Biostat 213 Final Project

Hubert Jenq

12/9/2016

1. The two dimensional Banana distribution can be directly simulated using a metropolis-hastings sampler. The proposal distribution used is a bivariate normal distribution with a diagnal covariance matrix (correlation=0). The proposal distribution is centered at (0,-10) since that is where the max occurs for the banana distribution. Two M-H samplers are attempted, one where the proposal distribution does not depend on the current state (independent M-H) and one where the proposal distribution is re-centered at the current state (random-walk M-H). The jump sizes of the proposal distribution was changed by scaling the covariance by a factor of .5, 1, or 2.

The metropolis hastings sampler works by choosing a candidate point from a proposal distribution and accepts it based on the candidate point compared to the current point's target distribution and proposal distribution.

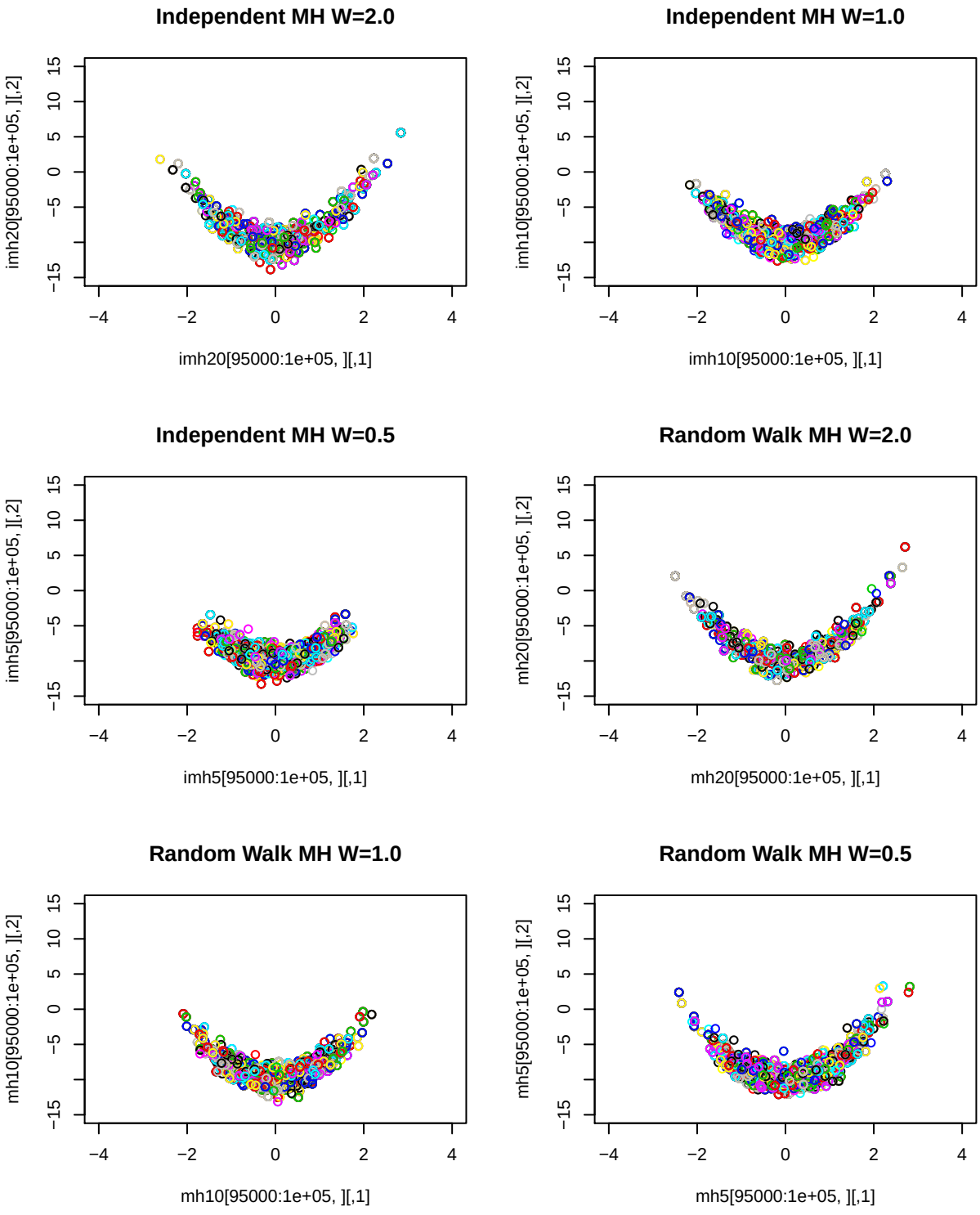
When the jump variance is increased, more samples seem to be rejected and the chain does not look as well mixing from the time series plots. However, the chain seems to sample from the lower-probability regions towards the edges of the banana shape better than when the variance is low.

```
#Random walk M-H
#n = number of iterations for markov chain to run
#a = scaling factor for covariance matrix the proposal bivariate normal distribution
gamm<-function (n, a)
{

  #store the iterations
  vec <- matrix(0, n,2)

  #bivariate normal parameters
  mu=c(0,-10)
  cov=a*diag(2)
  #look at acceptance probabilities
  aproblist=rep(0,n)

  #intial start condition
  x = mu

  vec[1,] <- c(x[1],x[2])
  for (i in 2:n) {
    can <- mvrnorm(1, x, cov)
    canx1=can[1]
    canx2=can[2]
    #acceptance probability, use exp(log()) of the probabilities to help with rounding errors.
    #draw from the proposal distribution centered at the current position x of the chain.
    aprob <- min(1, exp((log(exp(-canx1^2/2)*exp(-(canx2-2*(canx1^2-5))^2/2))-log(exp(-x[1]^2/2)*exp(-(
                        +log(dmvnorm(x, can, cov))-log(dmvnorm(can, x, cov)))))

    aproblist[i]=aprob
    u <- runif(1)

    #accept the point with probability aprob else stay at the same point
    if (u < aprob)
```

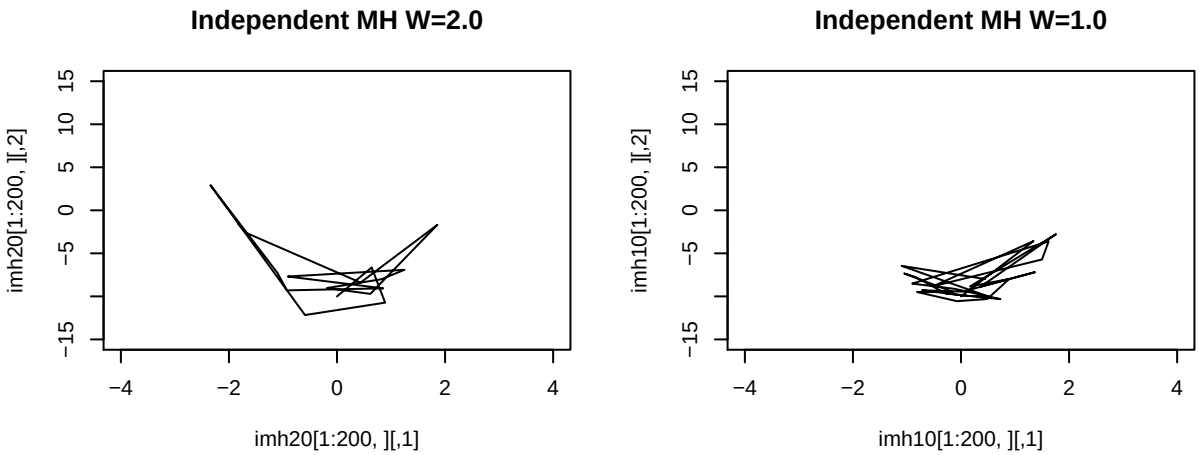
```
      x <- c(canx1,canx2)
    vec[i,] <- x
  }
  return(vec)
}

#Independent M-H
imm<-function (n, a)
{

  #store the iterations
  vec <- matrix(0, n,2)

  #bivariate normal parameters
  mu=c(0,-10)
  cov=a*diag(2)
  #look at acceptance probabilities
  aproblist=rep(0,n)

  #intial start condition
  x = mu

  vec[1,] <- c(x[1],x[2])
  for (i in 2:n) {
    can <- mvrnorm(1, mu, cov)
    canx1=can[1]
    canx2=can[2]
    #acceptance probability, use exp(log()) of the probabilities to help with rounding errors.
    #draw from the proposal distribution centered at the current position x of the chain.
    aprob <- min(1, exp((log(exp(-canx1^2/2)*exp(-(canx2-2*(canx1^2-5))^2/2))-log(exp(-x[1]^2/2)*exp(-(
                         +log(dmvnorm(x, mu, cov))-log(dmvnorm(can, mu, cov)))))

    aproblist[i]=aprob
    u <- runif(1)

    #accept the point with probability aprob else stay at the same point
    if (u < aprob)
      x <- c(canx1,canx2)
    vec[i,] <- x
  }
  return(vec)
}

#Gibbs sampler
gibbs<-function (n)
{
  mat <- matrix(ncol = 2, nrow = n)

  #intial start location
  x <- 0
  y <- -10
  mat[1, ] <- c(x, y)
```

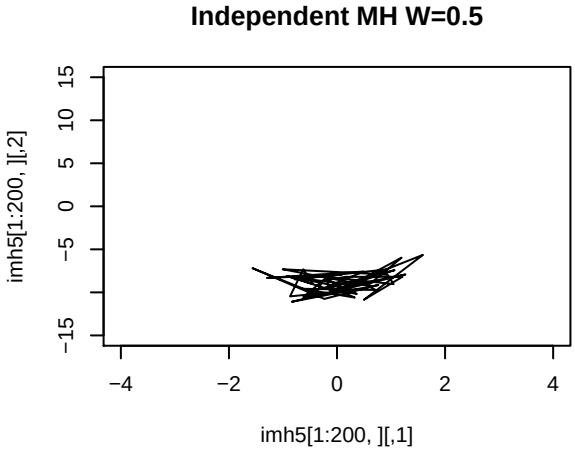
```
  #repeat n times
  for (i in 2:n) {
    #sample from marginal of x1
    x <- rnorm(1)
    #sample from conditional of x2|x1
    y <- rnorm(1, 2*(x^2-5),1)
    #save the point
    mat[i, ] <- c(x, y)
  }
  mat
}
```

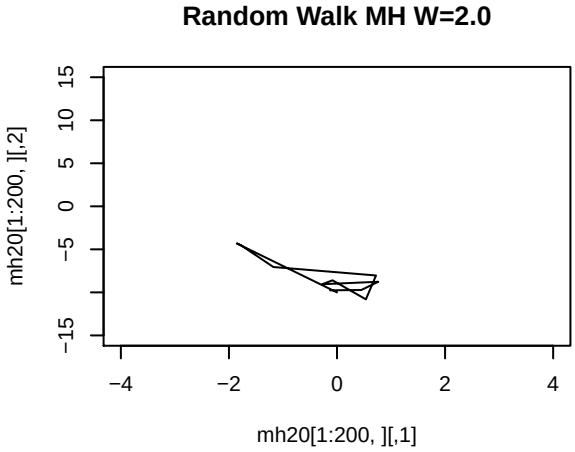
```
#w=2.0
mh20<-gamm(100000,20)
imh20<-imm(100000,20)

#w=1.0
mh10<-gamm(100000,10)
imh10<-imm(100000,10)

#w=0.5
mh5<-gamm(100000,5)
imh5<-imm(100000,5)

gib=gibbs(100000)

par(mfrow=c(3,2))
plot(imh20[95000:100000,],col=1:5000,xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=2.0")
plot(imh10[95000:100000,],col=1:5000,xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=1.0")
plot(imh5[95000:100000,],col=1:5000,xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=0.5")
plot(mh20[95000:100000,],col=1:5000,ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=2.0")
plot(mh10[95000:100000,],col=1:5000,ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=1.0")
plot(mh5[95000:100000,],col=1:5000,ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=0.5")
```

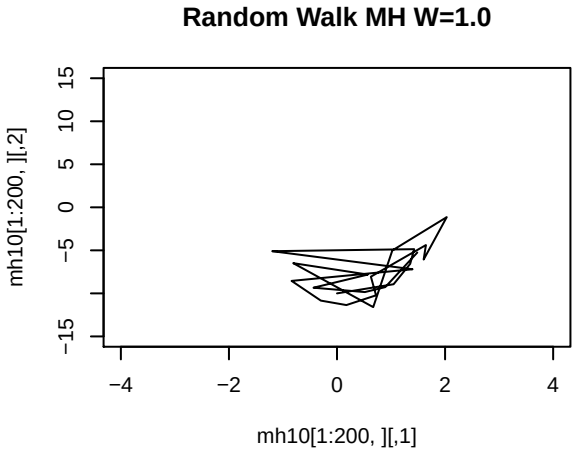


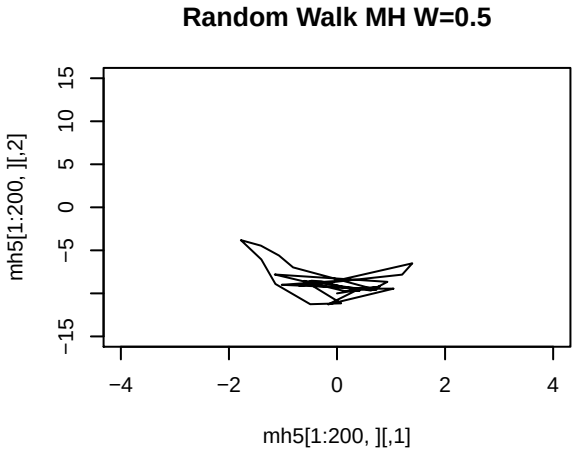

```
par(mfrow=c(3,2))
plot(imh20[1:200,],col=1:200,type="l",xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=2.0")
plot(imh10[1:200,],col=1:200,type="l",xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=1.0")
plot(imh5[1:200,],col=1:200,type="l",xlim=c(-4,4),ylim=c(-15,15),main="Independent MH W=0.5")
plot(mh20[1:200,],col=1:200,type="l",ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=2.0")
plot(mh10[1:200,],col=1:200,type="l",ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=1.0")
plot(mh5[1:200,],col=1:200,type="l",ylim=c(-15,15),xlim=c(-4,4),main="Random Walk MH W=0.5")
```



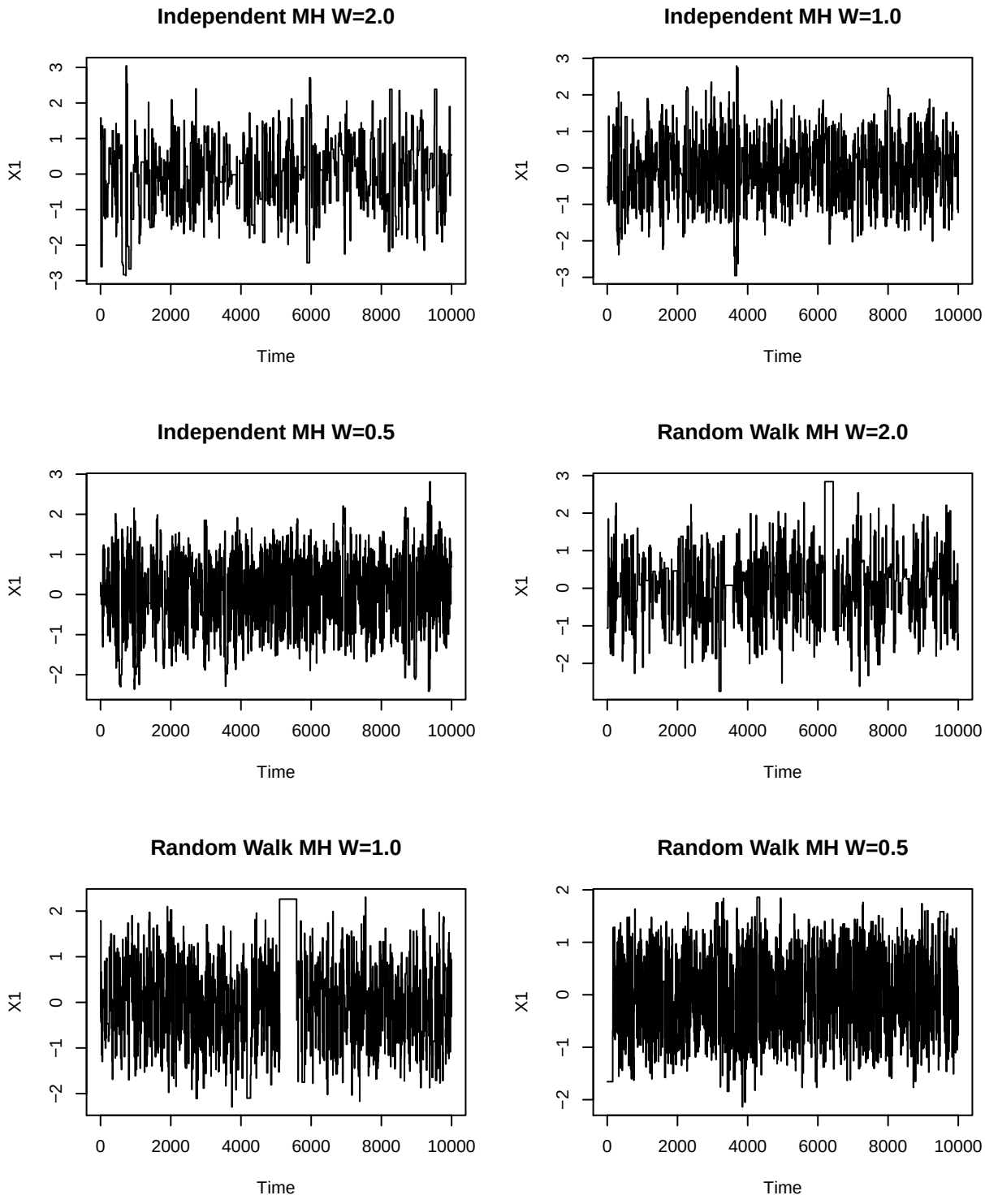




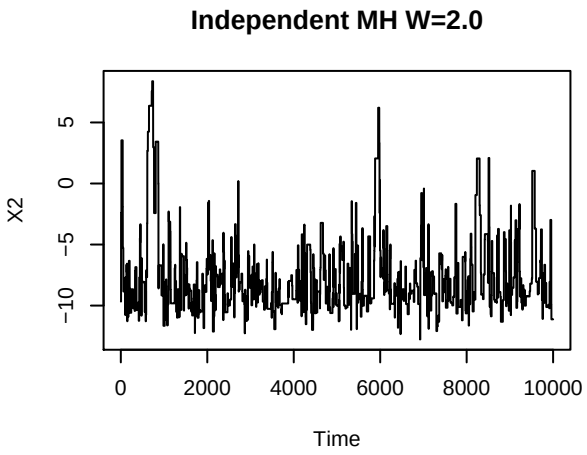


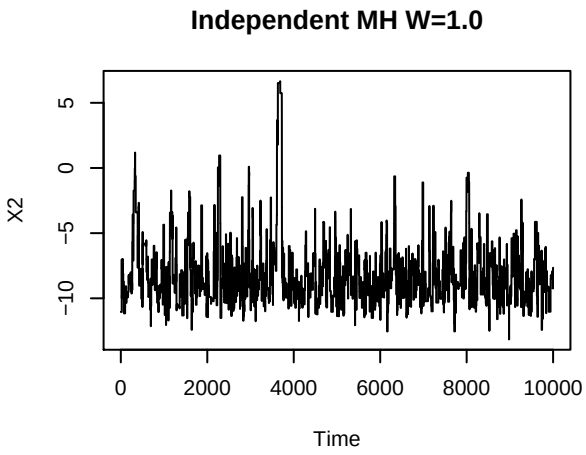


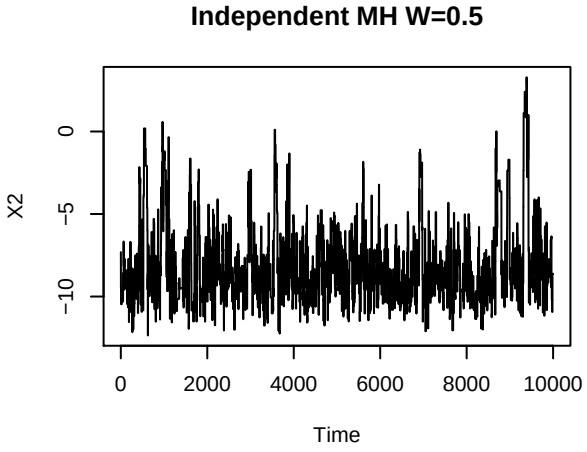

```
par(mfrow=c(3,2))
plot(ts(mh20[90000:100000,1]),main="Independent MH W=2.0",ylab="X1" )
plot(ts(mh10[90000:100000,1]),main="Independent MH W=1.0" ,ylab="X1" )
plot(ts(mh5[90000:100000,1]),main="Independent MH W=0.5" ,ylab="X1")
plot(ts(imh20[90000:100000,1]),main="Random Walk MH W=2.0" ,ylab="X1")
plot(ts(imh10[90000:100000,1]),main="Random Walk MH W=1.0",ylab="X1")
plot(ts(imh5[90000:100000,1]),main="Random Walk MH W=0.5",ylab="X1")
```

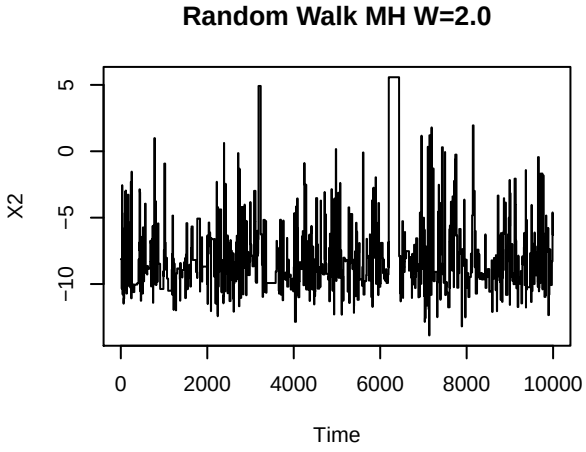


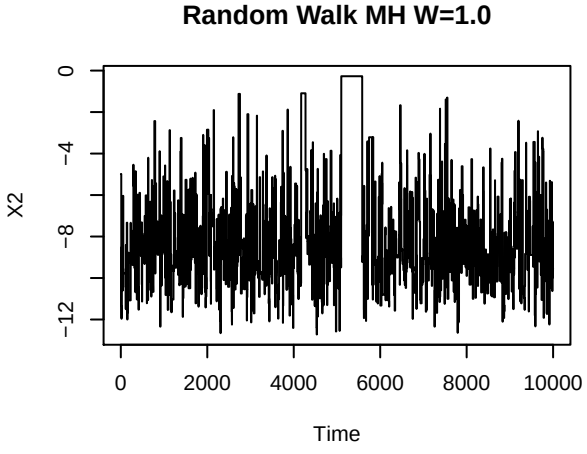

```
par(mfrow=c(3,2))
plot(ts(mh20[90000:100000,2]),main="Independent MH W=2.0",ylab="X2")
plot(ts(mh10[90000:100000,2]),main="Independent MH W=1.0",ylab="X2")
plot(ts(mh5[90000:100000,2]),main="Independent MH W=0.5",ylab="X2")
plot(ts(imh20[90000:100000,2]),main="Random Walk MH W=2.0",ylab="X2")
plot(ts(imh10[90000:100000,2]),main="Random Walk MH W=1.0",ylab="X2")
plot(ts(imh5[90000:100000,2]),main="Random Walk MH W=0.5",ylab="X2")
```

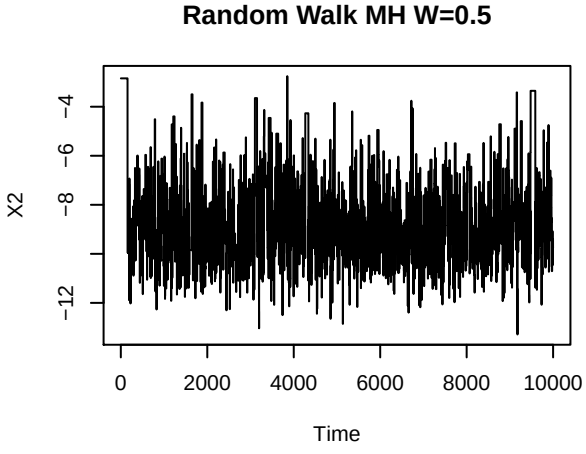

Independent MH W=2.0
X2
5
0
−5
−10
0
2000
4000
6000
8000
10000
Time


Independent MH W=1.0
X2
5
0
−5
−10
0
2000
4000
6000
8000
10000
Time


Independent MH W=0.5
X2
0
−5
−10
0
2000
4000
6000
8000
10000
Time

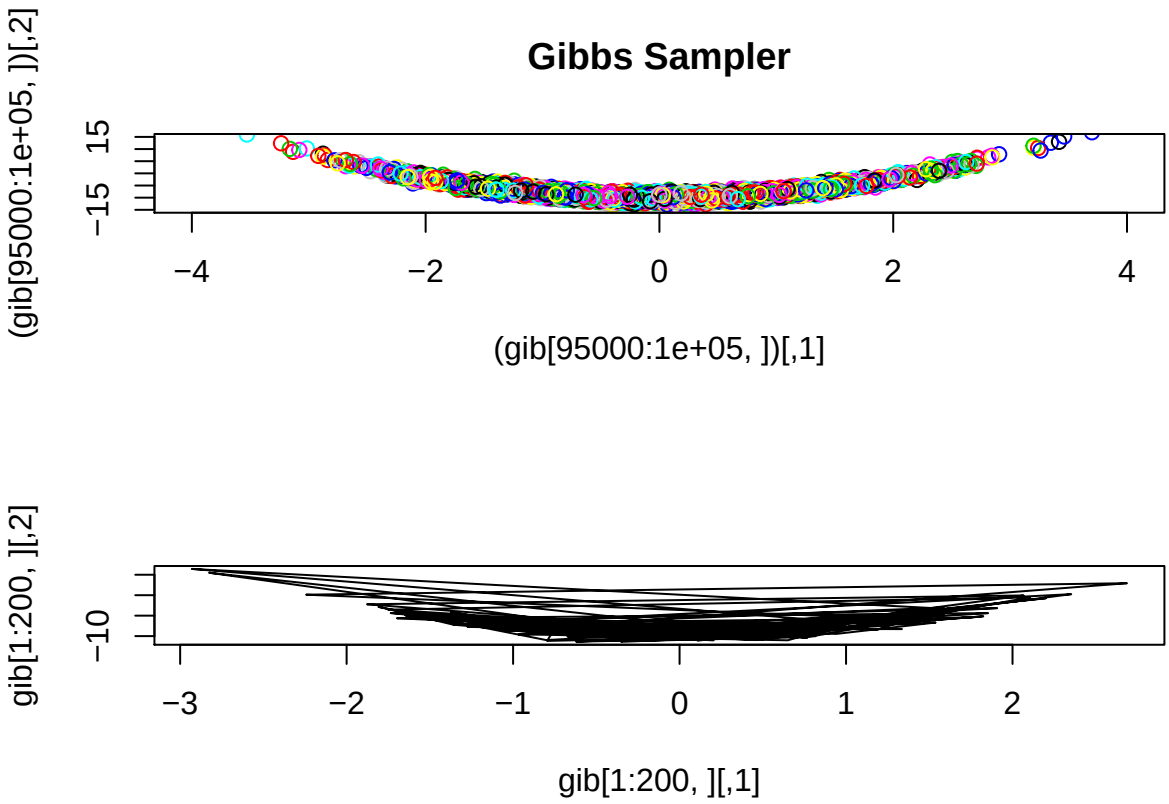

Random Walk MH W=2.0
X2
5
0
−5
−10
0
2000
4000
6000
8000
10000
Time


Random Walk MH W=1.0
X2
0
−4
−8
−12
0
2000
4000
6000
8000
10000
Time

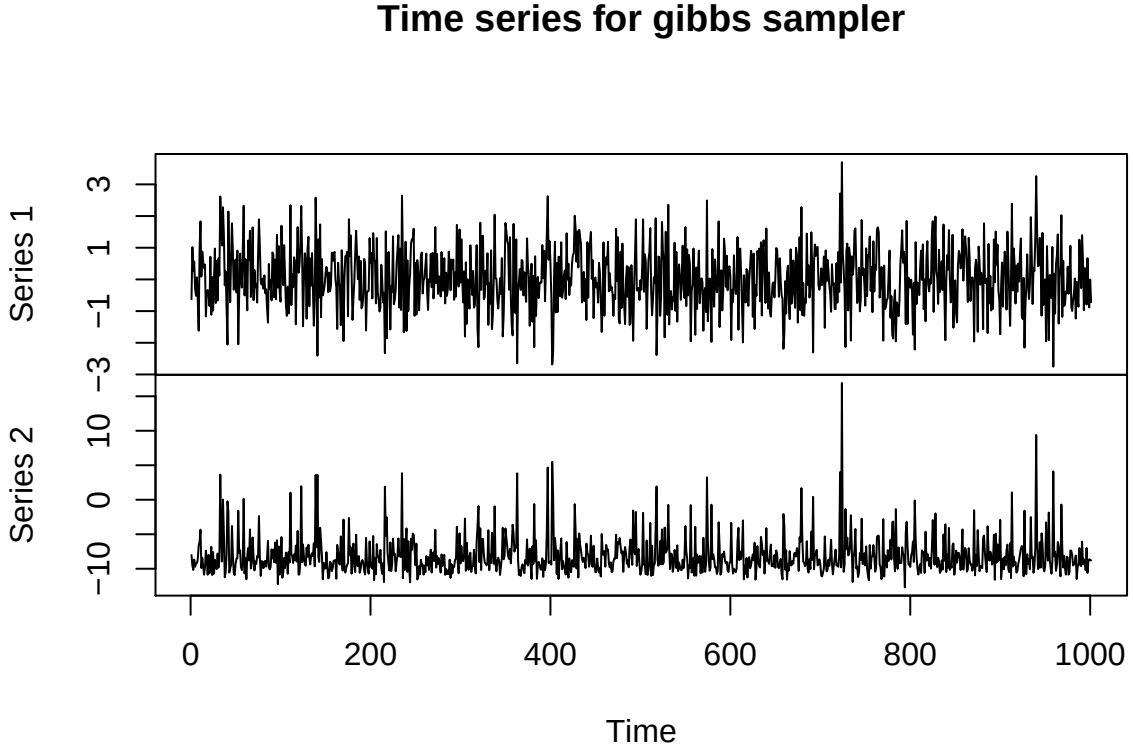

Random Walk MH W=0.5
X2
−4
−6
−8
−12
0
2000
4000
6000
8000
10000
Time

2. A Gibbs sampler is made using the marginal density of x1 and the conditional density of x2|x1. x1 is first drawn from, and then x2 is drawn from the conditional density. This is iteratively done for n samples. Compared to the MH we see that many more points are simulated since the acceptance probability of a gibbs sampler is 1. The gibbs sampler looks to be well mixing and converges quickly. Since a conditional form of the distribution was obtainable, this method seems to work much better than the M-H.

```
par(mfrow=c(2,1))
plot((gib[95000:100000,]),col=1:5000,ylim=c(-15,15),xlim=c(-4,4),main="Gibbs Sampler")
plot(gib[1:200,],type="l",col=1:200)
```




```
plot(ts(gib[99000:100000,]),main="Time series for gibbs sampler")
```


Time series for gibbs sampler
Series 1
Series 2
3
1
−1
−3
10
0
−10
0
200
400
600
800
1000
Time