hybrid_MCMC(from scratch)

Hyung Jun Lim

Hybrid MCMC

Diffrentiation function

```
deriv <- function(x, func){
  h <- 10^(-7)
  dx <- h
  df <- func(x+h) - func(x)
  deriv <- df/dx
  return(deriv)
}

#test
deriv(0.5, function(x) log(x)) # 1/0.5=2
```

```
## [1] 2
```

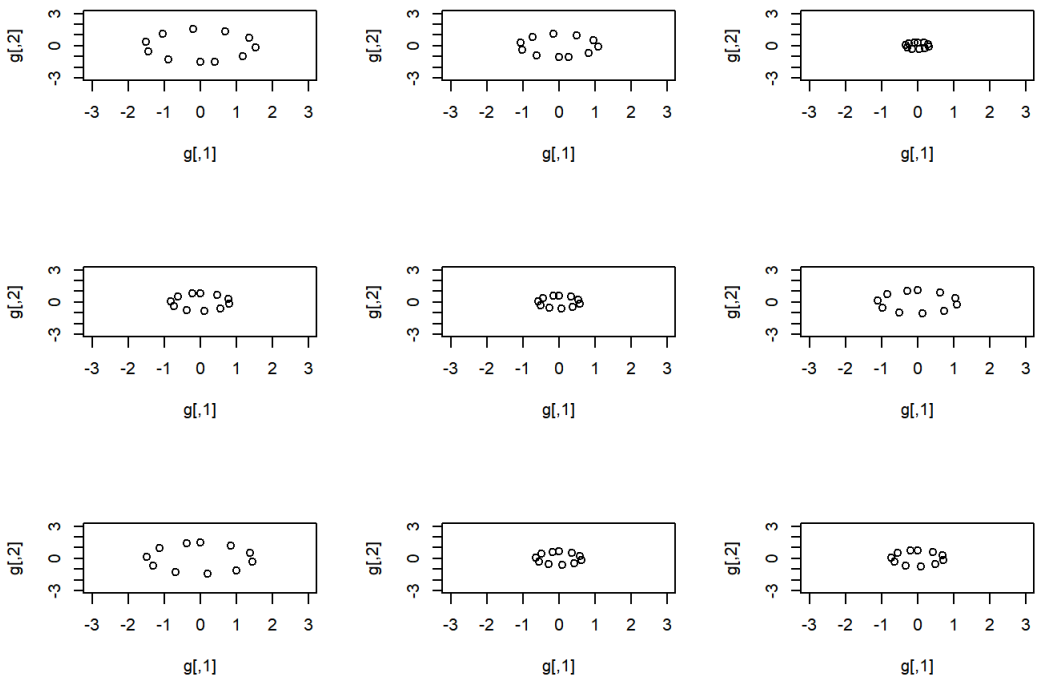
```
deriv(2, function(x) 1/x^2)  # -2/8 = -1/4
```

```
## [1] -0.25
```

Leapfrog integrator

```
## leapfrog function
LF <- function(x, p, dKdp, dVdx, step_size, step_iter){
  for(i in 1:step_iter){
    p <- p + step_size/2 * dVdx(x) #half step for momentum
    x <- x - step_size * dKdp(p) #whole step for x
    p <- p + step_size/2 * dVdx(x) # another half step for momentum
  }
  obj <- c(x,-p) # momentum flip
  return(obj)
}
```

```
# setting
V <- function(x) -log(dnorm(x))
dVdx <- function(x) deriv(x, V)
dKdp <- function(p) p

# leapfrog process for 9 different momentum values
par(mfrow=c(3,3))
for(k in 1:9){
  g <- matrix(c(0,rnorm(1)),1,2)
  for(i in 1:10){
    g <- rbind(g, LF(g[i,1], -g[i,2], dKdp, dVdx, 0.3, 2) )
  }
  plot(g, xlim=c(-3,3), ylim=c(-3,3))
}
```

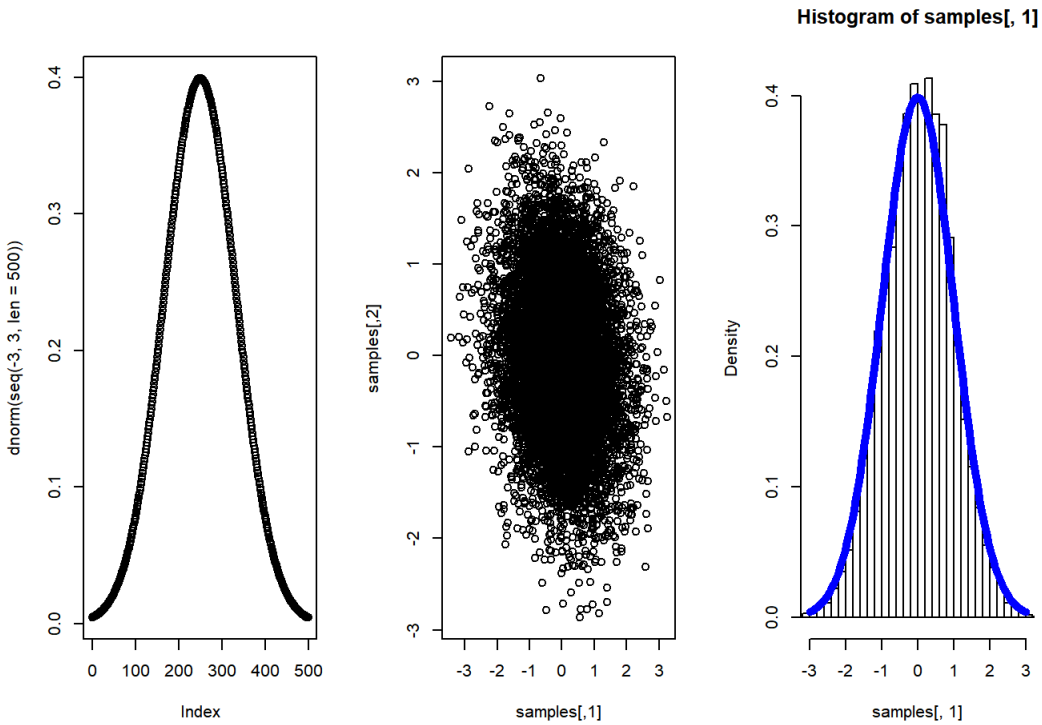


Hamiltonian MCMC function

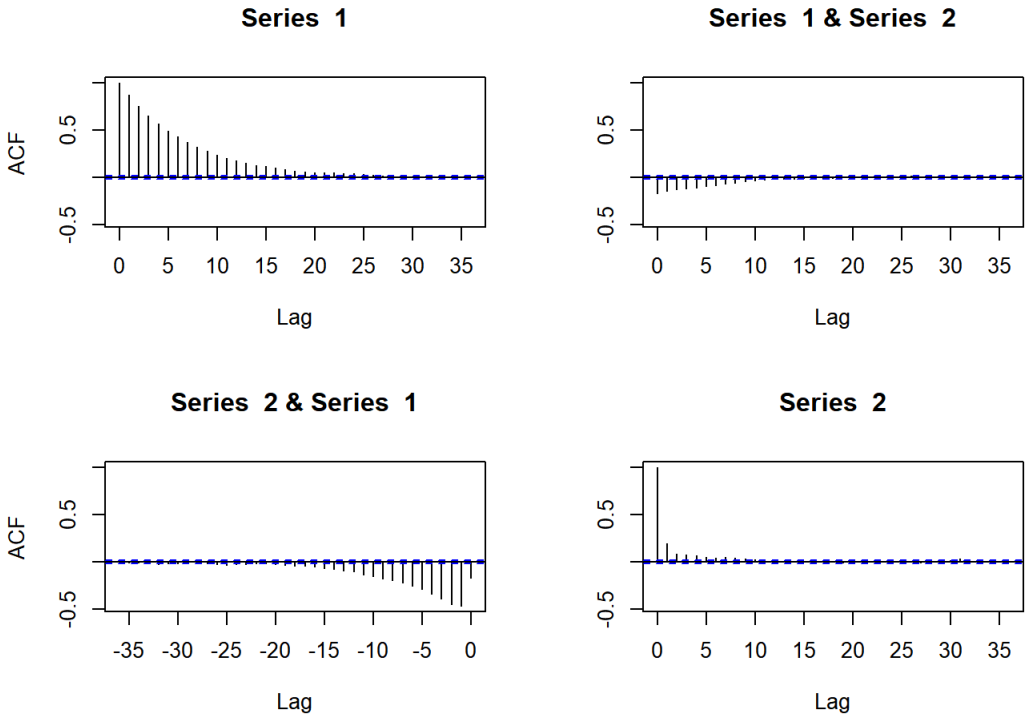
```
HamMC <- function(n_iter, initial_x, likelihood, step_size, step_iter){

  # set of sampled x, momentum p
  samples <- c()

  #initial momentum
  initial_p <- rnorm(1)

  # Hamiltonian function H = V+K
  V <- function(x) -log(likelihood(x))
  K <- function(p) -log(dnorm(p))

  #Derivatives functions
  dVdx <- function(x) deriv(x, V)
  dKdp <- function(p) p

  #leapfrog integration
  x <- initial_x
  p <- initial_p

  for(i in 1:n_iter){
    result <- LF(x,rnorm(1), dKdp, dVdx, step_size, step_iter)
    x_new <- result[1]
    p_new <- result[2]

    old_l <- likelihood(x)*dnorm(p)
    new_l <- likelihood(x_new)*dnorm(p_new)
    alpha <- min(1, new_l/old_l)
    if(alpha > runif(1)){
      new <- c(x_new, p_new)
      samples <- rbind(samples, new)
    }else{
      new <- c(x,p)
      samples <- rbind(samples, new)
    }
    x <- x_new
    p <- p_new
  }
  return(samples)
}
```

Test: Sampling N(0,1)

```
par(mfrow=c(1,3))
# test HamMC function (N(0,1))
plot(dnorm(seq(-3,3,len=500)))
samples <- HamMC(10000, 0, dnorm, 0.05, 10)
plot(samples)
hist(samples[,1],nclass = 30, probability=T, xlim=c(-3,3))
lines(x=seq(-3,3,len=100), y=dnorm(seq(-3,3,len=100)), col='blue', lwd=5)
```

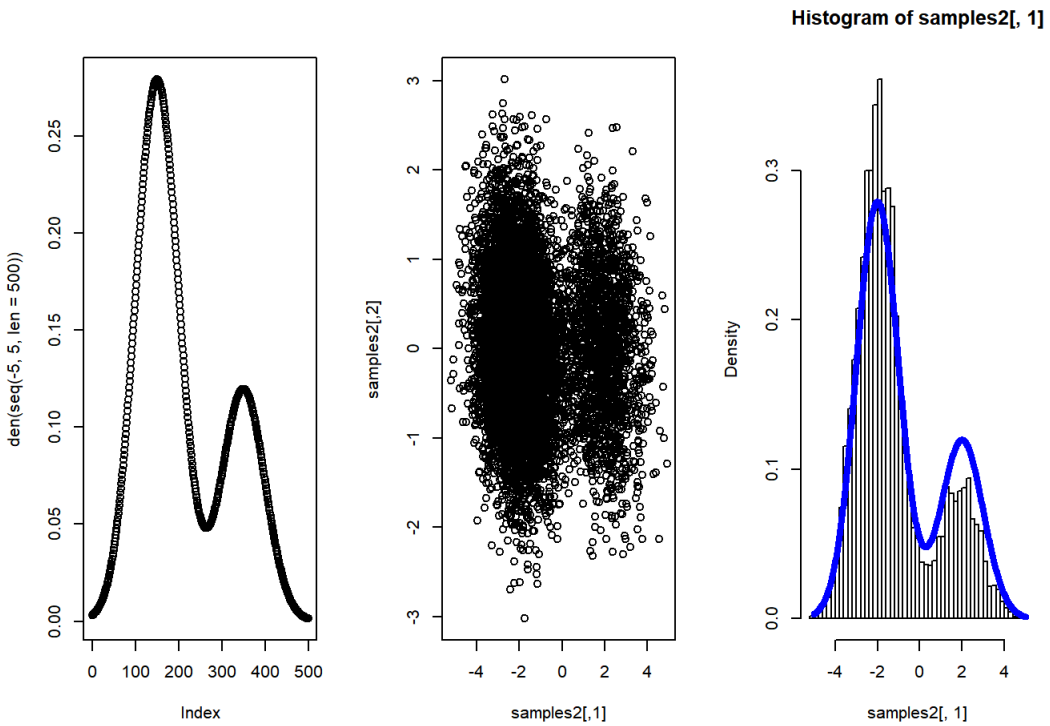



```
acf(samples)
```

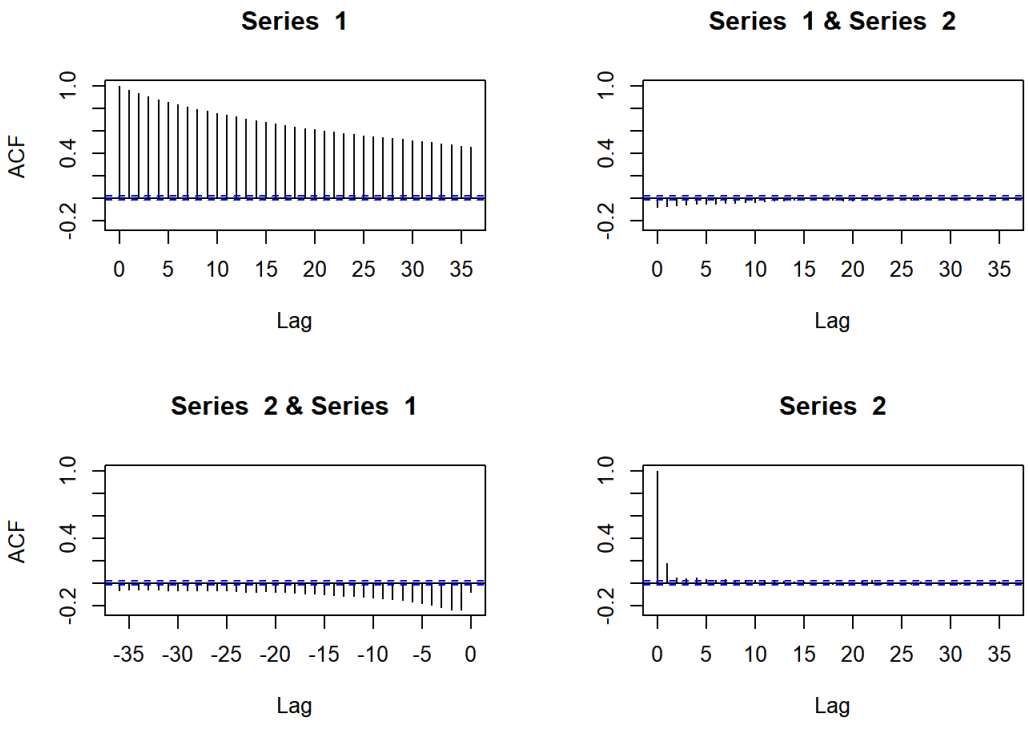


Test: Normal Mixture (Bimodal)

```
par(mfrow=c(1,3))

den <- function(x) 0.3*dnorm(x,mean=2,sd=1) + 0.7*dnorm(x, mean=-2, sd=1)
plot(den(seq(-5,5,len=500)))

samples2 <- HamMC(10000, 0, den, 0.05, 10)
plot(samples2)
hist(samples2[,1],probability = T, nclass = 40)
lines(x=seq(-5,5,len=500), y=den(seq(-5,5,len=500)), col='blue', lwd=4)
```

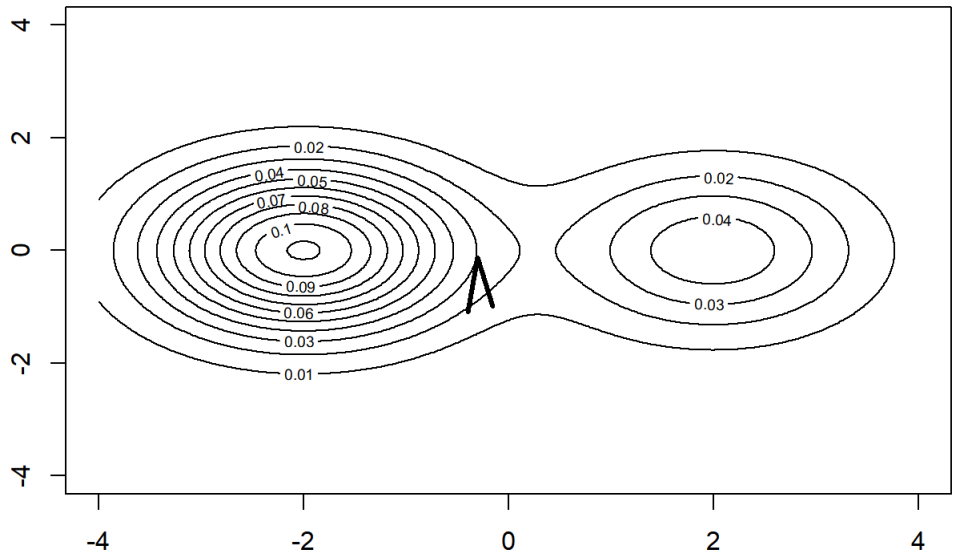



```
acf(samples2)
```

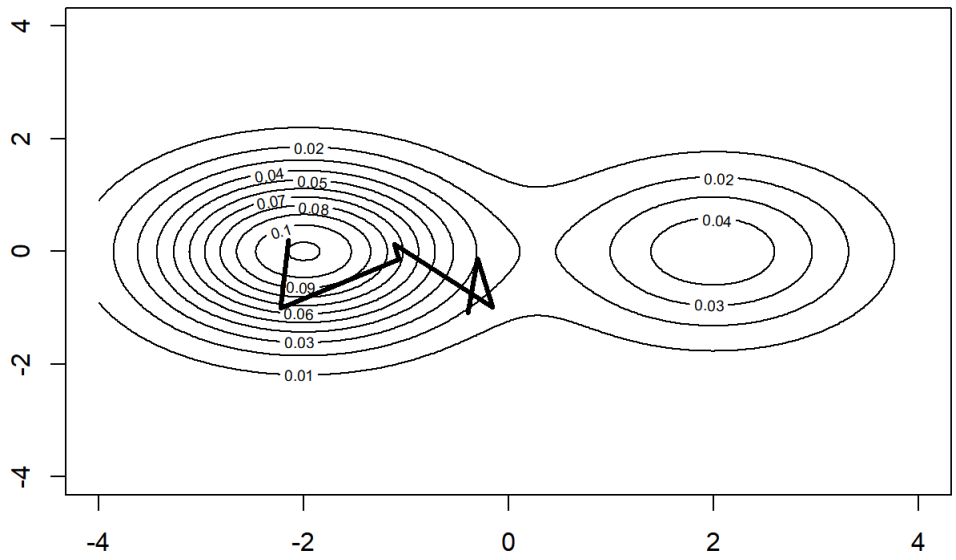


Contour plot: Sampling Process visualization

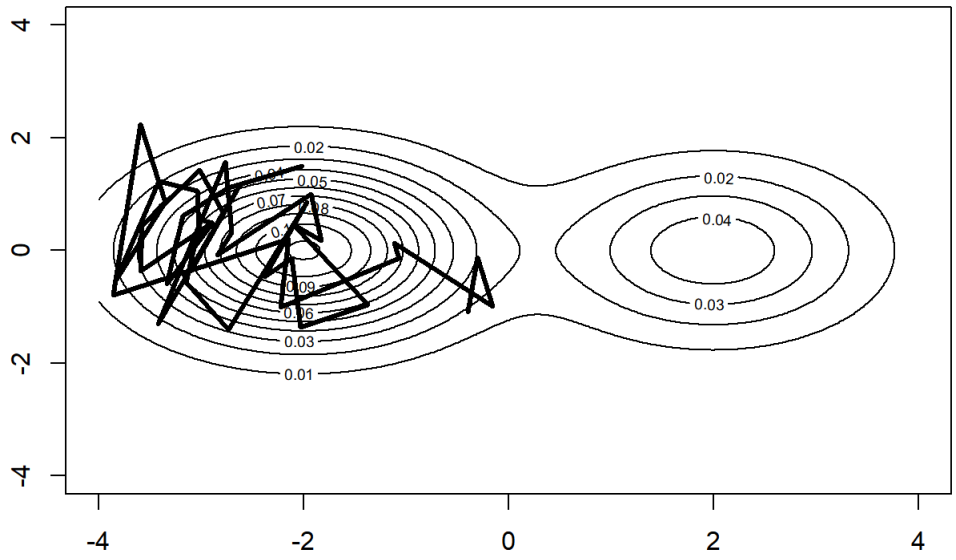
```
# test data
x <- seq(-4,4,len=500)
y <- x
z <- matrix(0,500,500)
for(i in 1:500){
  for(j in 1:500){
    z[i,j] <- (0.3*dnorm(x[i],mean=2,sd=1) + 0.7*dnorm(x[i], mean=-2, sd=1))*dnorm(y[j])
  }
}

contour(x,y,z)
points(samples2[1:5,],type='l', lwd=3)
```



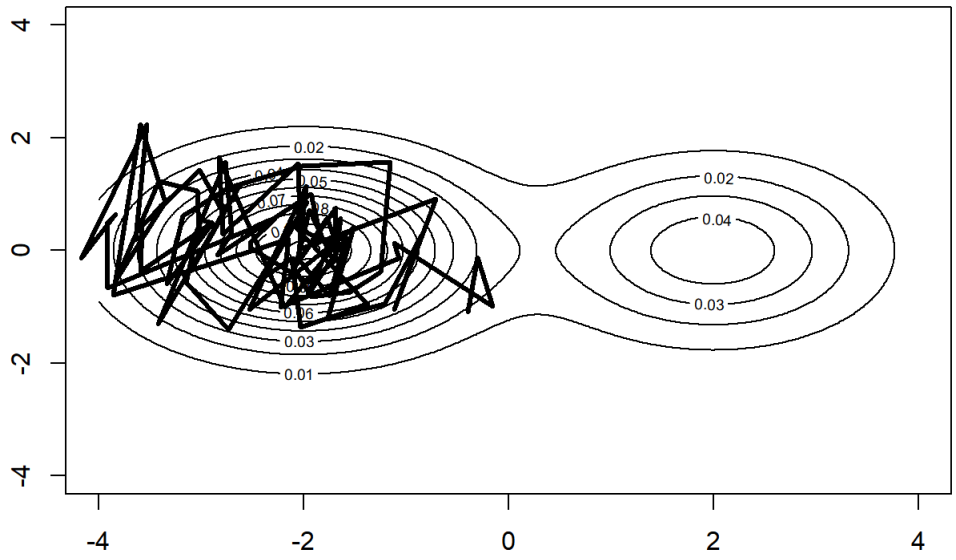

```
contour(x,y,z)
points(samples2[1:10,],type='l', lwd=3)
```



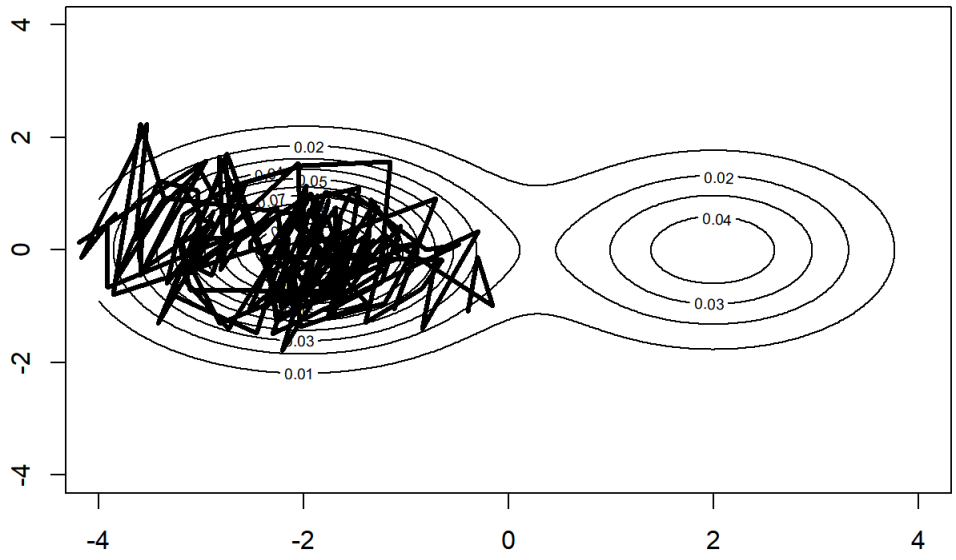
```
contour(x,y,z)
points(samples2[1:50,],type='l', lwd=3)
```



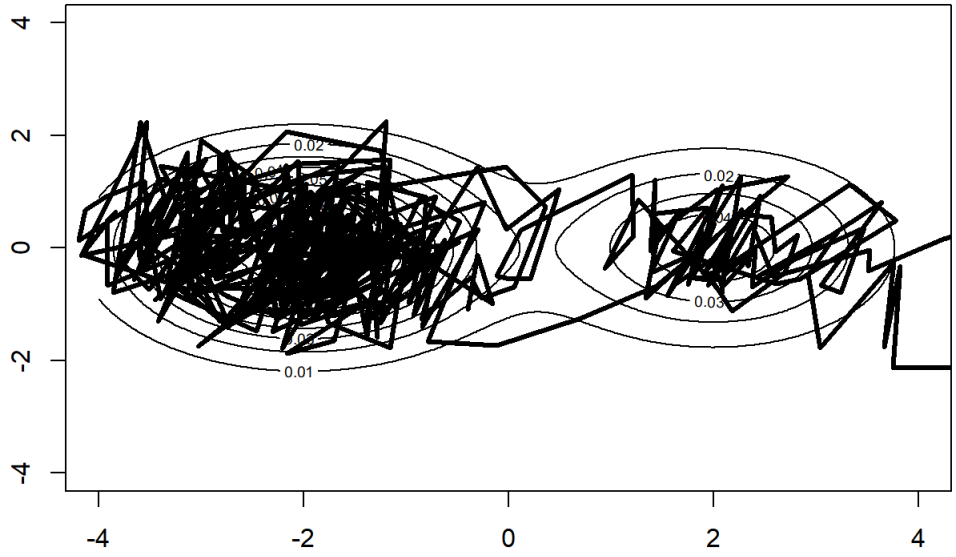

```
contour(x,y,z)
points(samples2[1:100,],type='l', lwd=3)
```



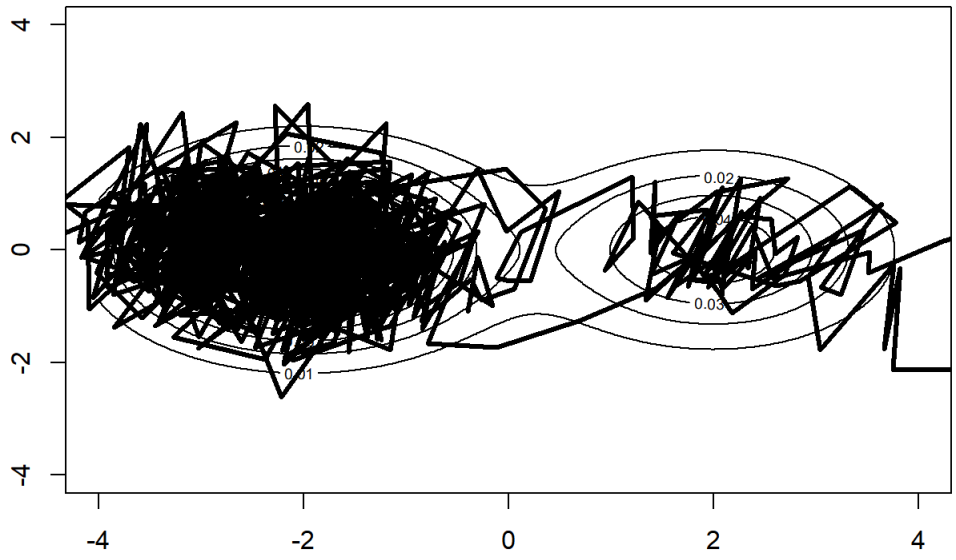

```
contour(x,y,z)
points(samples2[1:200,],type='l', lwd=3)
```



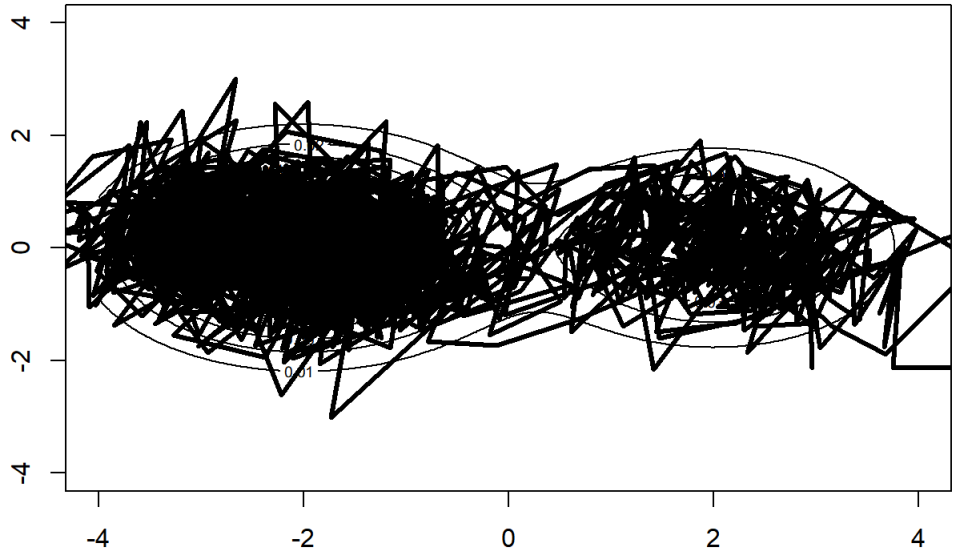

```
contour(x,y,z)
points(samples2[1:500,],type='l', lwd=3)
```



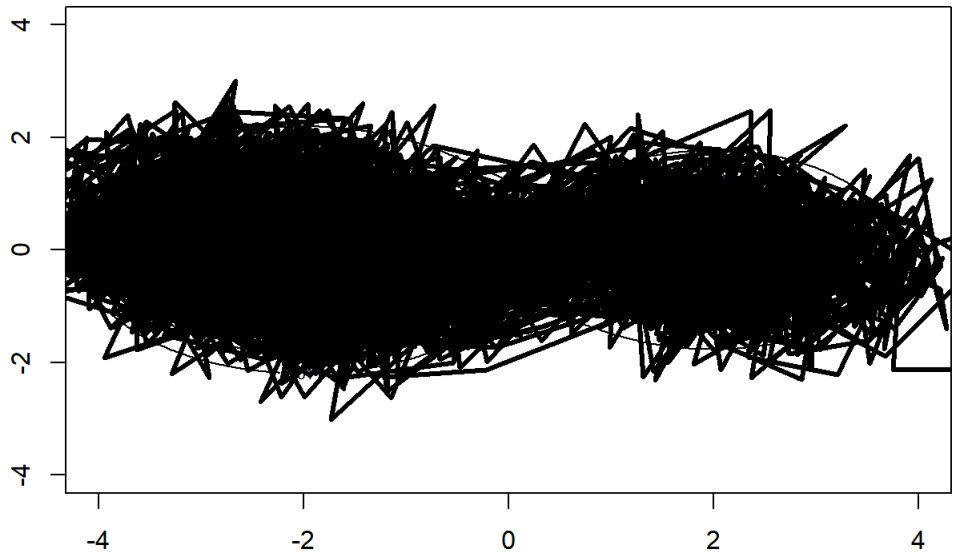

```
contour(x,y,z)
points(samples2[1:1000,],type='l', lwd=3)
```




```
contour(x,y,z)
points(samples2[1:2000,],type='l', lwd=3)
```




```
contour(x,y,z)
points(samples2,type='l', lwd=3)
```


4
2
0
-2
-4
-4
-2
0
2
4