SFB-1491 Graduate School: Astrostatistics

Metropolis Hastings Example

Dr Angus H Wright

2023-02-02

The Metropolis-Hastings Algorithm

The Gibbs sampler is a special case of the "Metropolis-Hastings" algorithm

Set $t = 1$ and while $t \leq N$:

1. Generate y from $Q(x_t, .)$ and U from $U(0, 1)$;
2. let $x_{t+1} = y$ if $U \leq \alpha(x_t, y)$, otherwise let $x_{t+1} = x_t$.
3. $t = t + 1$; iterate.

where

- $Q(.,.)$ is the **candidate generation function**, which can be interpreted as saying that, when a process is at the point x, the density generates a value y from $Q(x, y)$.
- $\alpha(.,.)$ is the **acceptance probability**, defined by:

$$\alpha(x, y) = \begin{cases} \min\left(1, \frac{\pi(y)Q(y,x)}{\pi(x)Q(x,y)}\right) & \text{if } \pi(x)Q(x, y) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

  That is, if the move is accepted, the provrdd moves to y, otherwise it remains at x.

The Gibbs Sampler is therefore a special case of the MH algorithm, where the candidate generating function $Q(x, y)$ is the conditional distribution:

$$Q(x, y) = \pi(x|y)$$

and the acceptance probability is unity for all x, y:

$$\alpha(x, y) = 1 \ \ \forall (x, y).$$

This process produces samples from the target distribution.

MH Demonstration: Bivariate Gaussian

The target density is a bivariate unit normal distribution $\pi(\theta|y) = N(\theta|0, I)$ where I is the 2×2 identity matrix.

The proposal distribution is also a bivariate Gaussian, which is centred at the current iteration:

$$Q(\theta^t|\theta^{t-1}) = N(\theta^\star|\theta^{t-1}, 0.2^I).$$

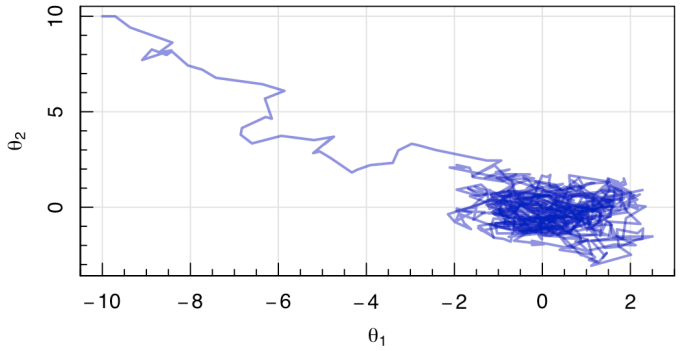
This is therefore a random walk proposal with symmetric probability.

```
#Example of writing our own MH algorithm in R
#Load the "multivariate Normal" package
library(mvtnorm)
#Write our Bivariate Normal MH function
bvnm.MH<-function(n,start) {
  #Initialise the variables
  X<-Y<-rep(NA,n)
  X[1]<-start[1]
  Y[1]<-start[2]
  Id<-diag(2) # The identity matrix
  #Loop over our iterations
  for (i in 2:n) {
    #Generate a proposal point
    prop<-rmvnorm(1,c(X[i-1],Y[i-1]),0.2*Id)
    #Compute the acceptance probability pi(y)/pi(x)
    accept<-dmvnorm(prop,c(0,0),Id)/
            dmvnorm(c(X[i-1],Y[i-1]),c(0,0),Id)
    #Draw a random univariate
    u<-runif(1)
    #Test the acceptance criteria
    if (u<accept) {
      #Accept the new data point
      X[i]<-prop[1]
      Y[i]<-prop[2]
    } else {
      X[i]<-X[i-1]
      Y[i]<-Y[i-1]
    }
  }
  return(cbind(X,Y))
}
```

MH Demonstration: Bivariate Gaussian

Now that we have our MH algorithm set up, we can run a chain:

```
#Running our MH Code
chain<-bvnm.MH(1e3,start=c(-10,10))
magplot(chain,type='l',col=hsv(h=2/3,v=0.75,alpha=0.5),lwd=2,xlab=expression(theta
        [1]),
       ylab=expression(theta[2]))
```



Important Questions in MCMC

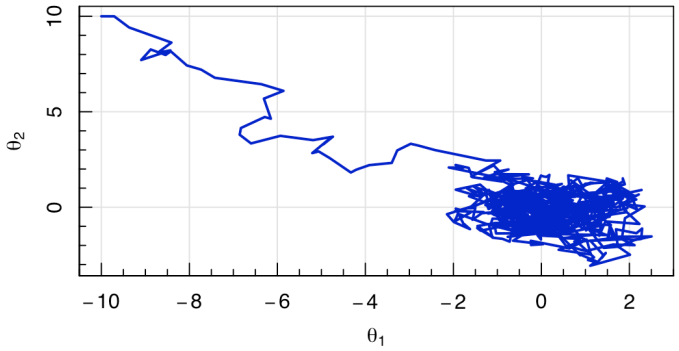
Iterative simulation from the posterior adds two primary difficulties to simulation inference. Namely:

1. How long does a chain need to run before it is “long enough”?
2. How many samples do we need to perform unbiased inference?

There are two primary ways of dealing with these questions.

1. Burn in
2. Thinning

Burn in



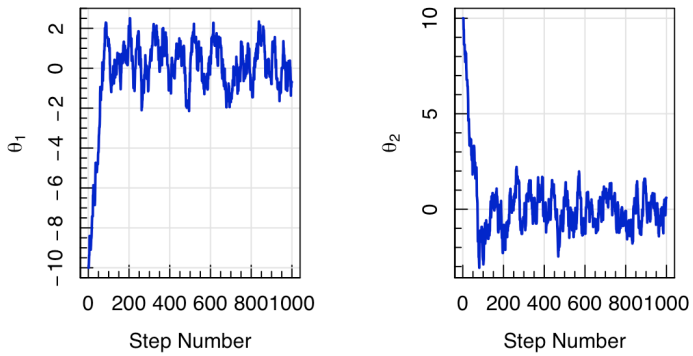
If the chain has not been run for a sufficiently large number of iterations, the simulated samples may be grossly unrepresentative of the posterior that we wish to infer.

The long trail of samples that travel from the starting guess into the main cloud of samples is a principle example.

These samples are known as the **burn in**, and should be discarded.

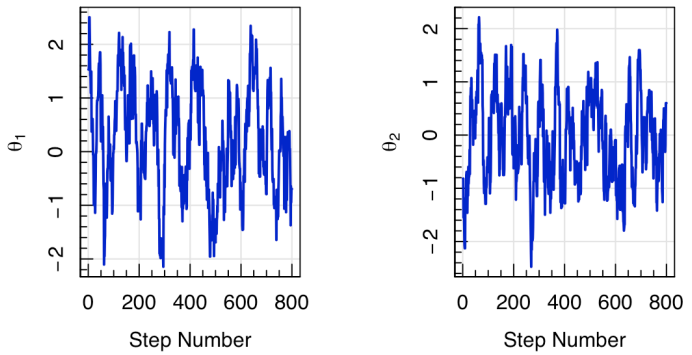
Burn in

```
#Burn in
layout(cbind(1,2))
magplot(chain[,1],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[1]))
magplot(chain[,2],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[2]))
```



Particularly for small numbers of samples (i.e. short chains) these burn-in samples can be pathological. However in all chains they clearly do not sample the target distribution, so should be discarded.

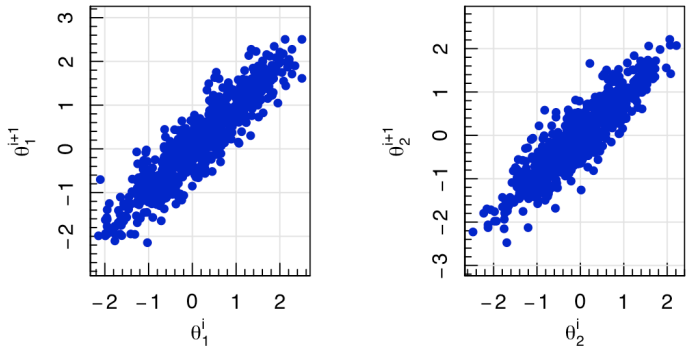
```
#Removing Burn in
chain<-chain[-200:0,]
layout(cbind(1,2))
magplot(chain[,1],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[1]))
magplot(chain[,2],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[2]))
```



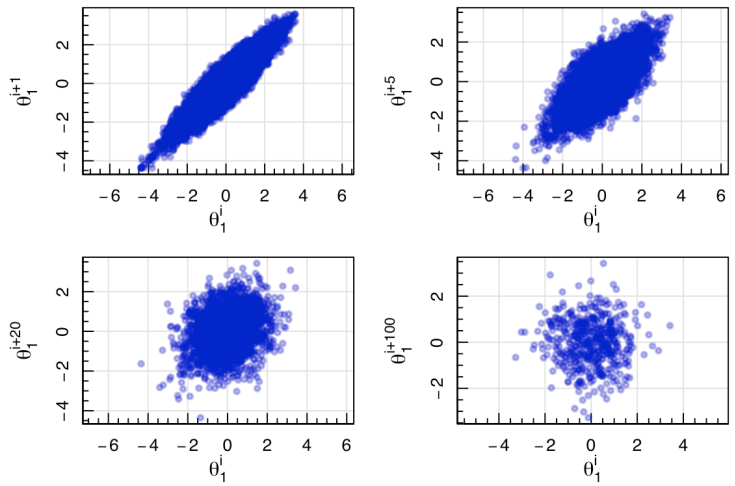
Thinning

We want our final posterior samples to be i.i.d observations from the target distribution. However, the iterative nature of the MCMC means that the sequence of draws is **not** independent.

```
#Correlation in Samples
layout(cbind(1,2))
magplot(chain[-1,1],chain[-nrow(chain),1],pch=20,col='blue3',lwd=2,
        ylab=expression(theta[1]^"i+1"),xlab=expression(theta[1]^i),asp=1)
magplot(chain[-1,2],chain[-nrow(chain),2],pch=20,col='blue3',lwd=2,
        ylab=expression(theta[2]^"i+1"),xlab=expression(theta[2]^i),asp=1)
```



We can circumvent this by **thinning** the samples to every n^{th} sample (but of course this necessitates a longer chain).



So the thinning to every
$\sim 20^{\text{th}}$ sample is reasonably independent, without producing a significant decrease in the accepted number of samples (as with the
100^{th} sample thinning).

A Bayesian MH Problem

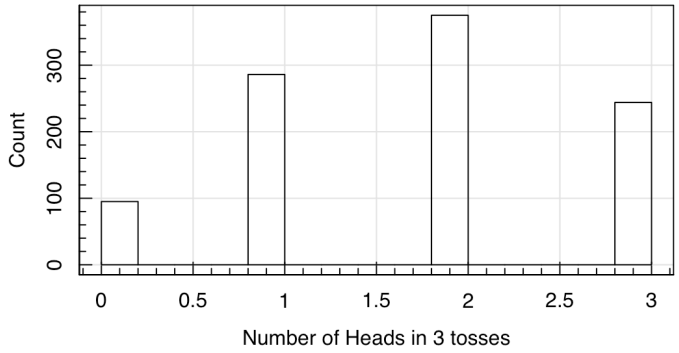
Let's go back to our old standard problem, coin tosses, but with a bit of a twist.

We are given a bag of coins, and are told that a certain fraction of the coins are biased.

We are not told how many coins are biased, nor what their biasing is.

We draw a random coin from the bag 1000 times, toss it 3 times, and record the number of observed heads.

```
#Generate and Plot some data
set.seed(42)
data<-simulate.throws(1e3)
maghist(data,xlab='Number of Heads in 3 tosses',ylab='Count',verbose=FALSE)
```



A Bayesian MH Problem

We want to infer, given the observations, the likely values of the fraction of coins in the bag which are biased θ_1, and the amount of biasing that affects each biased coin θ_2.

That is, we want to know the posterior distribution $P(\theta_1, \theta_2|x)$.

Recall the procedure for performing Bayesian Inference:

1. Specification of the likelihood model $p(x|\theta)$;
2. Determination of the prior $p(\theta)$;
3. Calculation of the posterior distribution $p(\theta|x)$; and
4. Draw inference from the posterior distribution.

Step 1: Specify the likelihood model

In our case, the likelihood model consists of 3 parts:

1. The probability of drawing a biased coin from the bag
2. The probability of a biased coin showing x heads in 3 trials
3. The probability of a fair coin showing x heads in 3 trails

All of these probabilities can be described using binomials:

1. $\text{biased coin}|\theta_1 \sim \text{Bin}(1, \theta_1)$
2. $x \text{ heads}|\text{biased coin}, \theta_2 \sim \text{Bin}(3, \theta_2)$
3. $x \text{ heads}|\text{fair coin} \sim \text{Bin}(3, 0.5)$

So we can construct our likelihood from these probabilities:

$$P(X|\theta_1,\theta_2) = P(X|n,0.5)(1-\theta_1) + P(X|n,\theta_2)\theta_1$$
$$= [\text{Bin}(3,0.5)(1-\theta_1) + \text{Bin}(3,\theta_2)\theta_1]$$

for $\theta_1, \theta_2 \in [0,1]$.

Which in **R** code is simply:

```
#Define a handy "interval" function
interval<-function(x,min,max) ifelse(x>=min & x<=max,x,0)
likelihood<-function(x,n,theta1,theta2) {
  #Ensure that theta's are in 0,1
  theta1=interval(theta1,0,1)
  theta2=interval(theta2,0,1)
  #Calculate the value of the likelihood at this theta1,theta2
  val=dbinom(x=x,size=n,prob=0.5)*(1-theta1)+
      dbinom(x=x,size=n,prob=theta2)*theta1
  #Return the value
  return(val)
}
```

Step 2: Specify the Priors

We have no idea what the values of the biases on the coins are, nor about what fraction of coins in the bag are biased.

Therefore, we ought to impose an uninformative uniform prior on all possible values of θ_1, θ_2.

$$P(\theta_i) = \begin{cases} 1 & 0 \leq \theta_i \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

```
prior<-function(theta1,theta2) {
  #Calculate the value of the prior at theta1,theta2
  val=dunif(x=theta1,min=0,max=1)*
      dunif(x=theta2,min=0,max=1)
  #Return the value
  return(val)
}
```

Step 3: Calculate the posterior distribution

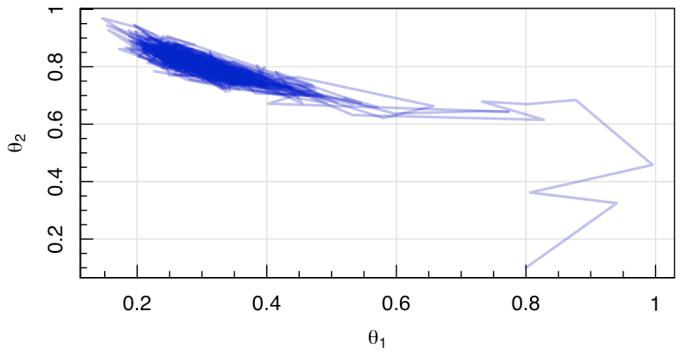
This is where our sampler comes in. We know from Bayes' theorem that the posterior is proportional to the likelihood times the prior, and we want to sample from this unknown distribution to produce points that we can use to perform inference about the likely values of θ_1 and θ_2.

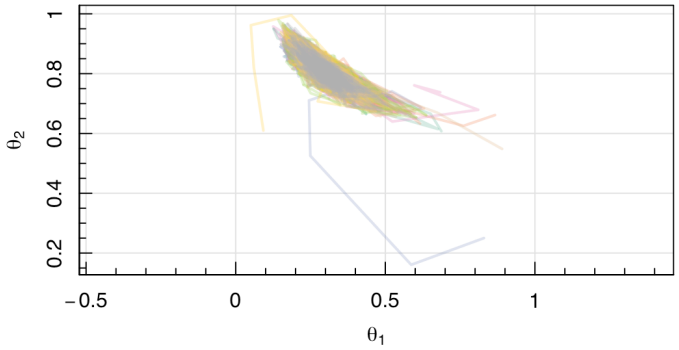
```
coins.MH<-function(n,start) {
  #Initialise the variables
  X<-Y<-rep(NA,n)
  X[1]<-start[1]
  Y[1]<-start[2]
  #Loop over our iterations
  Id<-diag(2)
  #Define the posterior evaluation
  log.posterior<-function(x,n,theta1,theta2)
    return(log(prior(theta1,theta2))+sum(log(likelihood(x=x,n=n,theta1=theta1,thet
        a2=theta2))))
  #Loop over the steps in the chain
  for (i in 2:n) {
    #Generate a proposal point
    prop<-rmvnorm(1,c(X[i-1],Y[i-1]),0.05*Id)
    #Compute the acceptance probability pi(y)/pi(x)
    #Use logarithms to avoid rounding errors
    accept<-exp(log.posterior(x=data,n=3,theta1=prop[1],theta2=prop[2])-
                log.posterior(x=data,n=3,theta1=X[i-1] ,theta2=Y[i-1] ))
    if (!is.finite(accept)) { accept<-0 }
    #Draw a random univariate
    u<-runif(1)
    #Test the acceptance criteria
    if (u<accept) {
      #Accept the new data point
      X[i]<-prop[1]
      Y[i]<-prop[2]
    } else {
      #Stay where we are
      X[i]<-X[i-1]
      Y[i]<-Y[i-1]
    }
  }
  return(cbind(X,Y))
}
```

Step 3: Calculate the posterior distribution

We now run our chain to construct our posterior samples:

```
#Running our MH Code
set.seed(42)
data<-simulate.throws(1e3)
chain<-coins.MH(1e4,start=c(0.8,0.1))
magplot(chain,type='l',col=seqinr::col2alpha('blue3',0.3),lwd=2,
        xlab=expression(theta[1]),ylab=expression(theta[2]))
```





What are the three things that we've forgotten?

1. Multiple starting points
2. Burn in removal
3. Trimming

Step 3: Calculate the posterior distribution

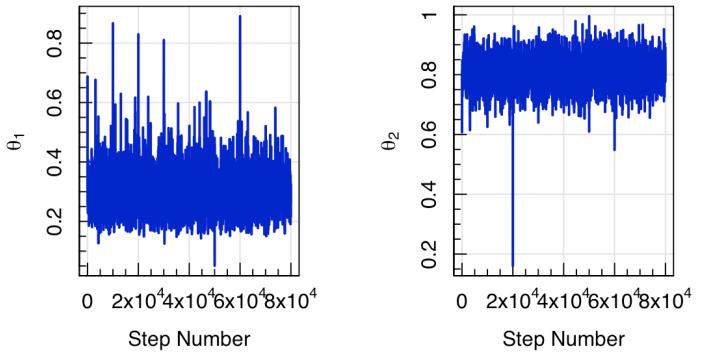
So let's do those. First our multiple starting points:

```
#Building more samples
set.seed(42)
library(foreach)
library(doParallel)
#Using 8 CPUs
registerDoParallel(cores=8)
#Run a chain with 8 different starting points
chain<-foreach(i=1:8,.combine='rbind',.inorder=TRUE)%dopar%{
  return(
    cbind(coins.MH(1e4,start=c(runif(1),runif(1))),i)
  )
}
#Set up the plot
magplot(chain[,1:2],type='n',pch=20,col=seqinr::col2alpha('blue3',0.3),lwd=2,
        xlab=expression(theta[1]),ylab=expression(theta[2]),asp=1)
#Plot each chain in a different colour
colours<-RColorBrewer::brewer.pal(8,"Set2")
for (i in 1:max(chain[,3])) {
  lines(chain[which(chain[,3]==i),1:2],
        col=seqinr::col2alpha(colours[i],0.3),lwd=2)
}
```

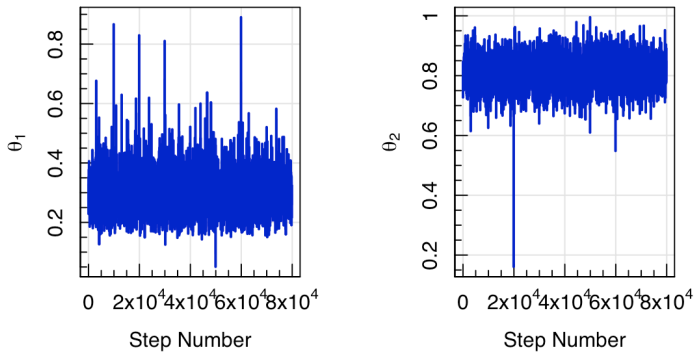
Step 3: Calculate the posterior distribution

Second, remove the burn-in:

```
#Cleaning up our posterior samples: Burn in removal
layout(cbind(1,2))
magplot(chain[,1],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[1]))
magplot(chain[,2],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[2]))
```



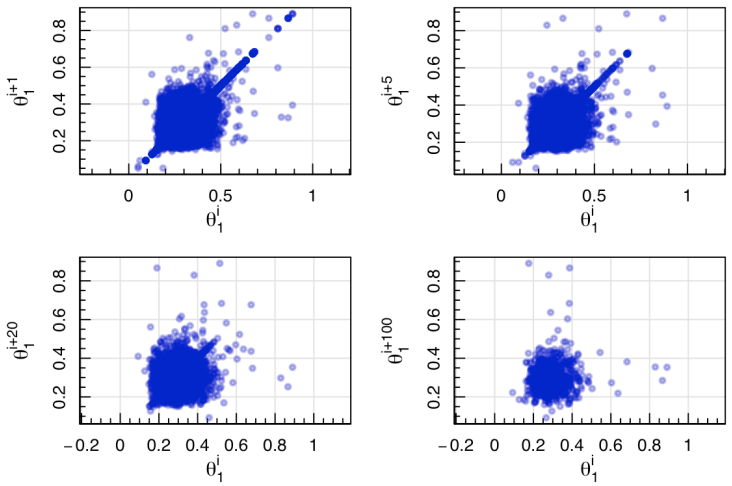
```
chain<-chain[-1e2:0,]
magplot(chain[,1],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[1]))
magplot(chain[,2],type='l',col='blue3',lwd=2,xlab="Step Number",
        ylab=expression(theta[2]))
```



Step 3: Calculate the posterior distribution

And third, our thinning:

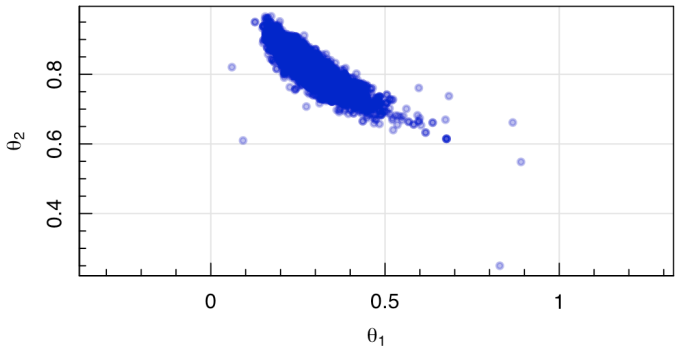
```
#Cleaning up our posterior samples: Thinning
layout(matrix(1:4,2,2,byrow=TRUE))
par(mar=c(3,4,1,1))
chain.fifth<-thin(chain,by=5)
chain.twent<-thin(chain,by=20)
chain.hundr<-thin(chain,by=100)
magplot(chain[-1,1],chain[-nrow(chain),1],pch=20,col=colour,lwd=2,
        ylab=expression(theta[1]^"i+1"),xlab=expression(theta[1]^i),asp=1)
magplot(chain.fifth[-1,1],chain.fifth[-nrow(chain.fifth),1],pch=20,col=colour,lwd=
         2,
        ylab=expression(theta[1]^"i+5"),xlab=expression(theta[1]^i),asp=1)
magplot(chain.twent[-1,1],chain.twent[-nrow(chain.twent),1],pch=20,col=colour,lwd=
         2,
        ylab=expression(theta[1]^"i+20"),xlab=expression(theta[1]^i),asp=1)
magplot(chain.hundr[-1,1],chain.hundr[-nrow(chain.hundr),1],pch=20,col=colour,lwd=
         2,
        ylab=expression(theta[1]^"i+100"),xlab=expression(theta[1]^i),asp=1)
```



Step 4: Draw inference from the posterior distribution

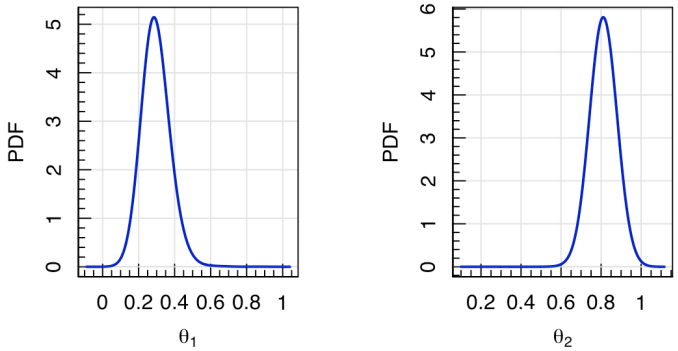
With our cleaned samples we can now do our posterior inference:

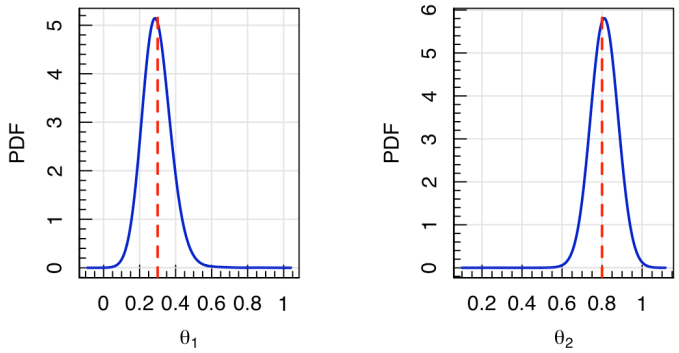
```
#Final posterior samples
chain<-thin(chain,10)
magplot(chain[,-3],type='p',pch=20,col=seqinr::col2alpha('blue3',0.3),lwd=2,
        xlab=expression(theta[1]),ylab=expression(theta[2]),asp=1)
```



Recall that these posterior samples allow us to trivially compute the marginal distributions:

```
#Final posterior samples
layout(cbind(1,2))
magplot(density(chain[,1],bw=0.05),col='blue3',lwd=2,
        xlab=expression(theta[1]),ylab="PDF")
magplot(density(chain[,2],bw=0.05),col='blue3',lwd=2,
        xlab=expression(theta[2]),ylab="PDF")
```





Bayesian Stats (and MCMC) works!

And the truth?

Thirty percent of the coins are biased in reality ($\theta_1 = 0.3$), and they have a bias of 80% ($\theta_2 = 0.8$):

```
#Final posterior samples
layout(cbind(1,2))
magplot(density(chain[,1],bw=0.05),col='blue3',lwd=2,
        xlab=expression(theta[1]),ylab="PDF")
abline(v=0.3,col='red',lty=2,lwd=2)
magplot(density(chain[,2],bw=0.05),col='blue3',lwd=2,
        xlab=expression(theta[2]),ylab="PDF")
abline(v=0.8,col='red',lty=2,lwd=2)
```