Series 5

1. **Gibbs sampler**

The goal of this exercise is to use the Gibbs sampler to simulate random variables from a bivariate normal distribution[1]

$$(X_1, X_2) \sim N\left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}\right), \quad -1 < \rho < 1.$$

a) Use the Cholesky decomposition to simulate $N = 1000$ samples from this bivariate normal distribution. Visualize your samples (X_1, X_2) in a two-dimensional scatter plot.

b) Construct a Gibbs sampler to simulate $N = 1000$ samples from this bivariate normal distribution. Use the point $(0, 0)$ as deterministic initial value.

– Plot the simulated values X_1 (or X_2) versus the iteration number. This produces so-called trace plots.

– Plot the autocorrelation function (ACF) for one of the simulated values (e.g. X_1).

– Visualize your samples (X_1, X_2) in a two-dimensional scatter plot and create a Q-Q plot to verify that the marginal distributions are indeed normal.

– Investigate the impact of varying ρ on the autocorrelation in the samples. At least, try $\rho = 0.8$, $\rho = 0.95$, and $\rho = 0.99$. What do you observe?

– Investigate the impact of varying the initial value. At least, also try $(20, 20)$ as initial value with $\rho = 0.95$ and $\rho = 0.99$. Visually determine the approximate length of the burn-in phase, i.e., the number of samples to be discarded before the stationary distribution is reached.

Hints:

- You can use the `R` function `chol` to calculate the Cholesky decomposition. This function returns a matrix R such that $R^T R = \Sigma$ where Σ is the original matrix. See the corresponding help entry (`?chol`) for more information about how `R` calculates the Cholesky decomposition.

[1] In practice, one would not use MCMC to sample from a bivariate normal distribution. Here, we use this example to learn the basic principles and understand how the algorithms work in various situations.

Solution

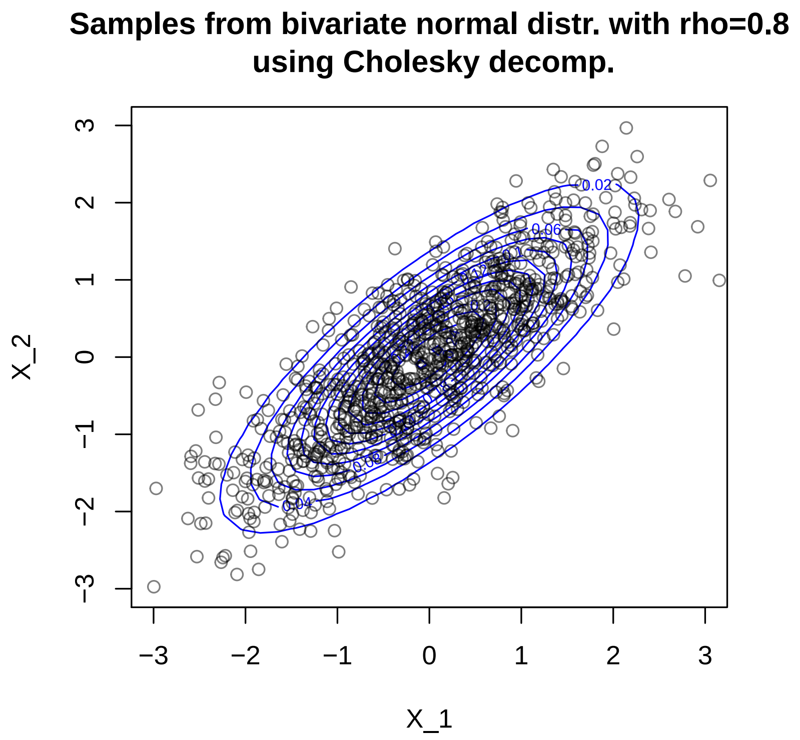
a) For simulation of the bivariate normal distribution using the Cholesky decomposition, we first decompose the matrix $\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ to obtain a matrix R such that $R^T R = \Sigma$. We then set

$$(X_1, X_2) = R^T Z, \quad Z = (Z_1, Z_2)^T,$$

where Z_1 and Z_2 are independent standard normal variables.

See the code below for an implementation in R. The plot shows contour lines of the bivariate normal density and the samples from the corresponding distribution generated using the Cholesky decomposition.

```
rho=0.8
S=matrix(c(1,rho,rho,1),ncol=2)
R=chol(S)##Calculate Cholesky decomposition
t(R)%*%R##Check that R is indeed a Cholesky factor

##      [,1] [,2]
## [1,]  1.0  0.8
## [2,]  0.8  1.0

N=1000
set.seed(1)
sn=matrix(rnorm(2*N),nrow=2)##Simulate iid N(0,1) variables
sim_chol=t(R)%*%sn

f.contour.biv.norm <- function(rho,...){
  ## Purpose: Plotting the density of bivariate normal distribution
  x <- seq(-3,3,length=32)
  S=matrix(c(1,rho,rho,1),ncol=2)
  Si=solve(S)
  cst=1/2/pi/sqrt(det(S))
  z=matrix(NA,ncol=32,nrow=32)
  for(i in 1:32){
    for(j in 1:32){
      xx=c(x[i],x[j])
      z[i,j]=cst*exp(-0.5*t(xx)%*%Si%*%xx)
    }
  }
  contour(x,x,z,nlevels=10,col="blue",...)
}
f.contour.biv.norm(rho,xlab="X_1",ylab="X_2",
     main=paste("Samples from bivariate normal distr. with rho=",
      rho, "\n using Cholesky decomp.",sep=""))
col=rgb(0,0,0,alpha=0.5)##Color of points used for plotting
points(sim_chol[1,],sim_chol[2,],col=col)
```



b) We choose a deterministic initial value (x_0, y_0) and simulate iteratively

$$X_{1t+1} \sim N(\rho x_{2t}, 1-\rho^2),$$

and

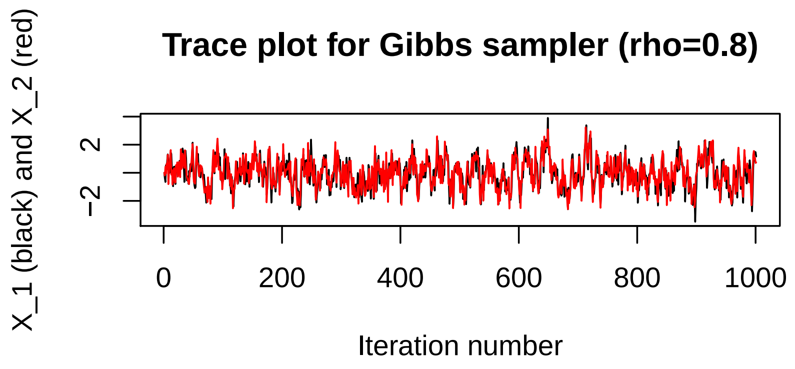
$$Y_{2t+1} \sim N(\rho x_{1t+1}, 1-\rho^2).$$

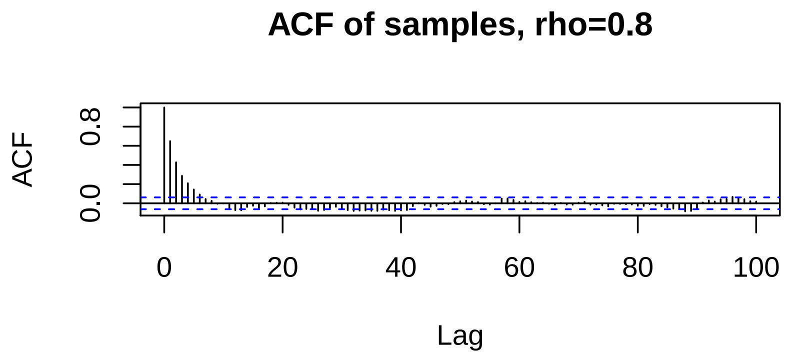
See the code below for an implementation of the Gibbs sampler in R.

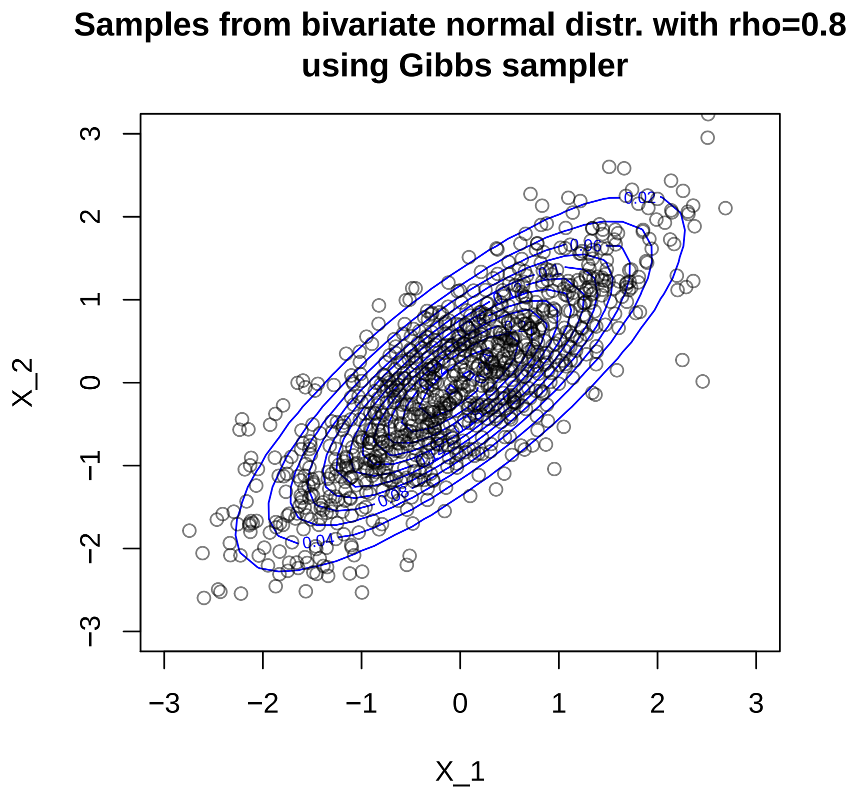
The variance of the conditional densities is $1-\rho^2$. This means that with increasing ρ, the conditional densities are more and more narrowly concentrated around the current values, and only small steps can be made. Since the Gibbs sampler can only move in the direction of one component, the joint distribution is sampled slowly, i.e., there is autocorrelation. This behavior is also apparent in the trace plots and the ACFs below.

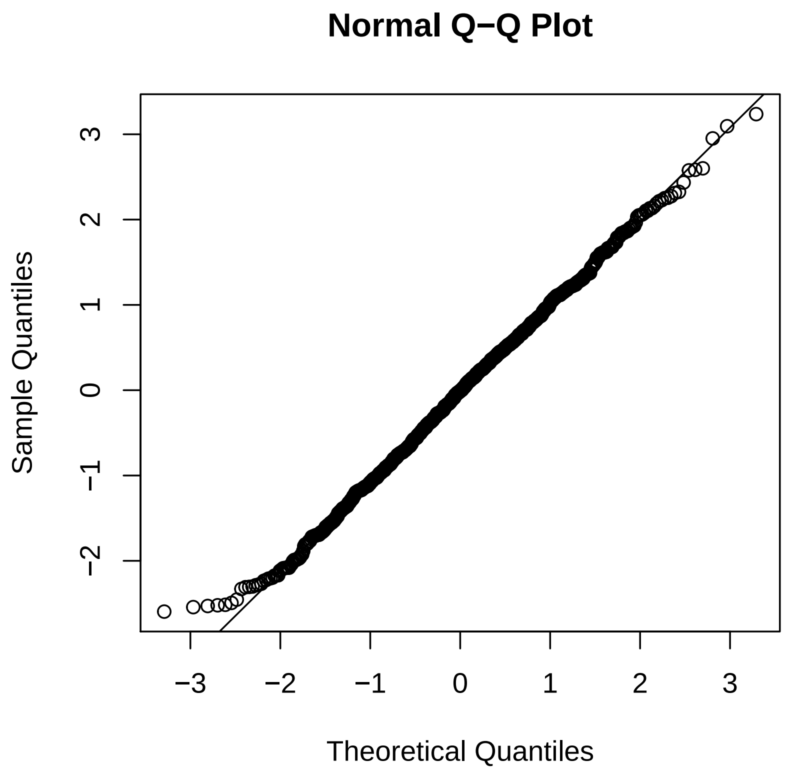
In the last two examples shown below, we use starting values that are far away from the center of the distribution. The trace plots show how the Gibbs sampler converges to the stationary distribution after a burn-in period. Using $(20, 20)$ as initial value and $\rho = 0.99$, the stationary distribution is reached after approximatetly 150-200 iterations. To be on the safe side, we would exlcude at least the first 200 simulated values.

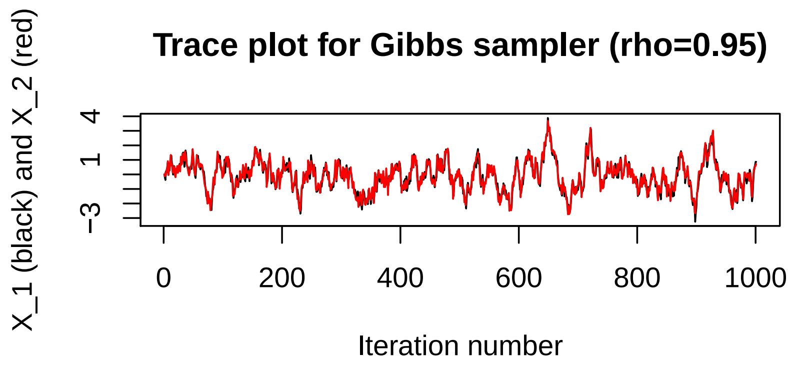
```
sim.gibbs.biv.normal=function(rho,N,sv=c(0,0),plotSamples=TRUE,burnin=NULL,...){
  sim_gibbs=matrix(NA,nrow=2,ncol=N+1)
  sim_gibbs[,1]=sv##Deterministic initial value
  set.seed(1)
  for(i in 1:N){
    sim_gibbs[1,i+1]=rnorm(1,mean=rho*sim_gibbs[2,i],sd=sqrt(1-rho^2))
    sim_gibbs[2,i+1]=rnorm(1,mean=rho*sim_gibbs[1,i+1],sd=sqrt(1-rho^2))
  }
  ##Plot results
  par(mfrow=c(2,1))
  plot(sim_gibbs[1,],type="l",
       main=paste("Trace plot for Gibbs sampler (rho=",rho,")",sep=""),
       xlab="Iteration number",ylab="X_1 (black) and X_2 (red)")
  lines(sim_gibbs[2,],col=2)
  ##Remove samples from burn-in phase (if there is a burn-in phase)
  if(!is.null(burnin)) sim_gibbs=sim_gibbs[,-(1:burnin)]
  acf(sim_gibbs[1,],main=paste("ACF of samples, rho=",rho,sep=""),lag.max=100)

  if(plotSamples){
    par(mfrow=c(1,1))
    f.contour.biv.norm(rho,xlab="X_1",ylab="X_2",
        main=paste("Samples from bivariate normal distr. with rho=",rho,
                   "\n using Gibbs sampler",sep=""))
    points(sim_gibbs[1,],sim_gibbs[2,],...)

    qqnorm(sim_gibbs[2,])
    qqline(sim_gibbs[2,])
  }
}

sim.gibbs.biv.normal(rho=0.8,N=N,col=col)
sim.gibbs.biv.normal(rho=0.95,N=N,col=col,plotSamples=FALSE)
sim.gibbs.biv.normal(rho=0.99,N=N,col=col,plotSamples=FALSE)
sim.gibbs.biv.normal(rho=0.95,N=N,col=col,sv=c(20,20),plotSamples=FALSE,burnin=50)
sim.gibbs.biv.normal(rho=0.99,N=N,col=col,sv=c(20,20),plotSamples=TRUE,burnin=200)
```

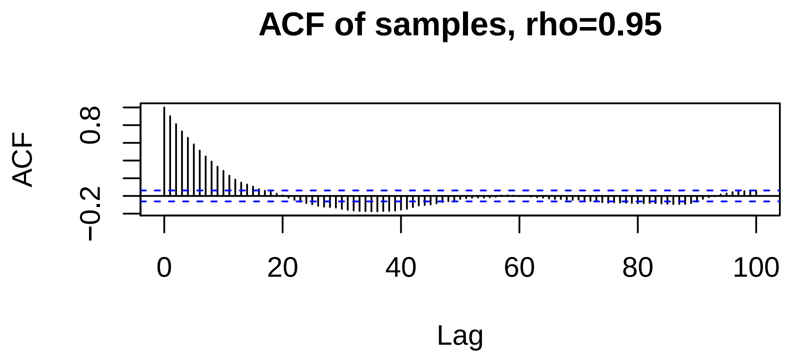

Trace plot for Gibbs sampler (rho=0.8)
X_1 (black) and X_2 (red)
−2
2
0
200
400
600
800
1000
Iteration number

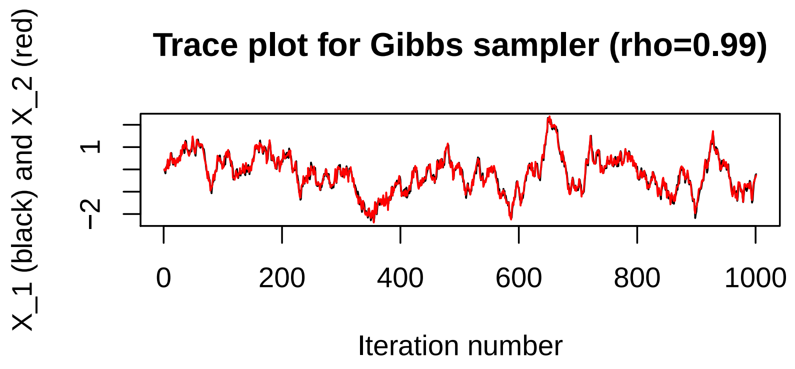

ACF of samples, rho=0.8
ACF
0.0
0.8
0
20
40
60
80
100
Lag

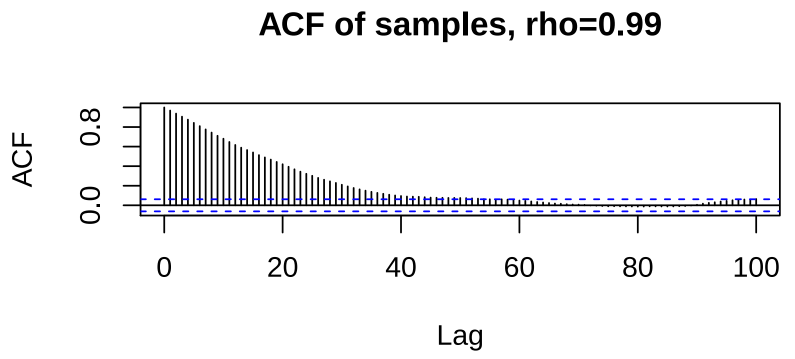

Samples from bivariate normal distr. with rho=0.8
using Gibbs sampler
X_1
X_2
−3
−2
−1
0
1
2
3
0.02
0.04
0.06
0.08

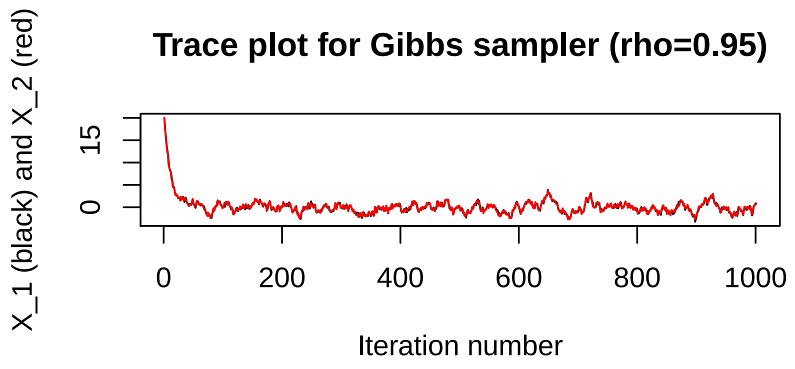

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
3

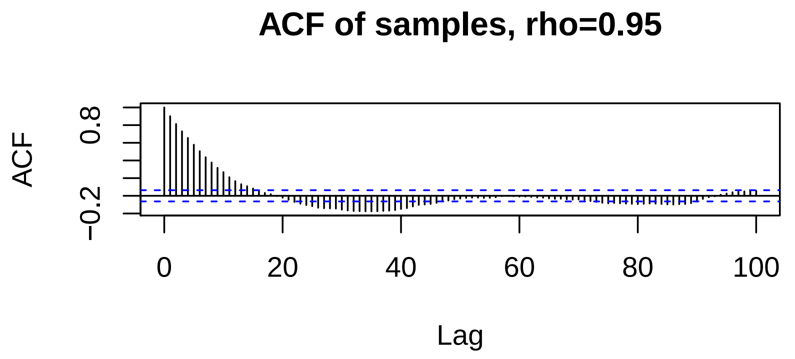

Trace plot for Gibbs sampler (rho=0.95)
X_1 (black) and X_2 (red)
4
1
−3
0
200
400
600
800
1000
Iteration number

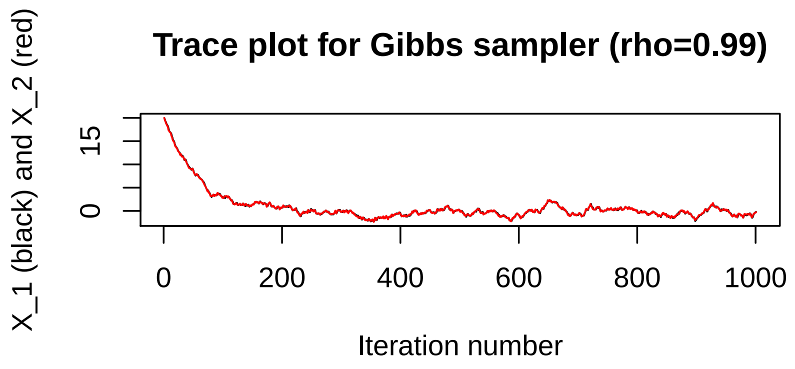

ACF of samples, rho=0.95
ACF
0.8
−0.2
0
20
40
60
80
100
Lag

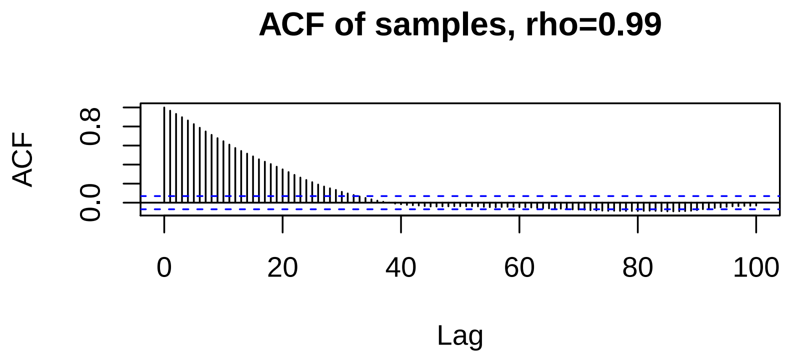

Trace plot for Gibbs sampler (rho=0.99)
X_1 (black) and X_2 (red)
−2
1
0
200
400
600
800
1000
Iteration number

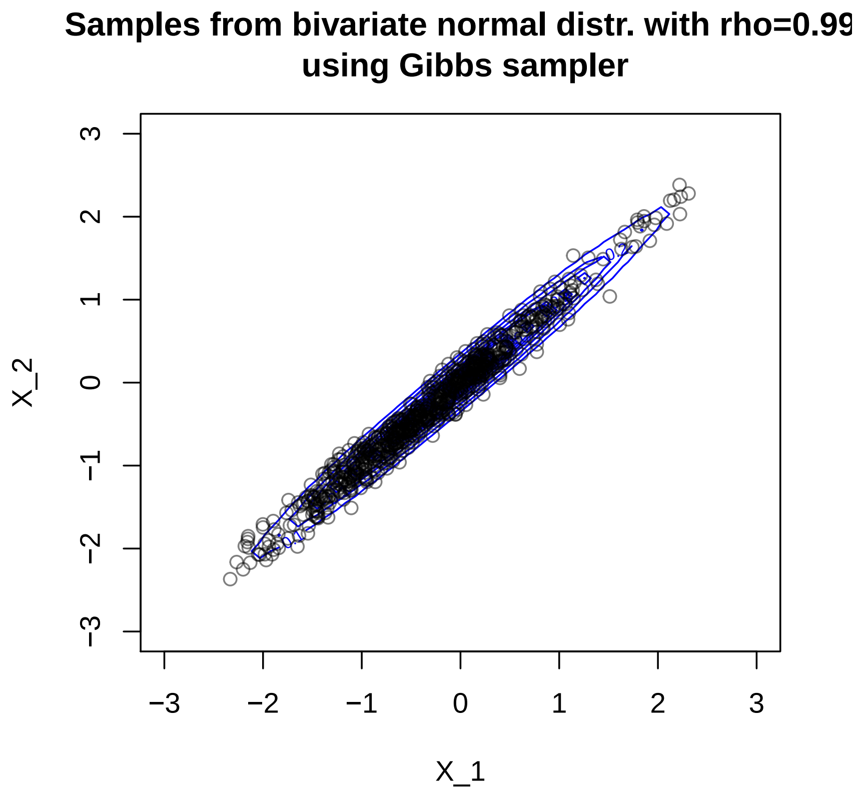

ACF of samples, rho=0.99
ACF
0.0
0.8
0
20
40
60
80
100
Lag

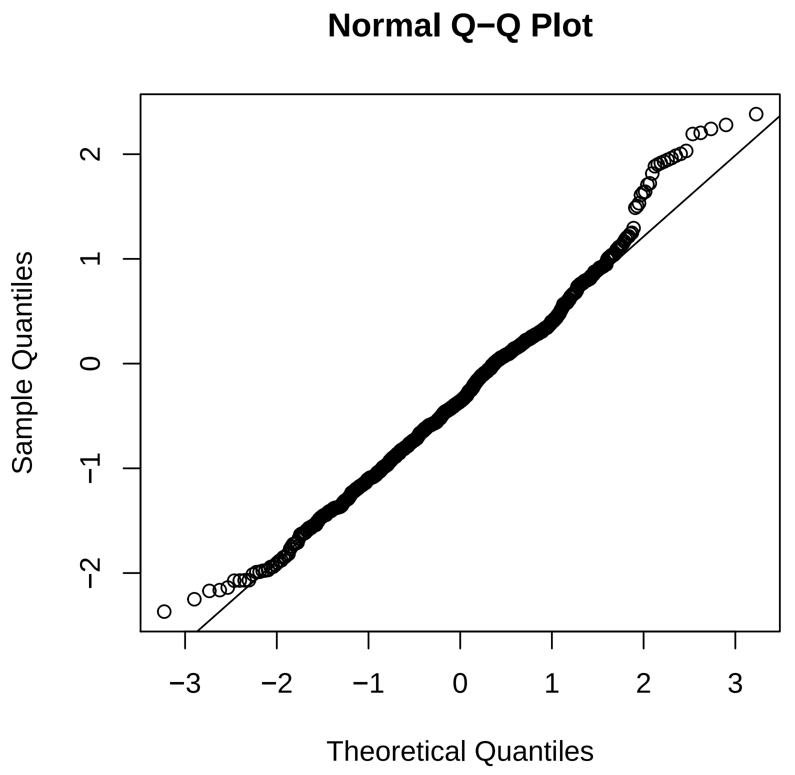

Trace plot for Gibbs sampler (rho=0.95)
X_1 (black) and X_2 (red)
15
0
0
200
400
600
800
1000
Iteration number


ACF of samples, rho=0.95
ACF
0.8
−0.2
0
20
40
60
80
100
Lag


Trace plot for Gibbs sampler (rho=0.99)
X_1 (black) and X_2 (red)
15
0
0
200
400
600
800
1000
Iteration number


ACF of samples, rho=0.99
ACF
0.8
0.0
0
20
40
60
80
100
Lag


Samples from bivariate normal distr. with rho=0.99
using Gibbs sampler
X_1
X_2
−3
−2
−1
0
1
2
3


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−2
−1
0
1
2
−3
−2
−1
0
1
2
3

2. **Random walk Metropolis algorithm**

   The goal of this exercise is to use the random walk Metropolis (RWM) algorithm to simulate random variables from the above bivariate normal distribution.

   a) Use the random walk Metropolis algorithm with a normal proposal densitiy with proposal covariance matrix

   $$\sigma^2 \cdot I_2$$

   to simulate $N = 1000$ samples from the bivariate normal distribution as target distribution.

   - Plot the simulated values X_1 (or X_2) versus the iteration number.
   - Calculate the percentage of the accepted values.
   - Plot the autocorrelation function (ACF) for one of the simulated values (e.g. X_1).
   - Visualize your samples (X_1, X_2) in a two-dimensional scatter plot and create a Q-Q plot to verify that the marginal distributions are indeed normal.

   b) Investigate the impact of varying ρ on the acceptance percentage and on the autocorrelation in the samples. At least, try $\rho = 0.5$, $\rho = 0.8$, $\rho = 0.95$, and $\rho = 0.99$. What do you observe?

   c) Investigate the impact of varying the proposal standard deviation σ on the acceptance rate and on the autocorrelation in the samples. Use $\sigma = 1$ as default choice and also try $\sigma = 0.2$ and $\sigma = 5$.

   d) Investigate the impact of varying the initial value. At least, try $(20, 20)$ as initial value with $\rho = 0.95$ and $\rho = 0.99$. Visually determine the approximate length of the burn-in phase, i.e., the number of samples to be discarded before the stationary distribution is reached.

Solution

a) We choose a deterministic initial value (x_0, y_0) and iteratively propose a new value y according to

$$Y \sim N(x_t, I_2).$$

This y is accepted with probability

$$a(x_t, y) = \min\left(1, \exp\left(-0.5(y^T\Sigma^{-1}y - x_t^T\Sigma^{-1}x_t)\right)\right).$$

I.e., we simulate $U \sim uniform(0,1)$ and if $U \leq a(x_t, y)$, we accept y. If y is accepted, we set $x_{t+1} = y$, otherwise we set $x_{t+1} = x_t$.
See the code below for a specific implementation of the random walk Metropolis algorithm in R.

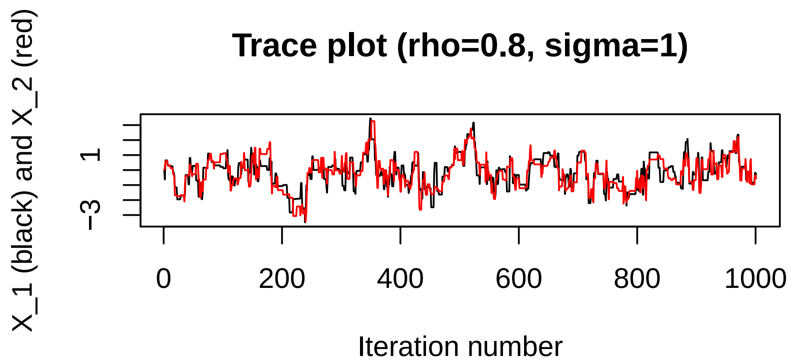
b) As in the case of the Gibbs sampler, the autocorrelation in the samples increases with higher ρ. The percentage of accepted proposals decreases with higher ρ.

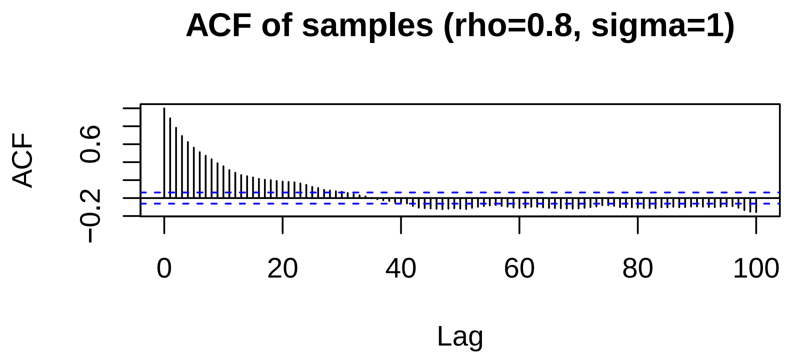
c) If we use $\sigma = 0.2$, the acceptance rate is larger, and if we use $\sigma = 5$, the acceptance rate is lower compared to $\sigma = 1$. Both $\sigma = 0.2$ and $\sigma = 5$ result in higher autocorrelation than $\sigma = 1$. I.e., neither a too high acceptance rate nor a too low acceptance rate is optimal.

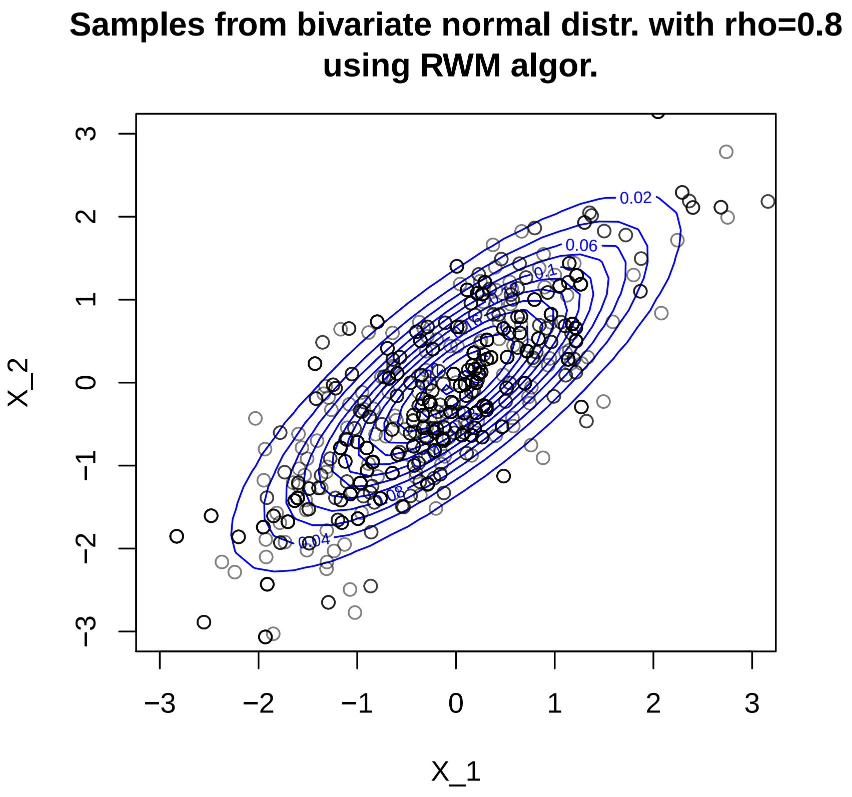
d) Using $(20, 20)$ as initial value and $\rho = 0.99$, the stationary distribution is reached after approximatetly 200-300 iterations. To be on the safe side, we would discard at least the first 300 simulated values.

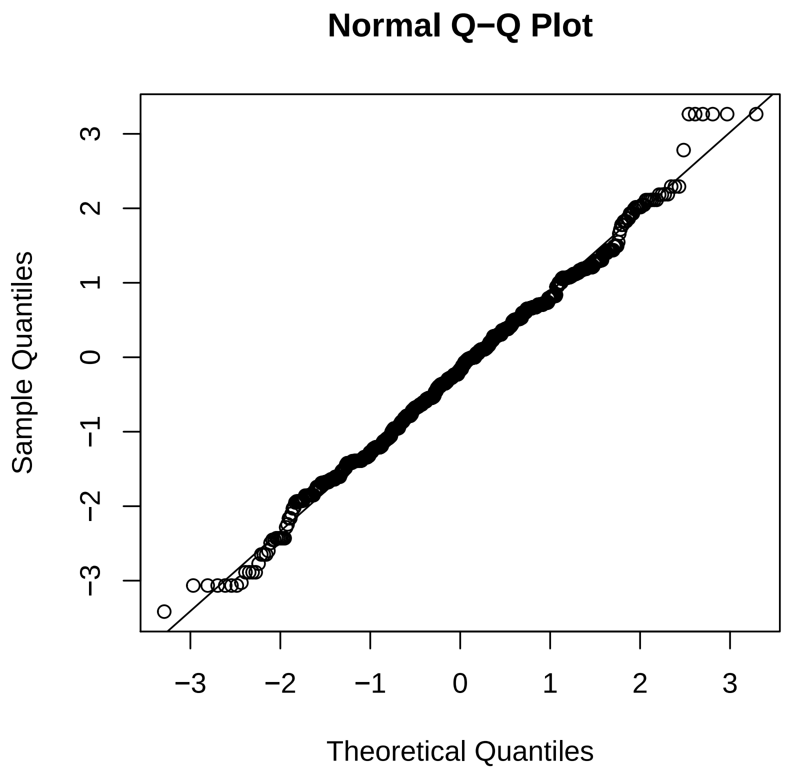
```r
sim.RMV.biv.normal=function(rho,N,sv=c(0,0),
                            plotSamples=TRUE,burnin=NULL,sigma=1,...){
  S=matrix(c(1,rho,rho,1),ncol=2)
  Si=solve(S)
  sim_rwm=matrix(NA,nrow=2,ncol=N+1)
  sim_rwm[,1]=sv##Deterministic initial value
  ##Ratio of the target distributions used in determining the acceptance probability
  ##Note that we do not need the normalizing constants
  a.ratio <- function(xprev,xprop,Si){
    exp(-0.5*(t(xprop)%*%Si%*%xprop-t(xprev)%*%Si%*%xprev))
  }
  set.seed(1)
  acc=0
  for(i in 1:N){
    prop=sim_rwm[,i]+rnorm(2,sd=sigma)
    if(runif(1)<=a.ratio(xprev=sim_rwm[,i],xprop=prop,Si=Si)){
      sim_rwm[,i+1]=prop
      acc=acc+1
    }else{
      sim_rwm[,i+1]=sim_rwm[,i]
    }
  }
  print(paste("Acceptance ratio: ",acc/N,sep=""))
  ##Plot results
  par(mfrow=c(2,1))
  plot(sim_rwm[1,],type="l",
       main=paste("Trace plot (rho=",rho,", sigma=",sigma,")",sep=""),
       xlab="Iteration number",ylab="X_1 (black) and X_2 (red)")
  lines(sim_rwm[2,],col=2)
  ##Remove samples from burn-in phase (if there is a burn-in phase)
  if(!is.null(burnin)) sim_rwm=sim_rwm[,-(1:burnin)]
  acf(sim_rwm[1,],main=paste("ACF of samples (rho=",
                              rho,", sigma=",sigma,")",sep=""),lag.max=100)
  if(plotSamples){
    par(mfrow=c(1,1))
    f.contour.biv.norm(rho,xlab="X_1",ylab="X_2",
                       main=paste("Samples from bivariate normal distr. with rho=",
                                  rho, "\n using RWM algor.",sep=""))
    points(sim_rwm[1,],sim_rwm[2,],...)

    qqnorm(sim_rwm[2,])
    qqline(sim_rwm[2,])
  }
}
N=1000
sim.RMV.biv.normal(rho=0.8,N=N,col=col)

## [1] "Acceptance ratio: 0.377"

sim.RMV.biv.normal(rho=0.95,N=N,col=col,plotSamples=FALSE)

## [1] "Acceptance ratio: 0.224"

sim.RMV.biv.normal(rho=0.99,N=N,col=col,plotSamples=FALSE)

## [1] "Acceptance ratio: 0.098"

sim.RMV.biv.normal(rho=0.95,N=N,col=col,plotSamples=FALSE,sigma=0.2)

## [1] "Acceptance ratio: 0.7"
```

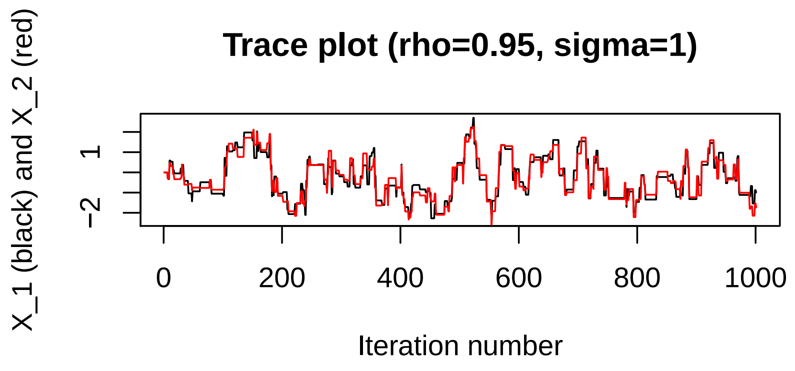
```
sim.RMV.biv.normal(rho=0.95,N=N,col=col,plotSamples=FALSE,sigma=5)

## [1] "Acceptance ratio: 0.02"

sim.RMV.biv.normal(rho=0.95,N=N,col=col,sv=c(20,20),plotSamples=FALSE,burnin=200)

## [1] "Acceptance ratio: 0.228"

sim.RMV.biv.normal(rho=0.99,N=N,col=col,sv=c(20,20),plotSamples=FALSE,burnin=300)

## [1] "Acceptance ratio: 0.125"
```

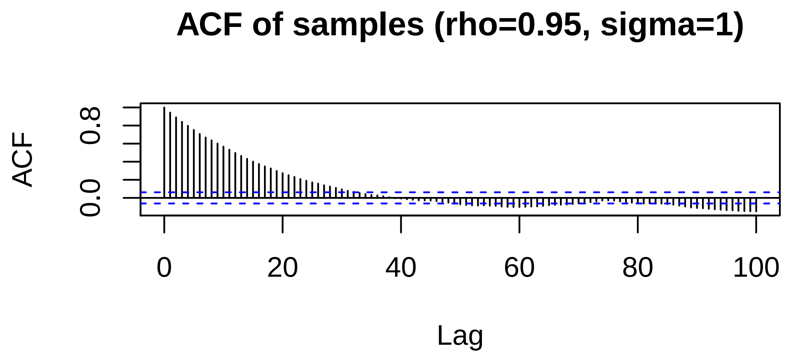


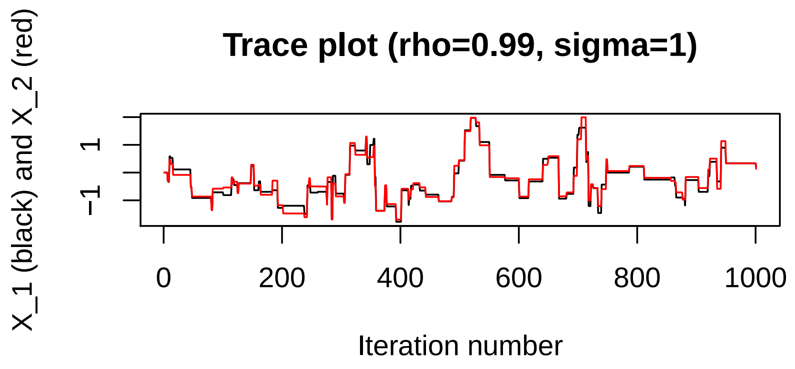


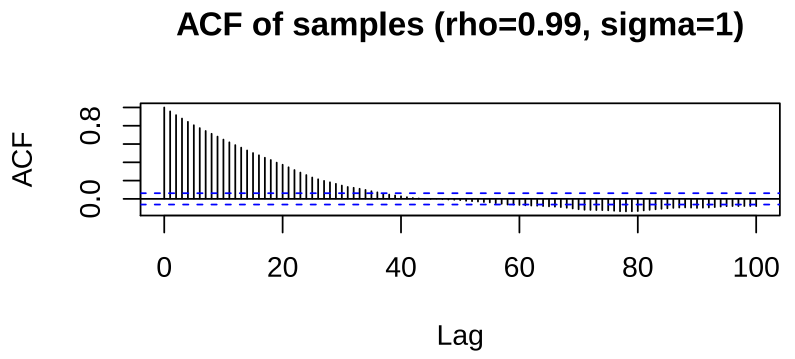

Samples from bivariate normal distr. with rho=0.8
using RWM algor.
X_1
X_2
0.02
0.06
0.1
0.04
−3
−2
−1
0
1
2
3

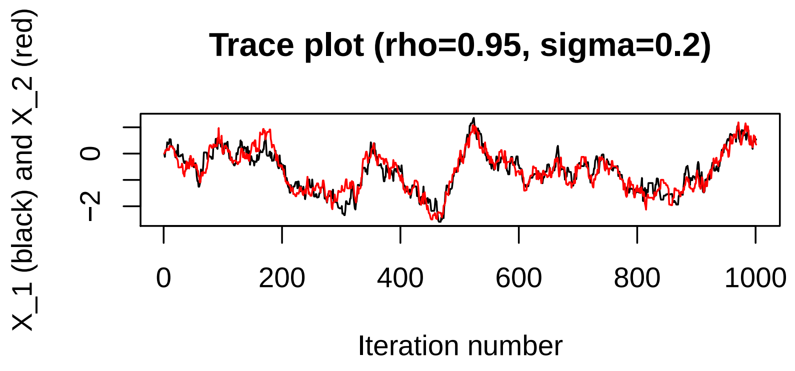

Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−3
−2
−1
0
1
2
3

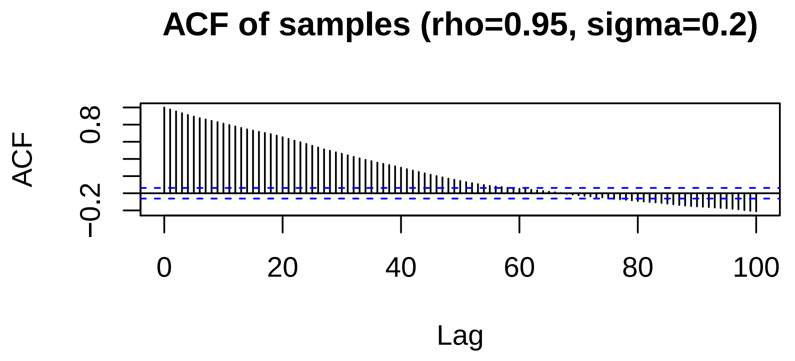

Trace plot (rho=0.95, sigma=1)
X_1 (black) and X_2 (red)
1
−2
0
200
400
600
800
1000
Iteration number

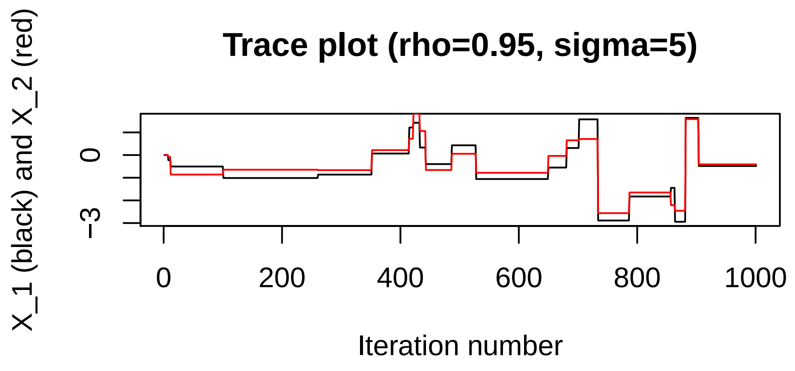

ACF of samples (rho=0.95, sigma=1)
ACF
0.0
0.8
0
20
40
60
80
100
Lag

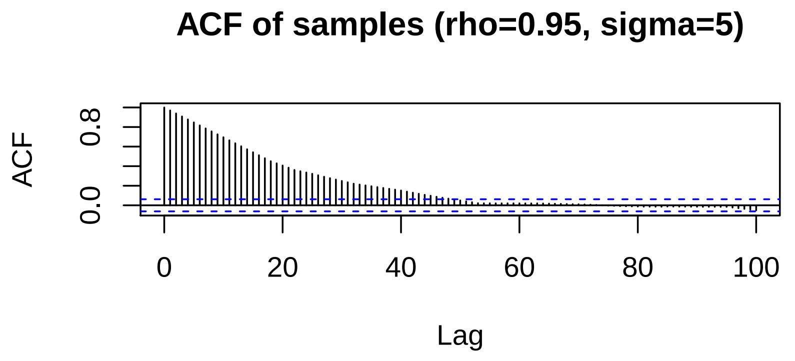

Trace plot (rho=0.99, sigma=1)
X_1 (black) and X_2 (red)
−1
1
0
200
400
600
800
1000
Iteration number

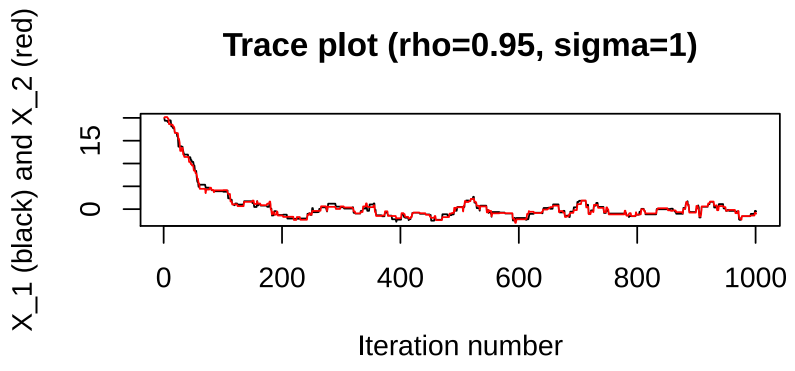

ACF of samples (rho=0.99, sigma=1)
ACF
0.0
0.8
0
20
40
60
80
100
Lag

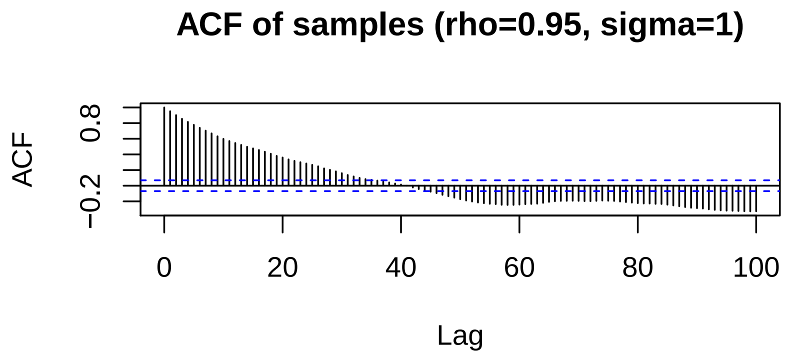

Trace plot (rho=0.95, sigma=0.2)
X_1 (black) and X_2 (red)
−2
0
0
200
400
600
800
1000
Iteration number

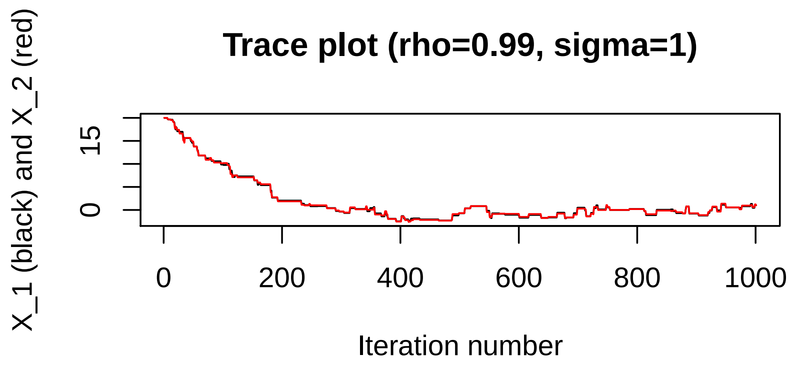

ACF of samples (rho=0.95, sigma=0.2)
ACF
−0.2
0.8
0
20
40
60
80
100
Lag

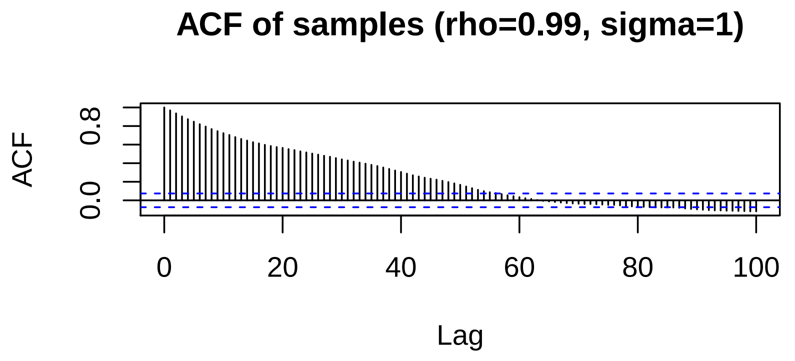

Trace plot (rho=0.95, sigma=5)
X_1 (black) and X_2 (red)
0
−3
0
200
400
600
800
1000
Iteration number


ACF of samples (rho=0.95, sigma=5)
ACF
0.8
0.0
0
20
40
60
80
100
Lag


Trace plot (rho=0.95, sigma=1)
X_1 (black) and X_2 (red)
15
0
0
200
400
600
800
1000
Iteration number


ACF of samples (rho=0.95, sigma=1)
ACF
0.8
−0.2
0
20
40
60
80
100
Lag


Trace plot (rho=0.99, sigma=1)
X_1 (black) and X_2 (red)
0
15
0
200
400
600
800
1000
Iteration number


ACF of samples (rho=0.99, sigma=1)
ACF
0.0
0.8
0
20
40
60
80
100
Lag