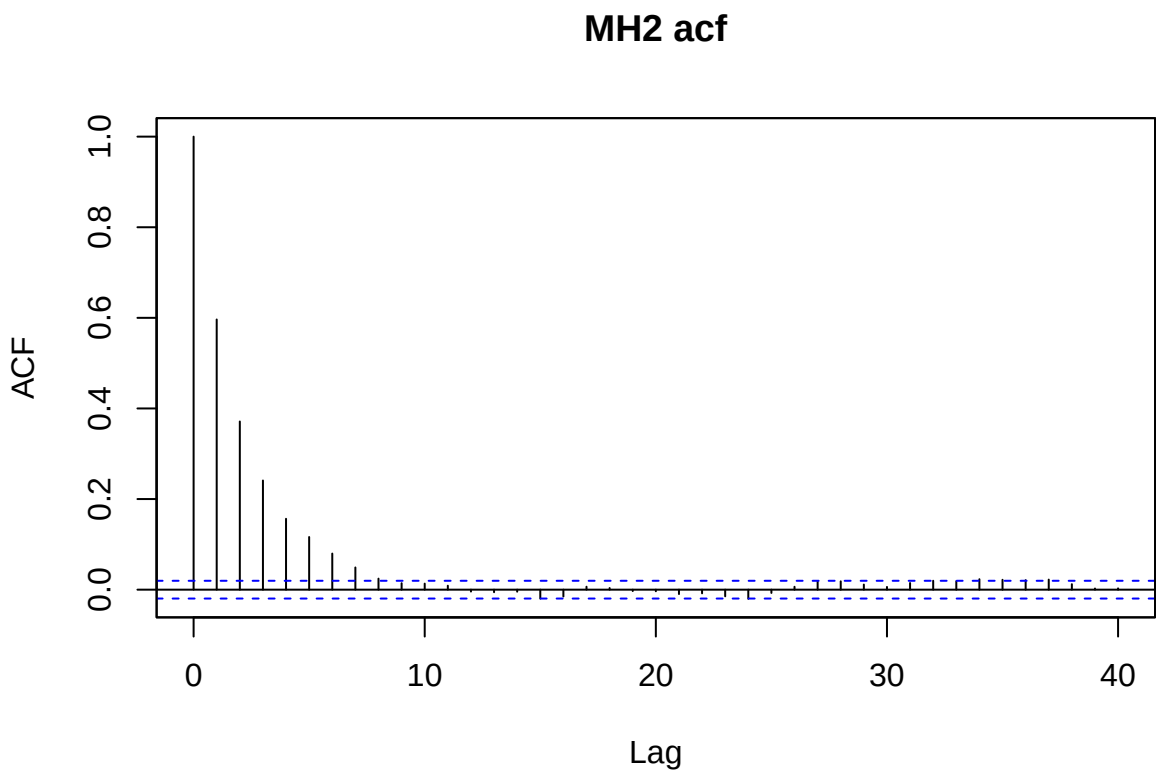


```
mean(MH.samp2)

+ [1] 8.271623
```

```
par(mar=c(4,4,3,0))
plot(MH.samp2,type='l',xlab='Iteration',ylab='X',main='MH2 Trace plot')
```




```
acf(MH.samp2,main='MH2 acf');box()
```



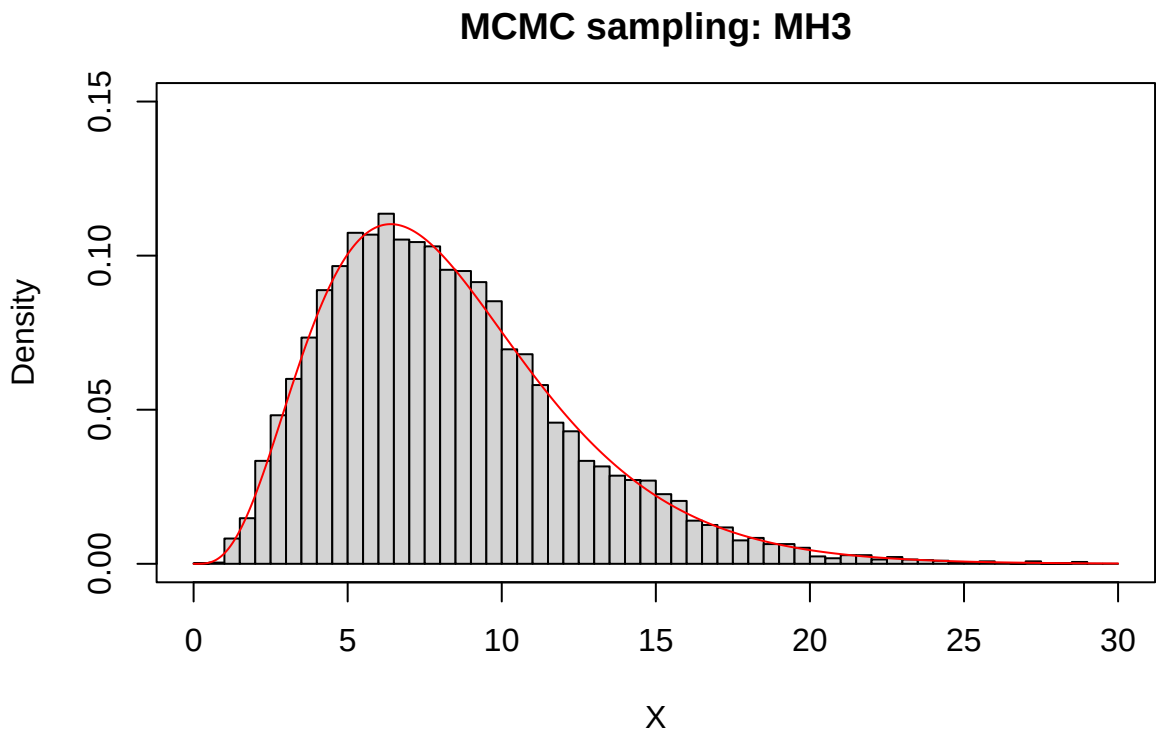
Performance seems better, with effective sample size now 2216.46099. Finally, we can try an **independence** proposal with $q(x, z) \equiv Gamma(4, 1/2)$.

```
old.x<-2
old.lfx<-dens.func(old.x,log=TRUE)
old.qx<-dgamma(old.x,4,0.5,log=T)
MH.samp3<-rep(0,nsamp)
ico<-acc.count<-0
for(iter in 1:nits){
    new.x<-rgamma(1,4,0.5)
    new.lfx<-dens.func(new.x,log=TRUE)
    new.qx<-dgamma(new.x,4,0.5,log=T)
    if(log(runif(1)) < new.lfx+old.qx - old.lfx-new.qx){
        old.x<-new.x
        old.lfx<-new.lfx
        old.qx<-new.qx
        acc.count<-acc.count+1
    }
    if(iter > nburn & iter %% nthin == 0){
        ico<-ico+1
        MH.samp3[ico]<-old.x
    }
}
```

The acceptance rate of MH proposals is 0.94291.

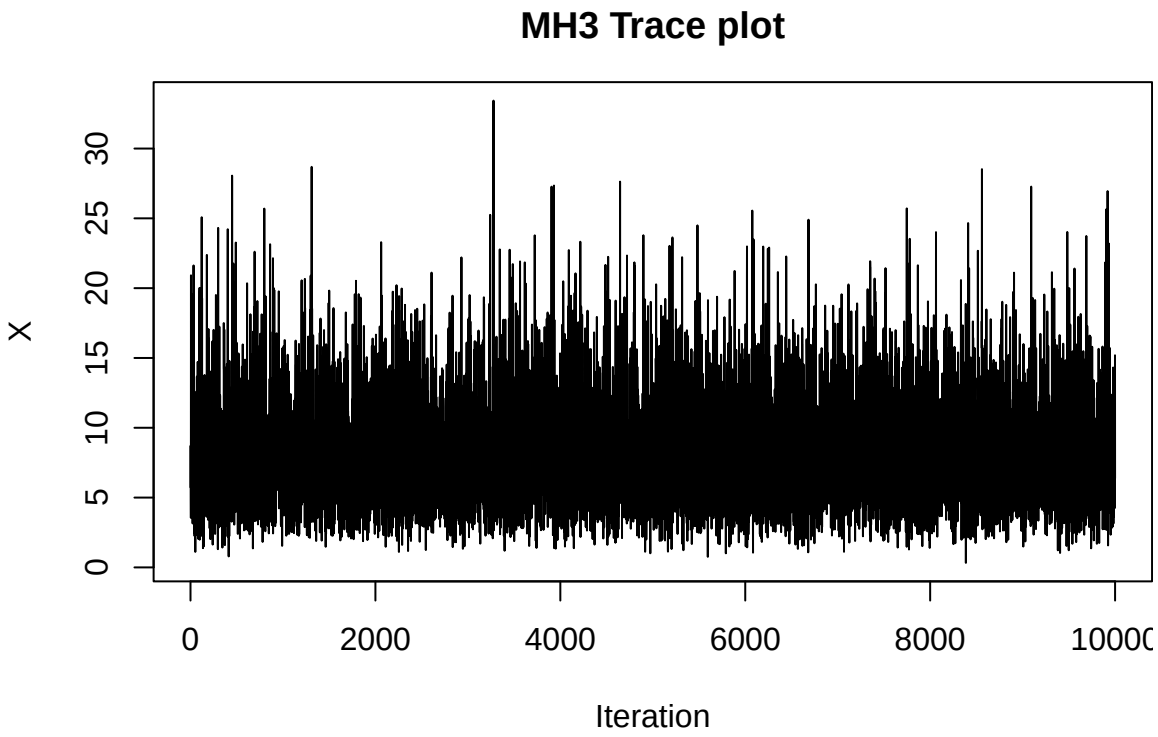
```
ESS3<-effectiveSize(MH.samp3);ESS3

+     var1
+ 7748.769
```

```
par(mar=c(4,4,3,0))
inc<-MH.samp3 < 30
hist(MH.samp3[inc],breaks=seq(0,30,by=0.5),freq=FALSE,ylim=range(0,0.15),
            main='MCMC sampling: MH3',xlab='X')
box()
lines(xv,yv,col='red')
```

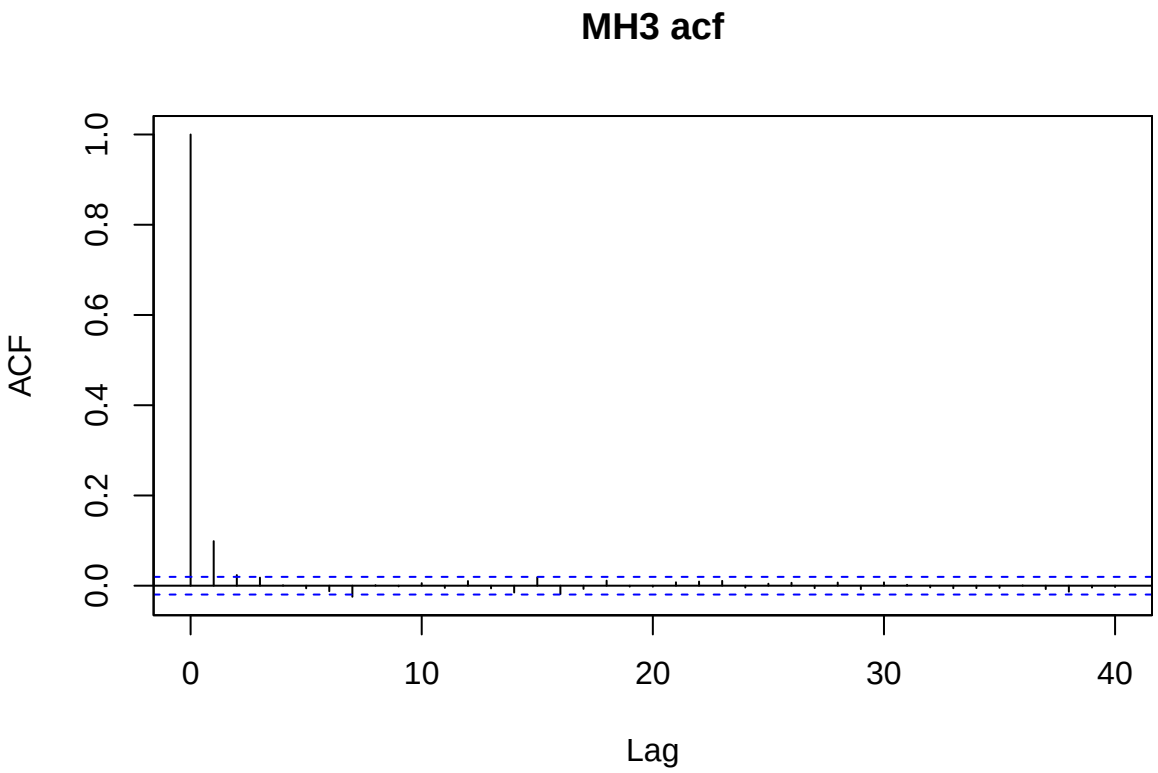



```
mean(MH.samp3)

+ [1] 8.304352
```

```
par(mar=c(4,4,3,0))
plot(MH.samp3,type='l',xlab='Iteration',ylab='X',main='MH3 Trace plot')
```




```
acf(MH.samp3,main='MH3 acf');box()
```



Performance seems good with effective sample size 7748.76910. 8 MARKS

For a comparison of the variance of the estimators, we will write some wrapper functions:

Rejection Sampling:

```
run.rej.samp<-function(nsp=10000){
    ico<-itot<-0
    samp.X<-numeric()
    while(ico < nsp){
        X<-rgamma(nsp,4,0.5)
        logu<-log(runif(nsp))
        frat<--1/sqrt(X)
        ival<-logu < frat
        ico<-ico+sum(ival)
        samp.X<-c(samp.X,X[ival])
        itot<-itot+nsp
    }
    samp.X<-samp.X[1:nsp]
    return(samp.X)
}
nreps<-500
rej.samp<-t(replicate(nreps,run.rej.samp(nsp=2000)))
rej.ests<-apply(rej.samp,1,mean)
mean(rej.ests)

+ [1] 8.385039

var(rej.ests)

+ [1] 0.007628961
```

Local Metropolis-Hastings Sampling:

```
run.local.mh.samp<-function(nsp=10000,sq=2,nb=20,nth=1){
    nits<-nb+nsp*nth
    oldx<-2
    oldlfx<-dens.func(oldx,log=TRUE)
    MHsamp<-rep(0,nsp)
    ico<-0
    for(iter in 1:nits){
        newx<-abs(oldx+rnorm(1)*sq)
        newlfx<-dens.func(newx,log=TRUE)
        if(log(runif(1)) < newlfx - oldlfx){
            oldx<-newx
            oldlfx<-newlfx
        }
        if(iter > nb & iter %% nth == 0){
            ico<-ico+1
            MHsamp[ico]<-oldx
        }
    }
    return(MHsamp)
}
```

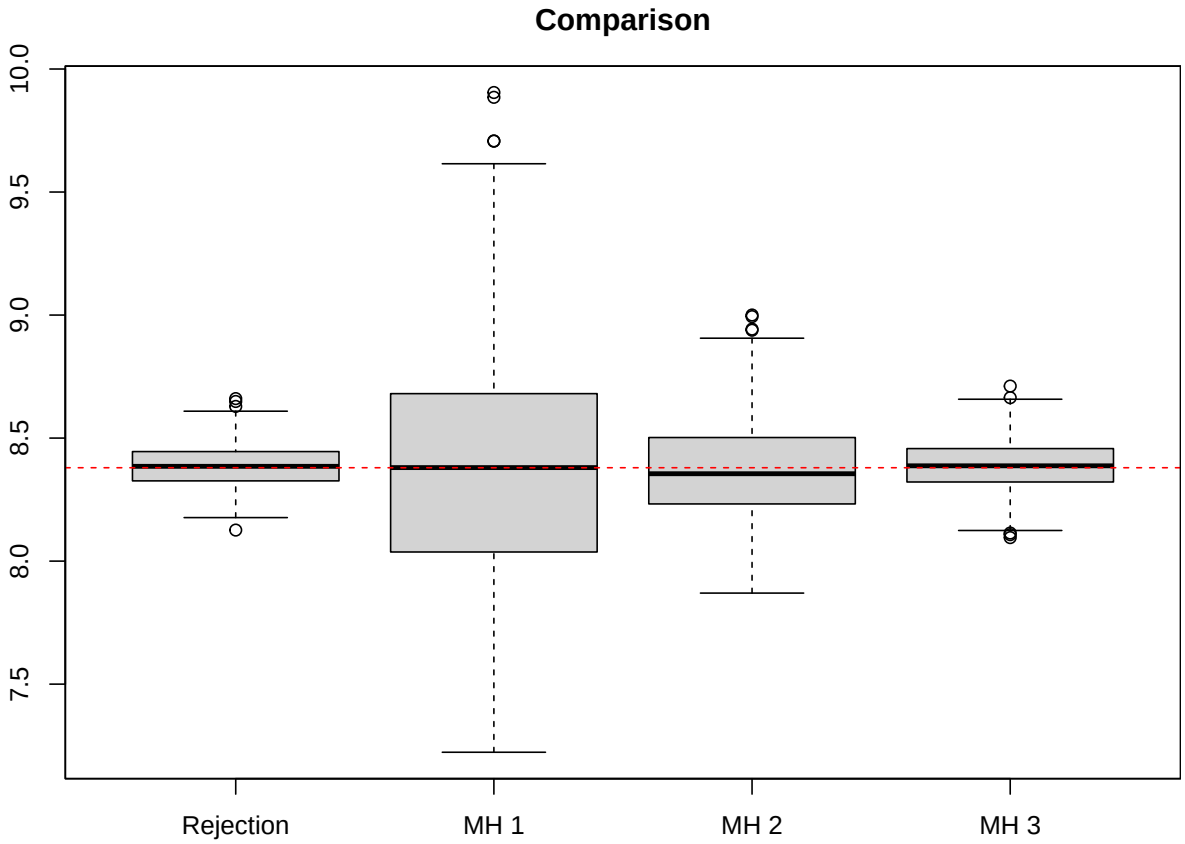
```
nreps<-500
mh1.samp<-t(replicate(nreps,run.local.mh.samp(nsp=2000,sq=2)))
mh1.ests<-apply(mh1.samp,1,mean)
mean(mh1.ests)

+ [1] 8.377135

var(mh1.ests)

+ [1] 0.2370366

mh2.samp<-t(replicate(nreps,run.local.mh.samp(nsp=2000,sq=8)))
mh2.ests<-apply(mh2.samp,1,mean)
mean(mh2.ests)

+ [1] 8.366098

var(mh2.ests)

+ [1] 0.04020222
```

Independence Metropolis-Hastings Sampling:

```
run.indpt.mh.samp<-function(nsp=10000,nb=20,nth=1){
    nits<-nb+nsp*nth
    oldx<-2
    oldlfx<-dens.func(oldx,log=TRUE)
    oldqx<-dgamma(oldx,4,0.5,log=T)
    MHsamp<-rep(0,nsp)
    ico<-0
    for(iter in 1:nits){
        newx<-rgamma(1,4,0.5)
        newlfx<-dens.func(newx,log=TRUE)
        newqx<-dgamma(newx,4,0.5,log=T)
        if(log(runif(1)) < newlfx+oldqx - oldlfx-newqx){
            oldx<-newx
            oldlfx<-newlfx
            oldqx<-newqx
        }
        if(iter > nb & iter %% nth == 0){
            ico<-ico+1
            MHsamp[ico]<-oldx
        }
    }
    return(MHsamp)
}
```

```
nreps<-500
mh3.samp<-t(replicate(nreps,run.indpt.mh.samp(nsp=2000)))
mh3.ests<-apply(mh3.samp,1,mean)
mean(mh3.ests)

+ [1] 8.386358

var(mh3.ests)

+ [1] 0.01052855
```

```
par(mar=c(4,4,3,0))
lvec<-c('Rejection','MH 1','MH 2','MH 3')
emat<-cbind(rej.ests,mh1.ests,mh2.ests,mh3.ests)
boxplot(emat,names=lvec,main='Comparison')
abline(h=true.val,lty=2,col='red')
box()
```




```
nreps*apply(emat,2,var)  #Comparison of variances

+   rej.ests   mh1.ests   mh2.ests   mh3.ests
+   3.814481 118.518288  20.101109   5.264273
```