18.S096 PSET 6

(Thanks K.P.)

Due: April 13th, 2018

1. **Problem 1. Monte-Carlo Option Pricing With Non-Gaussian Shocks**

(1a) mean:$\mu = 0 \Rightarrow a = 0$
variance:$2b^2 = 1 \Rightarrow b = \frac{1}{\sqrt{2}}$

```
#Create the function rlaplace() which generates pseudo-random realizations from this

rlaplace=function(n){
laplace_variates<-as.vector(rep(0,n))+
(sqrt(2))^(-1)*(2*rbinom(n,1,.5)-1)*rexp(n,rate=1)
return (laplace_variates)
}
```

(1b) mean: $\mu = E[X] = \frac{1}{n}\sum_{i=1}^{n} X_i$
variance: $\sigma^2 = \frac{v}{v-2}$, where v is the degrees of freedom

```
rt5normalized=function(n){
  rt5normalized_variates<-rt(n,df=5)
  rt5normalized_variates<-(rt5normalized_variates-mean(rt5normalized_variates))/sqrt(
  return(rt5normalized_variates)
}
```

(1c)
```
MCprice2 <- function(Price, Strike,
                     Rate, Time, Volatility, Steps, Paths, densfun) {
  # Price<-100.
  # Strike<-100.
  # Rate <-0.03
  # Time <-1/4
  # Volatility <- 0.20
  # Steps=100
  # Paths=1000
  #

# Monte Carlo pricer for vanilla options [8/8/2016 pfm]
#     Paul F Mende, 18.642 Lecture
# Input arguments use consistent units, e.g., annualized
# Price:  current price of underlying
```

```
# Strike: strike price of option contract
# Rate:   risk-free rate
# Time:   time to expiration
# Volatility
# Steps: number of time steps in discretization
# Paths:  number of Monte Carlo simulation paths

  S0    <- Price
  K     <- Strike
  rf    <- Rate
  T     <- Time
  sigma <- Volatility
  Nt    <- Steps
  Np    <- Paths
  dt    <- T/Nt

 # if (missing(densfun) || !(is.function(densfun) || !is.character(densfun)))
  #  stop("'densfun' must be supplied as a function or name")

distname <- tolower(densfun)
if(!(distname=="normal") & !(distname=="laplace") & !(distname=="t5"))
  stop("density function not supported")
# Select independent, normalized Laplace shocks
  if(distname=="laplace"){
    epsilon <- matrix(rlaplace(Nt*Np), ncol=Np)
  }
# Select independent, normalized Gaussian shocks
  if(distname=="normal"){
  epsilon <- matrix(rnorm(Nt*Np), ncol=Np)
  }
# Select independent, normalized t5 shocks
  if(distname=="t5"){
    epsilon <- matrix(rt5normalized(Nt*Np), ncol=Np)
  }
# Define IID returns for each step and path under risk-neutral measure

r <- (rf - 0.5*sigma^2)*dt + epsilon*sigma*sqrt(dt)

# Construct stochastic paths and price process

logX  <- apply(r,2,cumsum)
S     <- matrix(S0, Nt+1, Np)
S[-1,] <- S0 * exp(logX)

# Define payoff values for derivatives
C <- (S-K) * (S>K)
P <- (K-S) * (S<K)
```

```
# Compute call and put values as discounted expected payoffs
Call <- exp(-rf*T)*mean(C[Nt+1,])
Put  <- exp(-rf*T)*mean(P[Nt+1,])

# Return values
return(data.frame(call=Call, put=Put))
}
```

(1d)
```
## One-year horizon (T=1)
     #  Daily increments (Nt=252)
     #  10,000 paths
     #  At-the-money calls/puts (S0=K=100)
     #
     S0 <- 100; K <- 100; T <- 1; rf <- 0.03; sigma <- 0.3;
     Nt <- 252; Np <- 1e4; dt=T/Nt
     set.seed(1)
     #normal
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="normal")

##       call      put
## 1 13.26994 10.43697

     #laplace
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="laplace")

##      call      put
## 1 13.5304 10.19325

     #t5
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="t5")

##       call      put
## 1 13.31784 10.33615
```

2. **Problem 2. Monte-Carlo Option Pricing (Continued)**

(2a) The CLT would apply in approximating the distribution of the terminal path values. This would lead to obtaining the same MC prices for the different distributions in problem 1 since they are all normalized.

(2b)
```
#Only modify T
 ## One-month horizon (T=1/12)
     #  Daily increments (Nt=252)
     #  10,000 paths
     #  At-the-money calls/puts (S0=K=100)
     #
     S0 <- 100; K <- 100; T <- 1/12; rf <- 0.03; sigma <- 0.3;
```

```
     Nt <- 252; Np <- 1e4; dt=T/Nt
     set.seed(1)
     #normal
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="normal")

##       call      put
## 1 3.573277 3.363073

     #laplace
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="laplace")

##       call      put
## 1 3.640146 3.275013

     #t5
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="t5")

##       call      put
## 1 3.583226 3.331033

#Modify T and Nt
 ## One-month horizon (T=1/12)
     #  Daily increments (Nt=21)
     #  10,000 paths
     #  At-the-money calls/puts (S0=K=100)
     #
     S0 <- 100; K <- 100; T <- 1/12; rf <- 0.03; sigma <- 0.3;
     Nt <- 21; Np <- 1e4; dt=T/Nt
     set.seed(1)
     #normal
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="normal")

##       call      put
## 1 3.554667 3.297664

     #laplace
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="laplace")

##       call      put
## 1 3.470322 3.312862

     #t5
     MCprice2(S0,K,rf,T,sigma,Nt,Np,densfun="t5")

##       call      put
## 1 3.546338 3.296014
```

(2c) The Monte Carlo option price is a bit sensitive to the choice of the shock distribution; the gap between the call and put prices is typically largest when the shock follows a normalized laplace distribution, and is fairly indistinguisible between the normalized t distribution with 5 degrees of freedom and the normal distribution.

The MC option is not very sensitive to N_t, and is very sensitive to T.

3. **Problem 3. Stationary and Ergodic Distributions**

(3a)

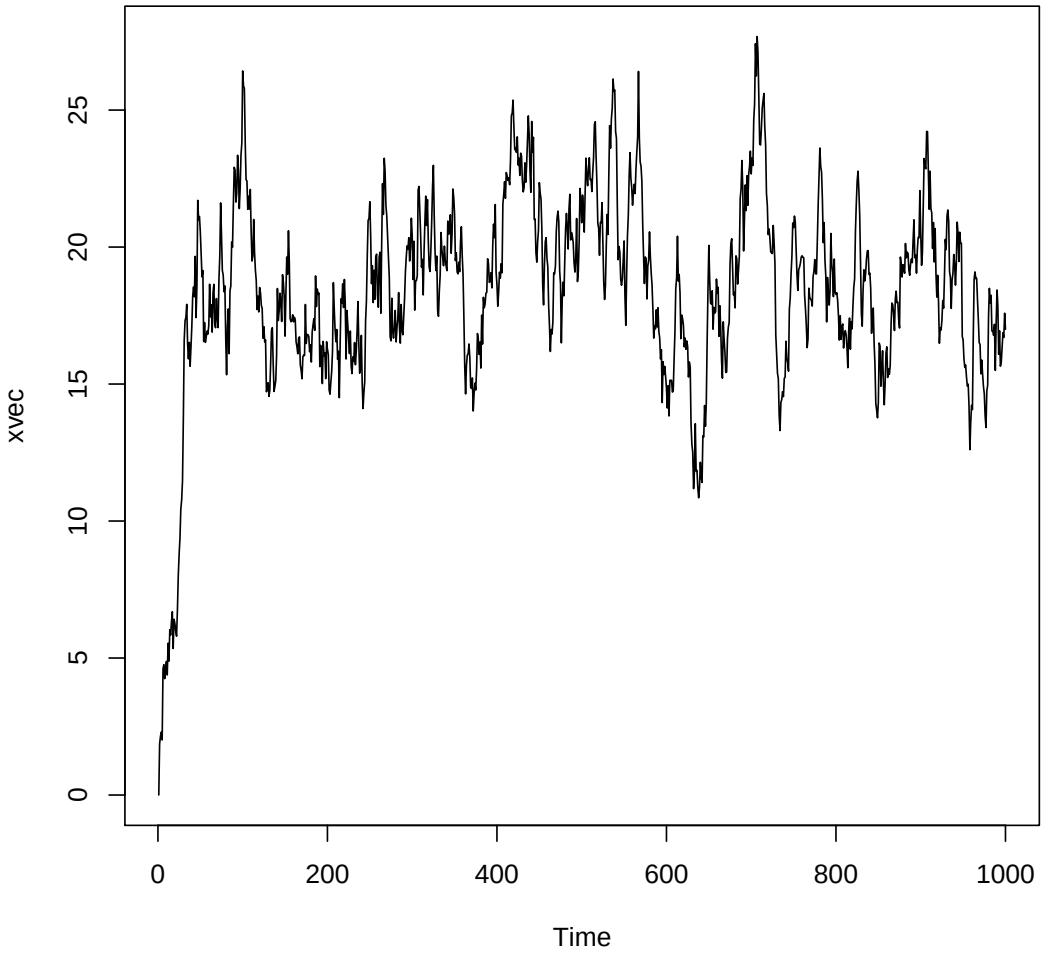
$$X_{t+1} = \alpha_0 + \alpha_1 X_t + \epsilon_{t+1}, \epsilon_t \sim Normal(0, \sigma^2)$$
$$X_{t+1} - \alpha_0 - \alpha_1 X_t \sim Normal(0, \sigma^2)$$
$$f(X_{t+1}|X_t = x_t) = \frac{1}{\sqrt{(2\pi\sigma^2)}} exp[-\frac{1}{2\sigma^2}(X_{t+1} - \alpha_0 - \alpha_1 x_t)^2]$$

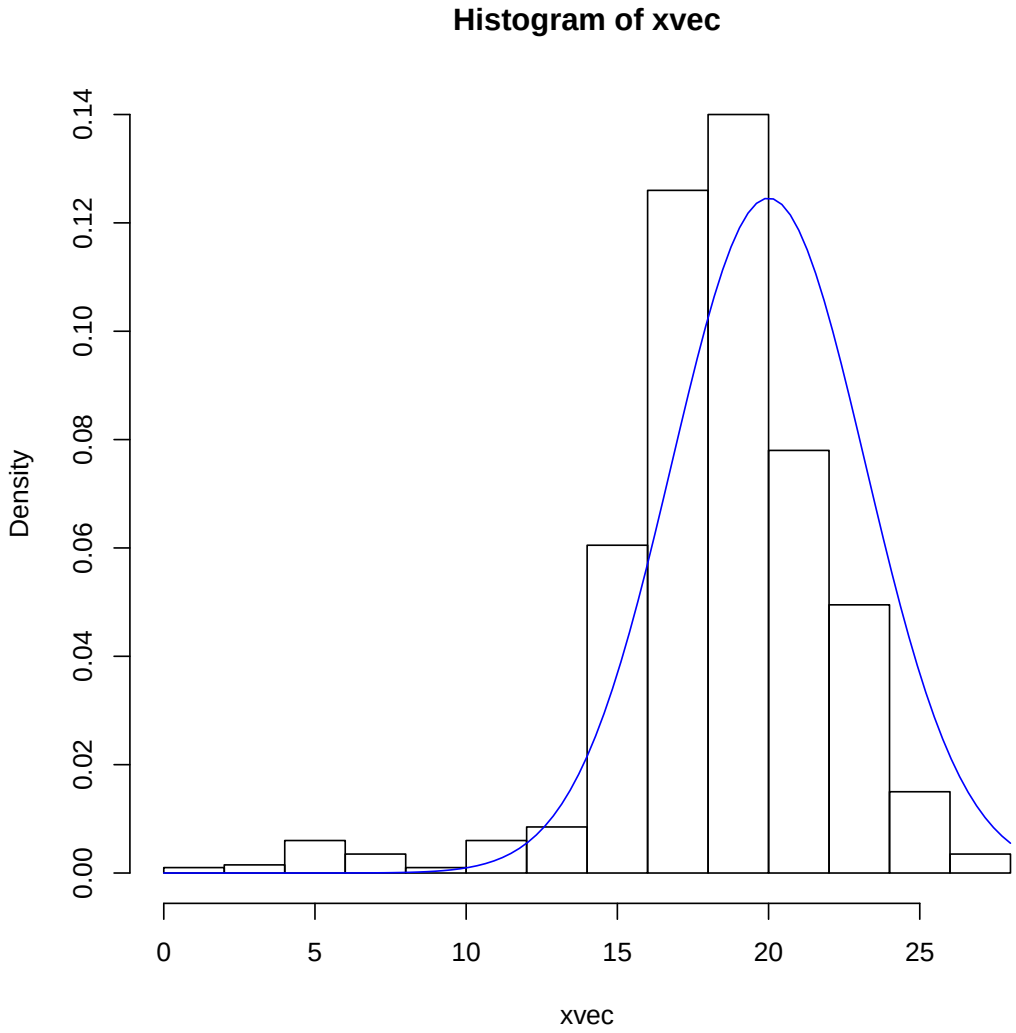
(3b) Since $|\alpha_1| < 1$, we know that the observations have a stationary distribution.

$$\begin{aligned}
X_1|X_0 &= \alpha_0 + \alpha_1 X_0 + \epsilon_1 \\
X_2|X_0, X_1 &= \alpha_0 + \alpha_1 X_1 + \epsilon_2 \\
&= \alpha_0 + \alpha_1(\alpha_0 + \alpha_1 X_0 + \epsilon_1) + \epsilon_2 \\
X_3|X_0, X_1, X_2 &= \alpha_0 + \alpha_1(\alpha_0 + \alpha_1(\alpha_0 + \alpha_1 X_0 + \epsilon_1) + \epsilon_2) + \epsilon_3 \\
X_{t+1}|X_t, X_{t-1}, ... &= \alpha_0 + \alpha_1\alpha_0 + \alpha_1^2\alpha_0 + \alpha_1^3\alpha_0 + ... + \alpha_1^{t+1}X_0 + ... + \alpha_1^2\epsilon + \alpha_1\epsilon + \epsilon \text{ since } \epsilon \text{ are i.i.d}
\end{aligned}$$

Since $|\alpha_1| < 1$, we have a convergent sum; $\alpha_0 + \alpha_1\alpha_0 + \alpha_1^2\alpha_0 + \alpha_1^3\alpha_0 + ... = \frac{\alpha_0}{1-\alpha_1}$. We also note that since $|\alpha_1| < 1$, we can treat the $\alpha_1^{t+1}X_0$ term as 0, and the sum of $... + \alpha_1^2\epsilon + \alpha_1\epsilon + \epsilon$ is a (convergent) sum of i.i.d normal variables with variance σ^2 scaled by $\frac{1}{1-\alpha_1^2}$. Hence, by the same argument used in (3a), the marginal distribution of x_{t+1} is $Normal(\mu_0, \sigma_0^2)$ with $\mu_0 = \frac{\alpha_0}{1-\alpha_1}$ and $\sigma_0^2 = \frac{\sigma_0^2}{1-\alpha_1^2}$.

(3c)
```
arp1=function(a0,a1,sigma,x0,t){
  xvec<-c(x0)
  set.seed=1
  for (i in 2:t){
  xvec[i]=a0+a1*xvec[i-1]+rnorm(1,mean=0,sd=sigma)
  }
  xvec<-as.ts(xvec)

#graphs
  ts.plot(xvec)
  hist(xvec,freq = FALSE)
  stmean<-a0/(1-a1)
  stsigma<-sqrt(sigma^2/(1-(a1)^2))
  curve(dnorm(x,mean=stmean,sd=stsigma),col='blue',add=TRUE)
}
arp1(x0=0,t=1000,a0=1,a1=.95,sigma = 1)
```

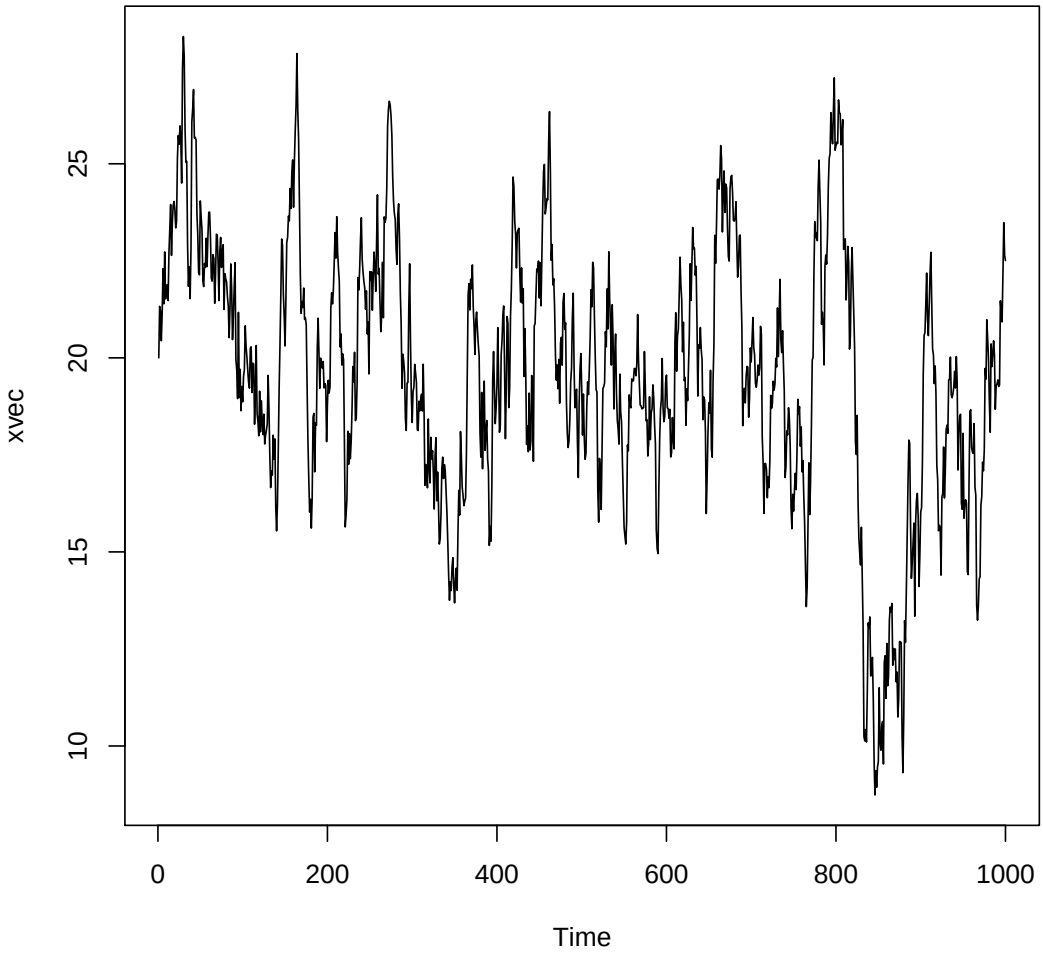

xvec
0
5
10
15
20
25
0
200
400
600
800
1000
Time

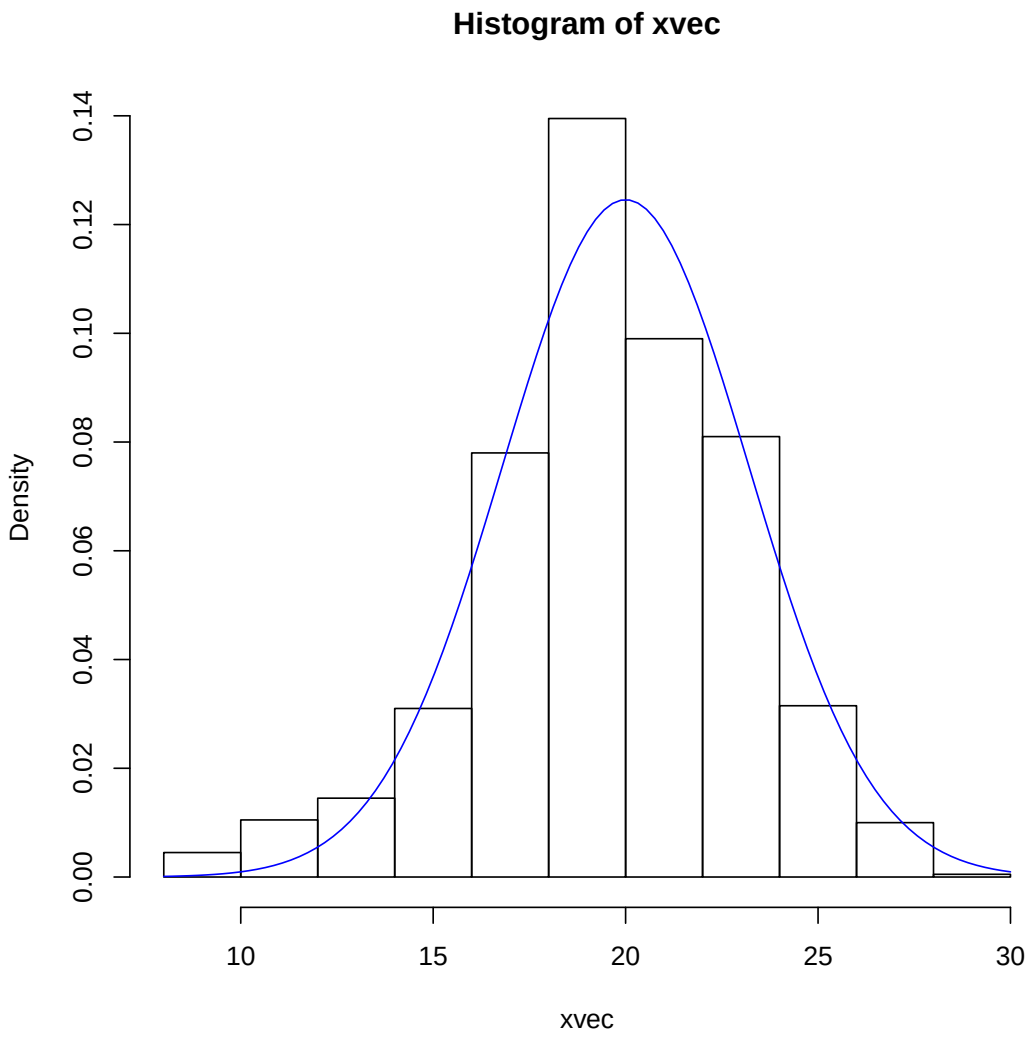


The fit of the stationary distribution on the histogram is close.

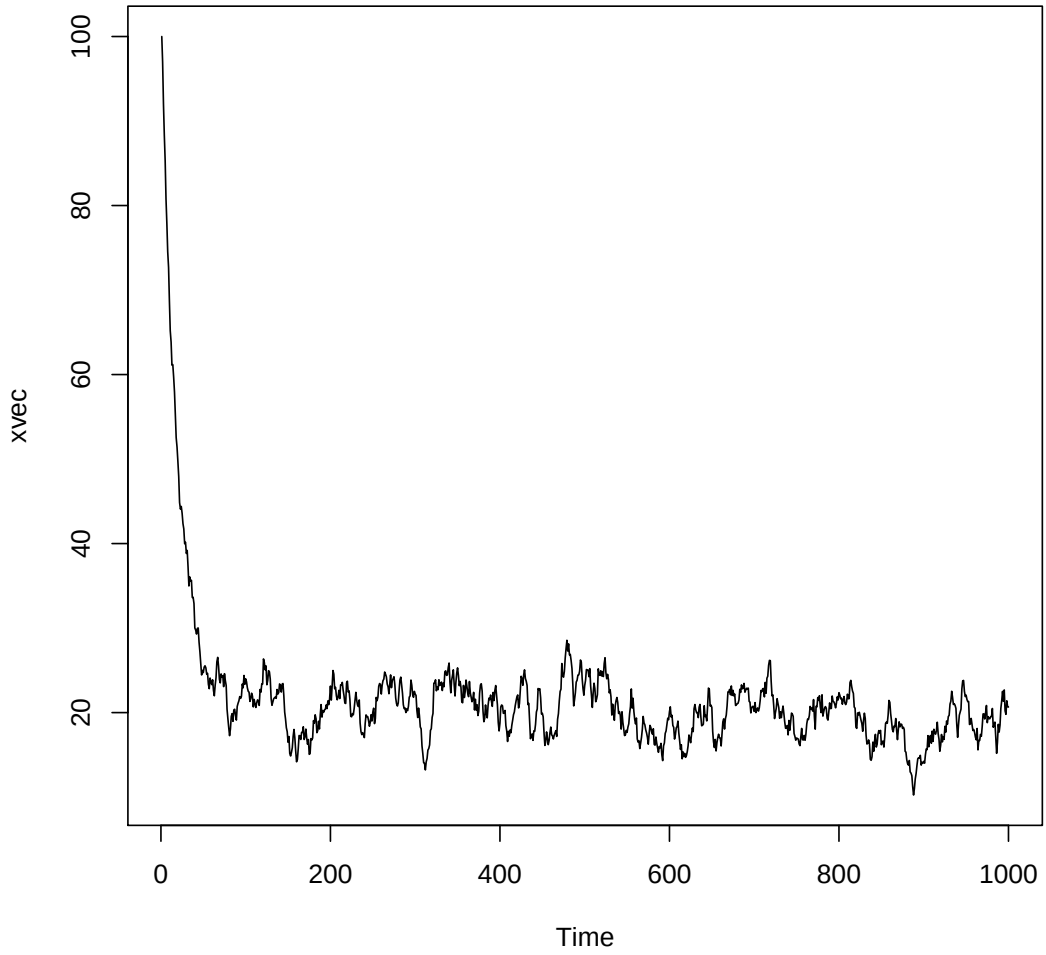
(3d) **Sensitivity to starting value.**

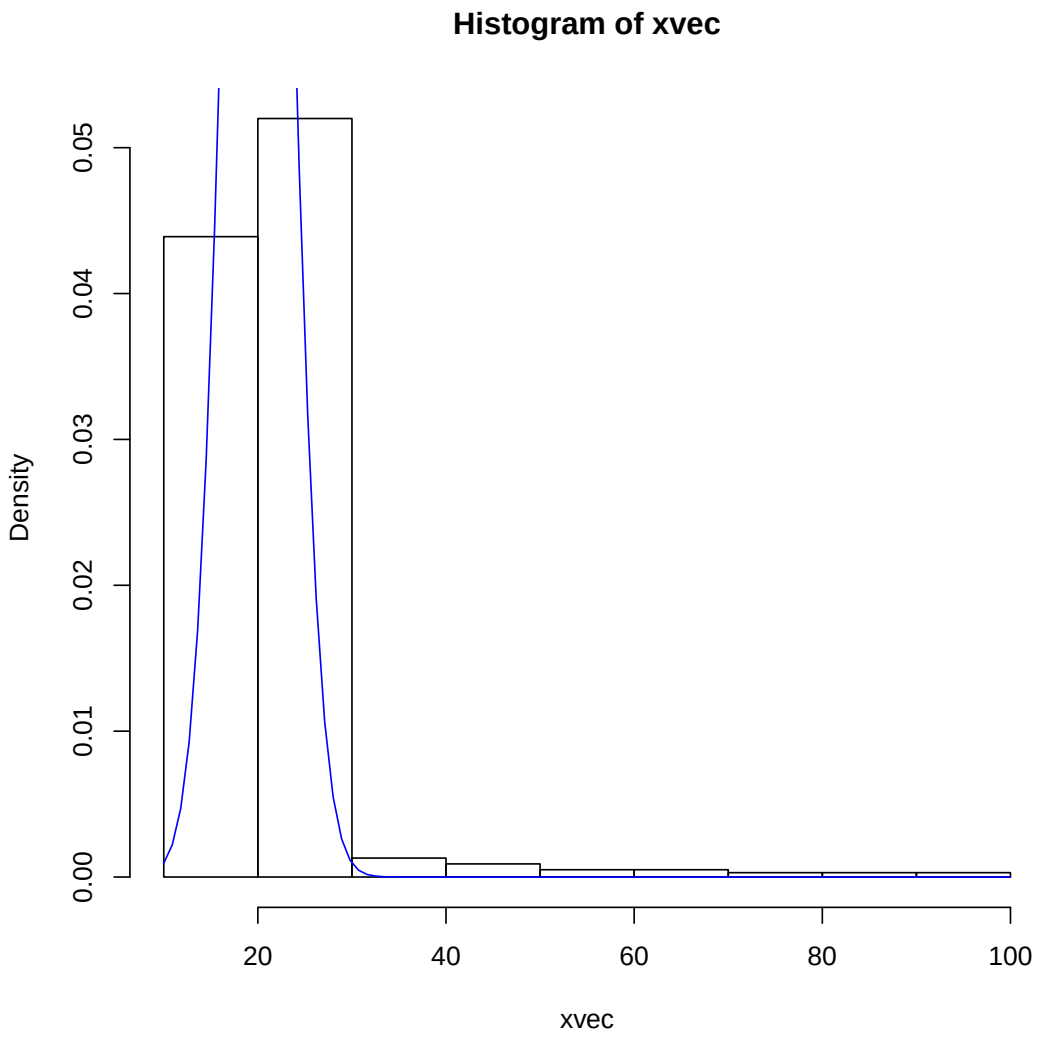
```
#x0=20
arp1(x0=20,t=1000,a0=1,a1=.95,sigma = 1)
```


xvec
10
15
20
25
0
200
400
600
800
1000
Time




```
#x0=100
arp1(x0=100,t=1000,a0=1,a1=.95,sigma = 1)
```

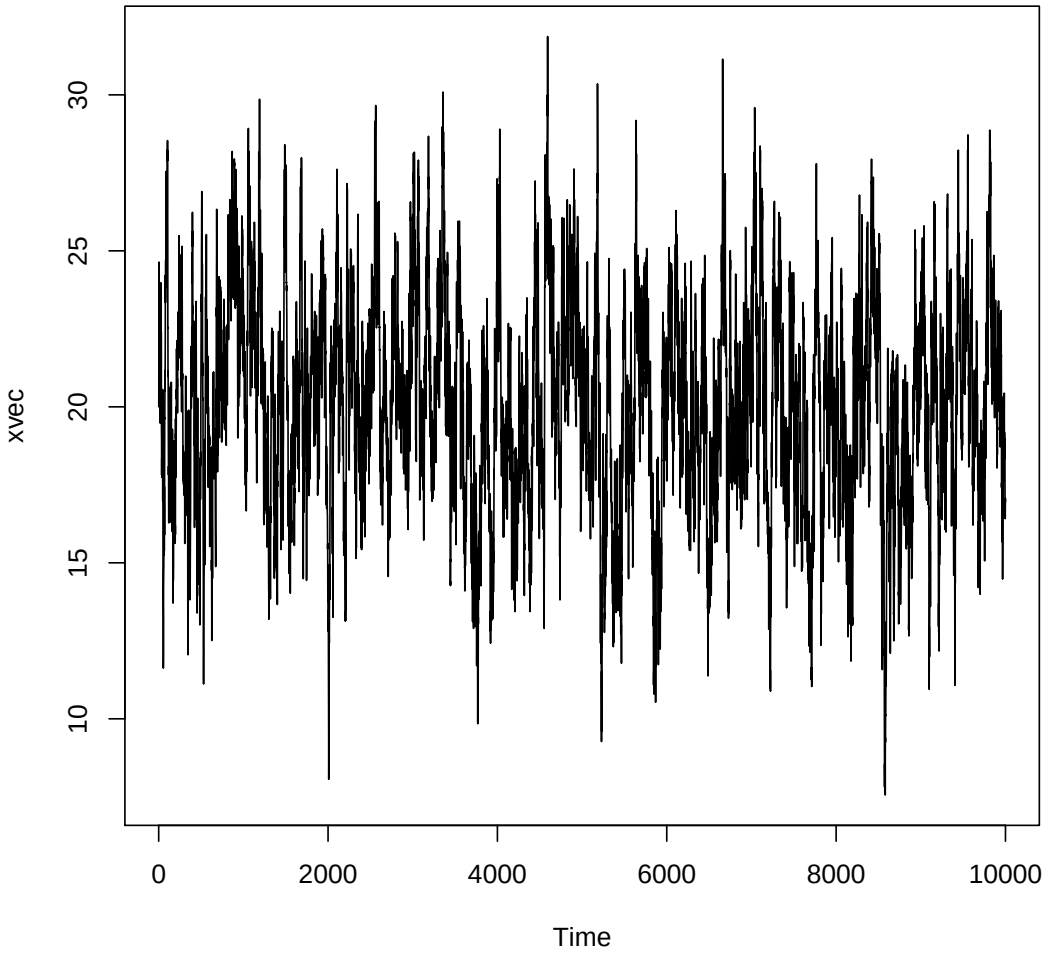

xvec
100
80
60
40
20
0
200
400
600
800
1000
Time

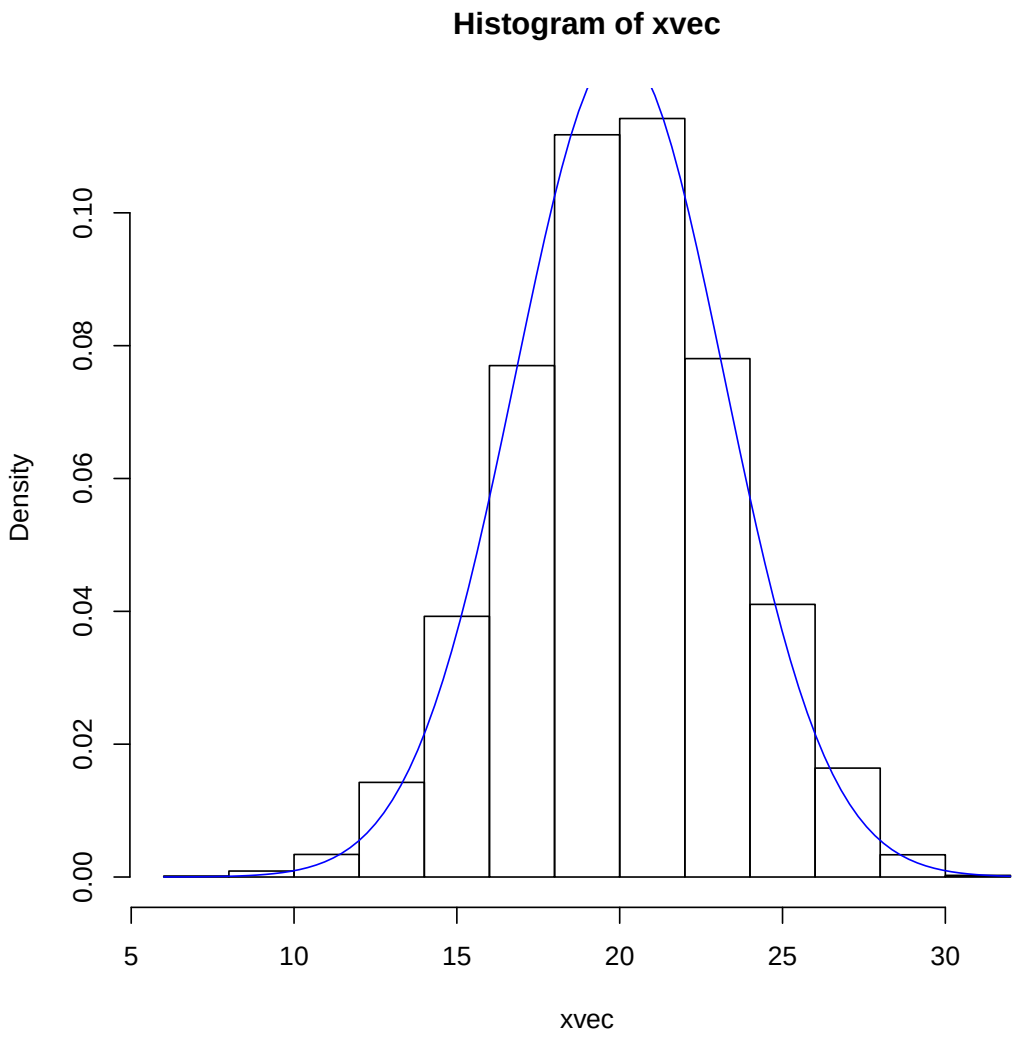


A starting value that is close to the mean of the stationary distribution quickly converges to the stationary distribution, wheras a starting value that is far from the mean of the stationary distribution appears to take longer to reach the stationary distribution.

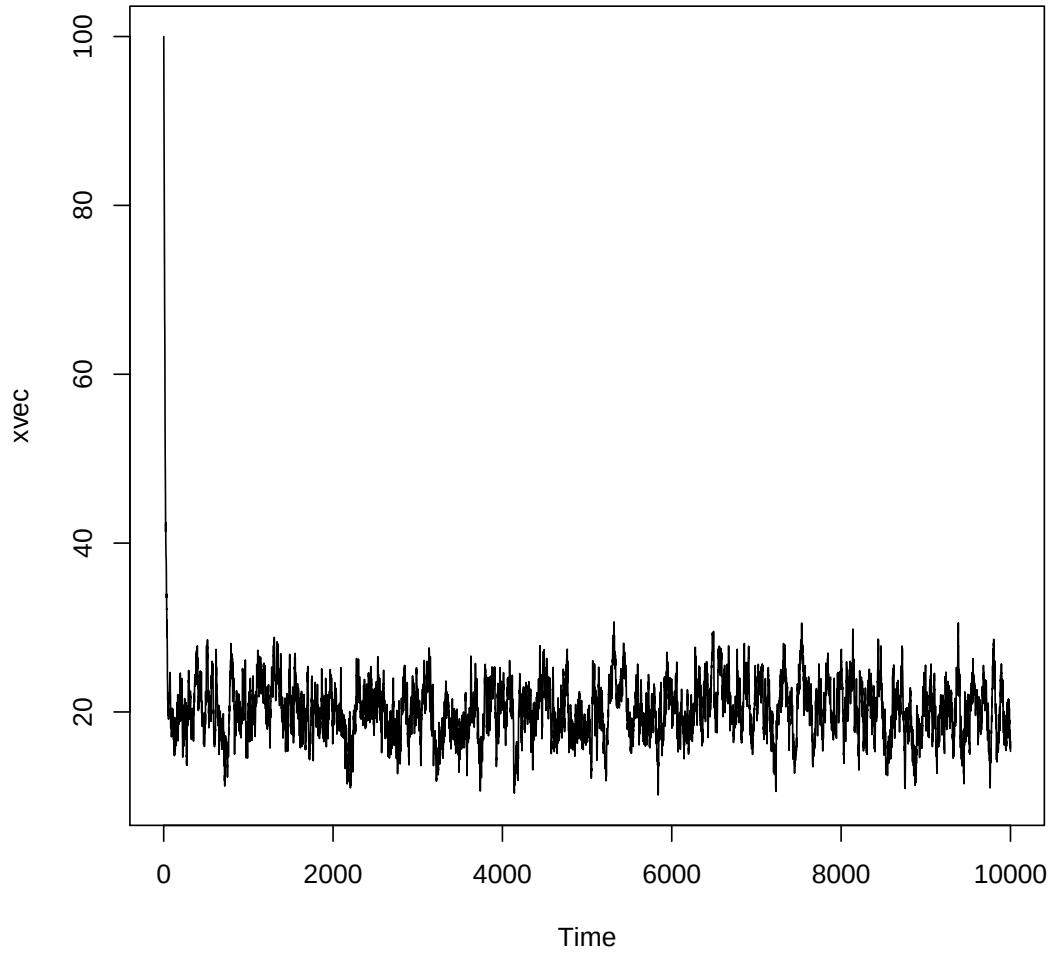
(3e) **Decreasing sensitivity to starting value with longer series.**

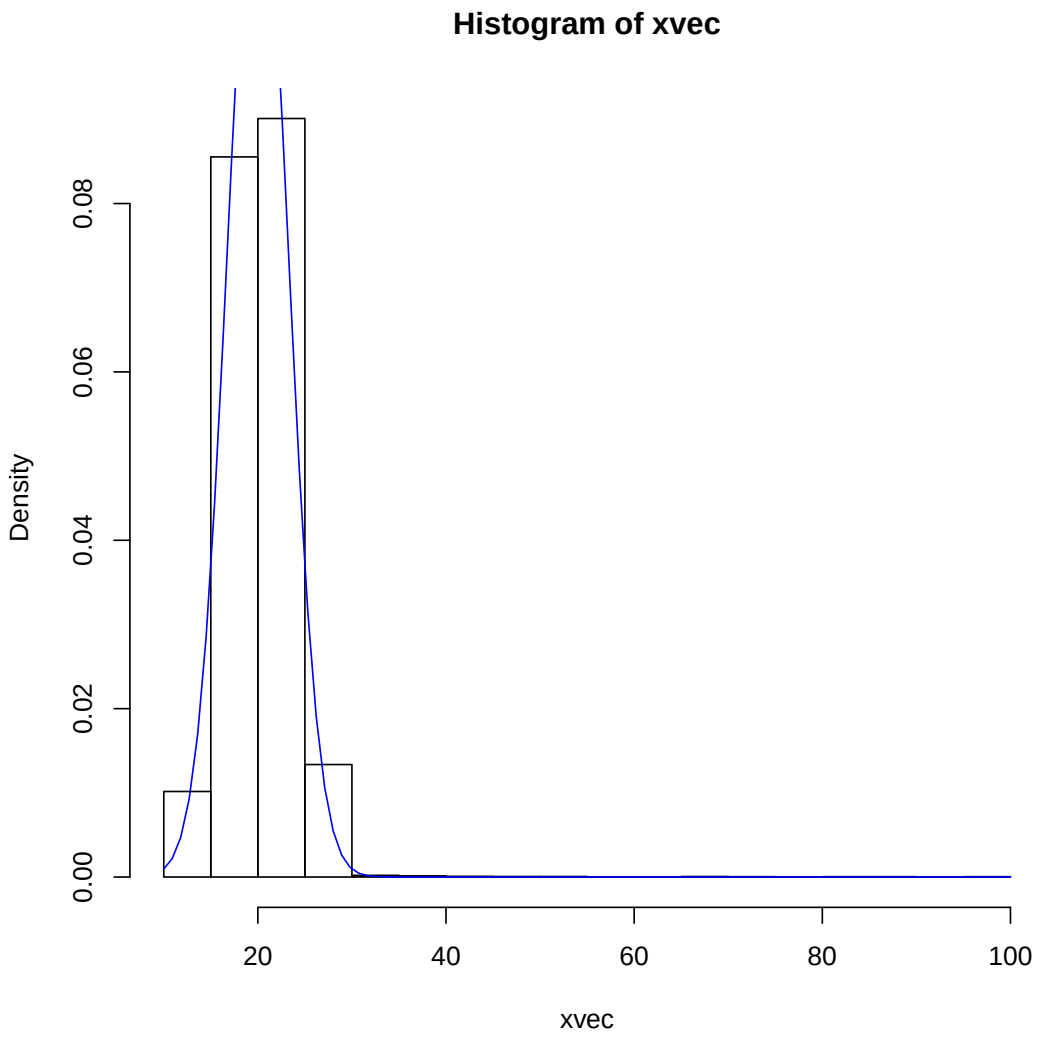
```
#x0=20
arp1(x0=20,t=10000,a0=1,a1=.95,sigma = 1)
```


xvec
10
15
20
25
30
0
2000
4000
6000
8000
10000
Time




```
#x0=100
arp1(x0=100,t=10000,a0=1,a1=.95,sigma = 1)
```


xvec
100
80
60
40
20
0
2000
4000
6000
8000
10000
Time

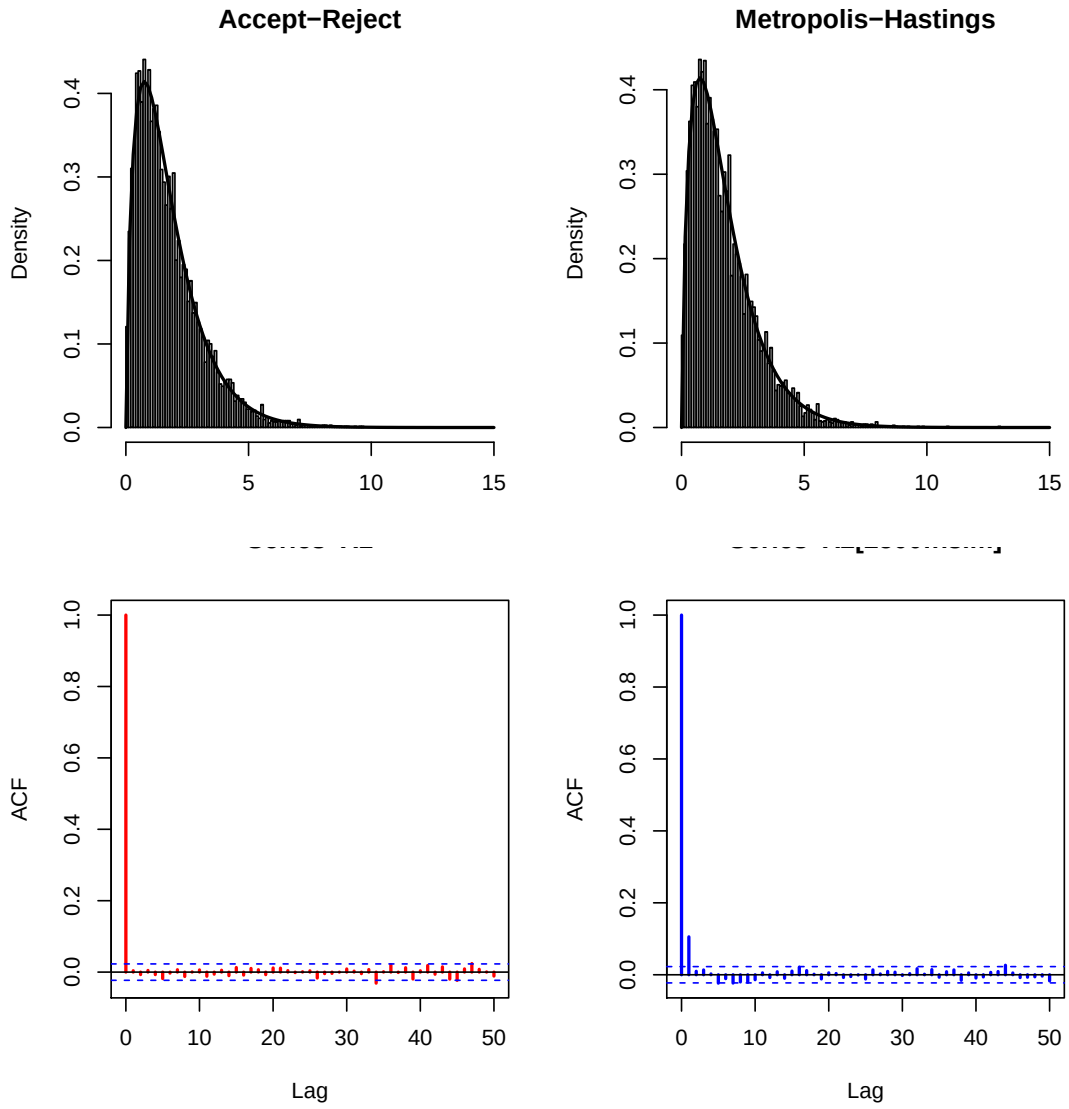


Compared with 3d, the effect of the starting value is a lot less pronounced.

(3f) Excluding path values during an initial burn-in period can be valuable when the initial value is very far from mean of the stationary distribution. From this data, it seems that the length of an effective burn-in period depends on the starting value. Choosing initial values that are close to the mean of the theoretical stationary distribution if one can calculate it may help mitigate the cost of a burn-in period. As for selecting a burn-in period, it may be hard to know when one has reached the stationary distribution, especially if one doesn't have good ex-ante information of the the stationary distribution; if one does, then discarding initial values that are very far from the stationary distribution may be useful.

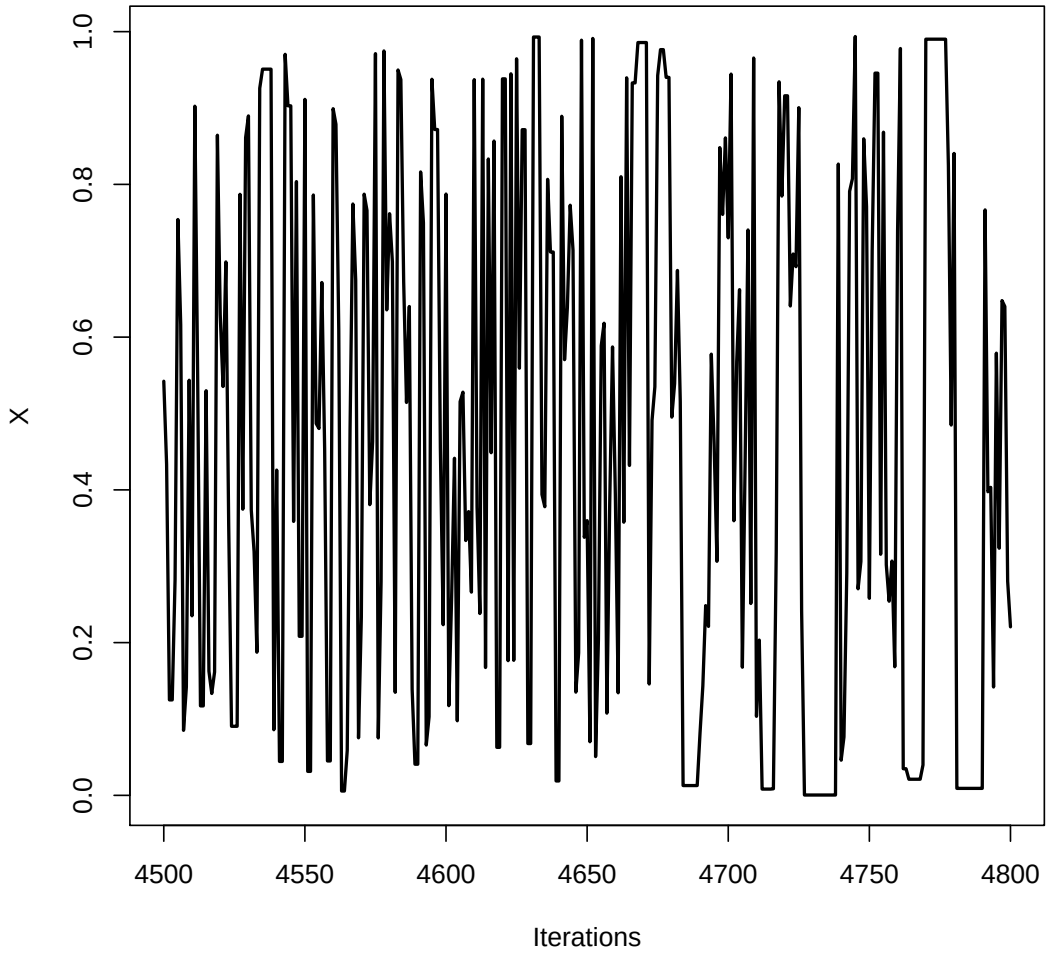
(4b)
```
library("mcsm")
## Loading required package:  MASS
## Loading required package:  coda

a=1.75;nsim=10000;
```

```
X1=X2=array(0,dim=c(nsim,1))              #AR & MH
X1[1]=X2[1]=rgamma(1,a,rate=1)                  #initialize the chain
for (i in 2:nsim){
    Y=rgamma(1,floor(a),rate=floor(a)/a)    #candidate
    rhoAR=(exp(1)*Y*exp(-Y/a)/a)^(a-floor(a))
    rhoMH=(dgamma(Y,a,rate=1)/dgamma(X2[i-1],a,rate=1))/(dgamma(Y,floor(a),
        rate=floor(a)/a)/dgamma(X2[i-1],floor(a),rate=floor(a)/a))
    rhoMH=min(rhoMH,1)
    X1[i]=Y*(runif(1)<rhoAR)                        #accepted values
    X2[i]=X2[i-1] + (Y-X2[i-1])*(runif(1)<rhoMH)
}
X1=X1[X1!=0]                      #The AR sample
par(mfrow=c(2,2),mar=c(4,4,2,2))
hist(X1,col="grey",nclas=125,freq=FALSE,xlab="",main="Accept-Reject",xlim=c(0,15))
curve(dgamma(x, a, rate=1),lwd=2,add=TRUE)
hist(X2[2500:nsim],nclas=125,col="grey",freq=FALSE,xlab="",main="Metropolis-Hastings"
curve(dgamma(x, a, rate=1),lwd=2,add=TRUE)
acf(X1,lag.max=50,lwd=2,col="red")               #Accept-Reject
acf(X2[2500:nsim],lag.max=50,lwd=2,col="blue")   #Metropolis-Hastings
```

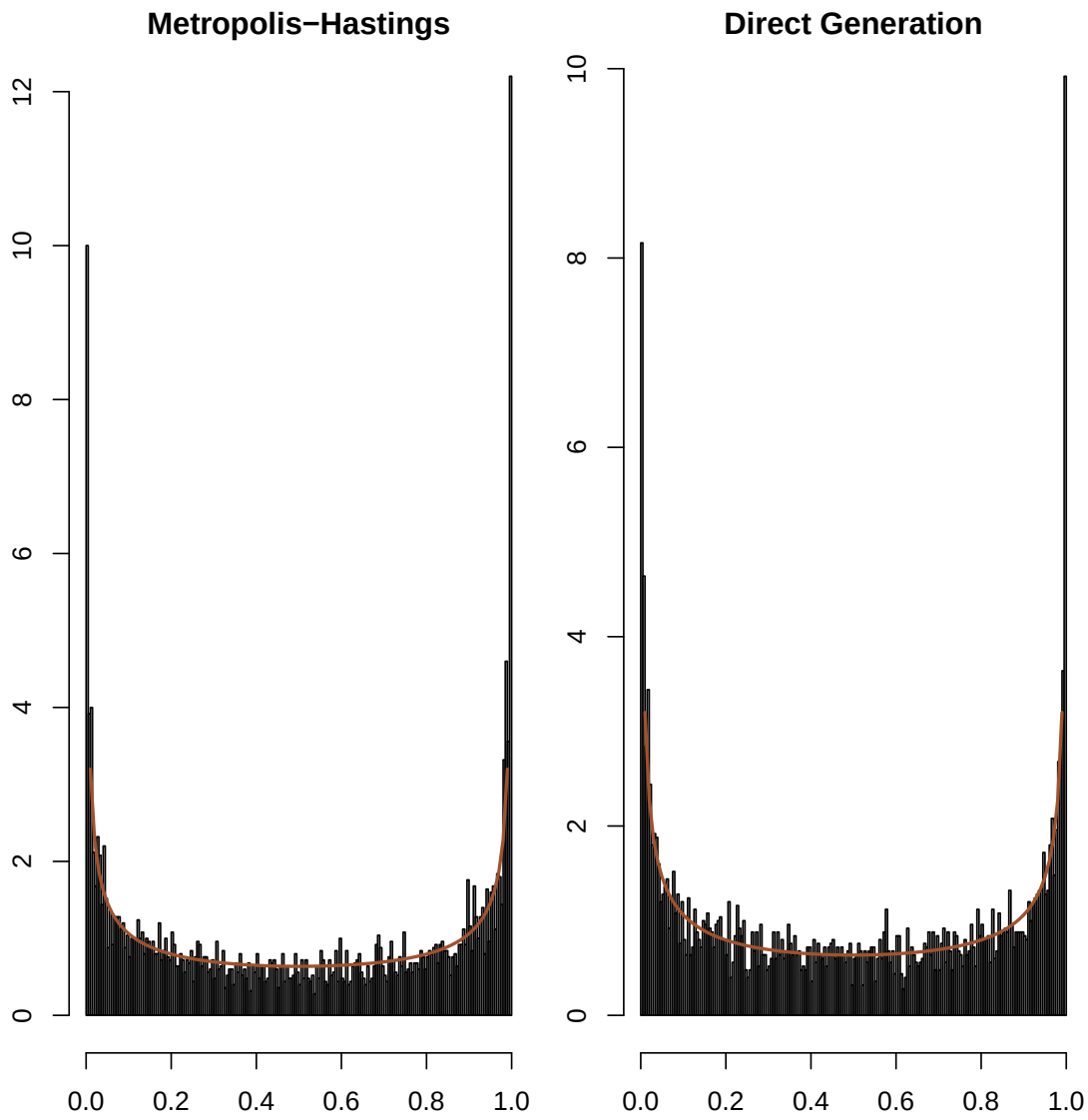


(4b)

```
library("mcsm")
a=.5; b=.5; c=2.669 # initial values
nsim=5000
X=rep(runif(1),nsim)  # initialize the chain
for (i in 2:nsim){
   Y=runif(1)
   rho=dbeta(Y,a,b)/dbeta(X[i-1],a,b)
   X[i]=X[i-1] + (Y-X[i-1])*(runif(1)<rho)
   }
#X11(h=3.5);m
plot(4500:4800,X[4500:4800],ty="l",lwd=2,xlab="Iterations",ylab="X")
```



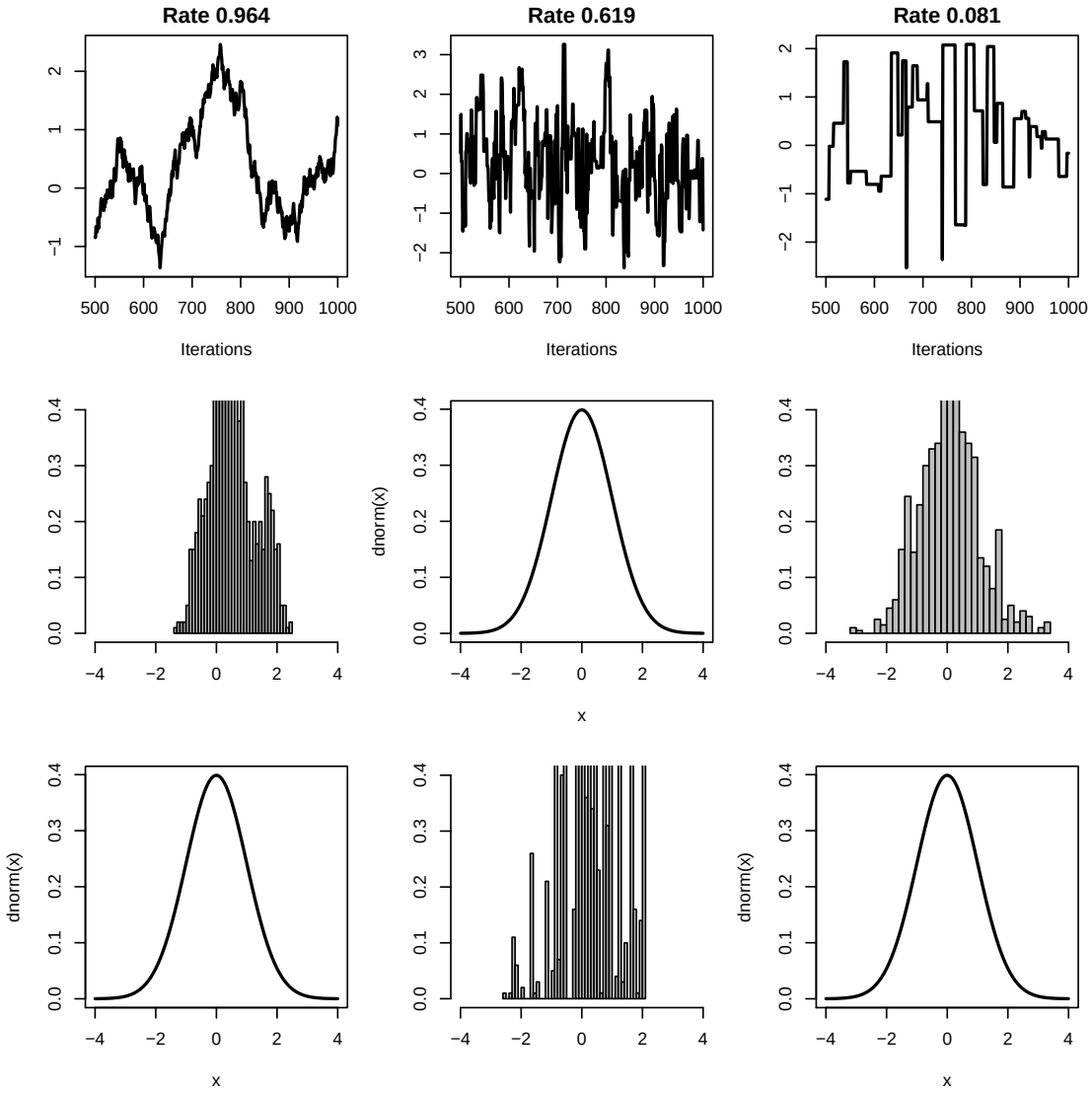

```
#S=readline(prompt="Type  <Return>   to continue : ")
#dev.off()
ks.test(jitter(X),rbeta(5000,a,b))

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  jitter(X) and rbeta(5000, a, b)
## D = 0.0334, p-value = 0.007562
## alternative hypothesis: two-sided

par(mfrow=c(1,2),mar=c(2,2,1,1))
hist(X,nclass=150,col="grey",main="Metropolis-Hastings",fre=FALSE)
curve(dbeta(x,a,b),col="sienna",lwd=2,add=TRUE)
hist(rbeta(5000,a,b),nclass=150,col="grey",main="Direct Generation",fre=FALSE)
curve(dbeta(x,a,b),col="sienna",lwd=2,add=TRUE)
```

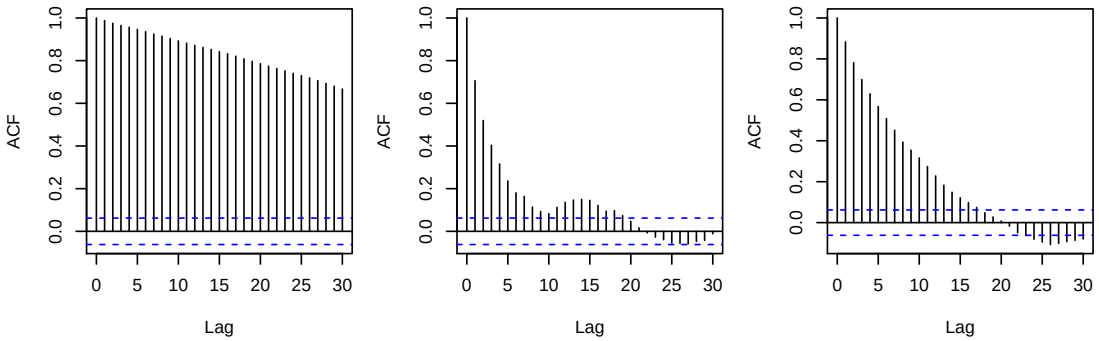


(4c) For the $Beta(a = b = 0.5)$ distribution in (b) it is not possible to apply the Accept-Reject method because the density of the $Beta(a = b = 0.5)$ distribution is unbounded. To apply the Accept-Reject method to a disribution, one must be able to bound it with a box (or with another known, bounded distribution as we saw in class).

(4d)
```
library("mcsm")
    hastings2<-hastings
    # Then edit hastings2 directly
   # fix(hastings2) # change the vector a
    # Execute the revised function
    hastings2=function (nsim = 10^3)
{
    a = c(0.2, 2, 20)
    na = length(a)
    x = array(0, c(na, nsim))
    for (i in 1:na) {
```

```
        acc = 0
        for (j in 2:nsim) {
            y <- x[i, (j - 1)] + runif(1, min = -a[i], max = a[i])
            r = min(exp(-0.5 * ((y^2) - (x[i, (j - 1)]^2))),
                1)
            u <- runif(1)
            acc = acc + (u < r)
            x[i, j] <- y * (u < r) + x[i, (j - 1)] * (u > r)
        }
    }
    par(mfrow = c(3, na), mar = c(4, 4, 2, 1))
    for (i in 1:na) plot((nsim - 500):nsim, x[i, (nsim - 500):nsim],
        ty = "l", lwd = 2, xlab = "Iterations", ylab = "", main = paste("Rate",
            (length(unique(x[i, ]))/nsim), sep = " "))
    for (i in 1:na) {
        hist(x[i, ], freq = F, xlim = c(-4, 4), ylim = c(0, 0.4),
            col = "grey", ylab = "", xlab = "", breaks = 35,
            main = "")
        curve(dnorm(x), lwd = 2, add = T)
    }
    for (i in 1:na) acf(x[i, ], main = "")
}

    hastings2()
```


Rate 0.964
Rate 0.619
Rate 0.081
Iterations
Iterations
Iterations
dnorm(x)
x
x
dnorm(x)
dnorm(x)



In this problem, the paths for the simulated Markov-Chain are generated using a random walk where the step-sizes are a symmetric uniform distribution of width 2a. The acceptance rate decreases with a, because with a larger a, more proposed moves may be too large and will be rejected, so the chain won't move as much; conversely, if the a is too small, most proposed moves will be accepted, but it will explore the parameter space much slower. In the extreme cases, if a is too small, it will accept all moves, and if a is too large, it will reject all moves.

It is not always better to always use a candidate distribution with a higher acceptance rate. It may be the case that a high acceptance rate indicates too small steps, which may mean that the parameter space has not been fully explored (e.g. parts of f's domain with small values). Although it should be noted that a low acceptance rate does not necessarily mean that it has fully explored f's domain; the random walk may reach borders along the support of f, or low probability regions under f, so it can miss an important but isolated peak of the pdf. Hence, one should calibrate a to achieve efficient convergence with reasonable movements and mixing.

The ACF function has higher values the smaller the value of a, the half-width of the

uniform candidate density for the aforementioned reasons: very small steps often do not fully explore the parameter space, which can lead to unreasonably high acceptance rates.