Outline

The Metropolis Algorithm

Random Walk Metropolis

Independence Proposal

The Log Normal Prior for σ

Metropolis within Gibbs

Regresssion with the Log-normal Prior

The Metropolis Algorithm

Hierarchical modeling allows us to construct complex models out of simple pieces.

The Gibbs sampler then allows us to draw the posterior of each "piece" conditional on the rest.

It is often convenient to structure the model (in particular the choice of prior) so that the conditionals are "conditinally conjugate".

The draws are then made using standard families of distributions.

Sometimes our modeling problem just does not lend itself to the conjugate structure.

In this case we need simple yet broadly applicable methods for drawing from a distribution only being able to compute (typically) the posterior up to a proportionality constant.

For example, we could use a grid.
Drawing the conditionals using a grid is known as the "griddy Gibbs sampler".

The Metropolis-Hastings algorithm is a general approach for drawing from a distribution when we can compute the density up to a proportionality constant.

Our basic formula often allows us to do this:

$$p(\theta \mid y) \propto f(y \mid \theta)\, p(\theta).$$

To compute a conditional we have

$$p(\theta_1 \mid \theta_2, y) \propto p(\theta_1, \theta_2 \mid y) \propto f(y \mid \theta)\, p(\theta).$$

The MH is a Markov Chain Monte Carlo method in that rather than generating iid draws, it is a method for construction a Markov chain such that the stationary distribution is the one we want to draw from.

The algorithm works as follows.

(i)

We can an compute $p(\theta) \propto f(\theta)$, where f is the density (or mass function in the discrete case) we wish to draw from.

(ii)

We specify a markov chain definied by the transition kernel $q(\theta_{t+1} \mid \theta_t)$.

(iii)

We construct a new Markov chain using q and p.

Suppose we are currently at θ_t.

(1) draw $\theta \sim q(\theta \mid \theta_t)$.

(2) Compute

$$\alpha = \min \left\{ 1, \frac{p(\theta)\, q(\theta_t \mid \theta)}{p(\theta_t)\, q(\theta \mid \theta_t)} \right\}$$

(3)
With probability α, $\theta_{t+1} = \theta$, else $\theta_{t+1} = \theta_t$.

Notes:

(i)
The MH modifies the proposal chain q by *repeating* some values.

(ii)
The intuition behind $p(\theta)/p(\theta_t)$ is clear, if the proposed θ is more likely under the target distribution than the current θ_t then maybe be should go there.

(iii)
The $q(\theta_t \mid \theta)$ on the top says it is ok to go if we can get back.

(iv)
The $q(\theta \mid \theta_t)$ on the bottom says if it is hard to get there then we had better take advantage of this opportunity.

Random Walk Metropolis

To use the MH, you have to pick q.

A common choice is for q to be a *random walk*:

$$\theta = \theta_t + \epsilon$$

where ϵ is draw iid from a specified density $g(\epsilon)$. In this case

$$q(\theta \mid \theta_t) = g(\theta - \theta_t).$$

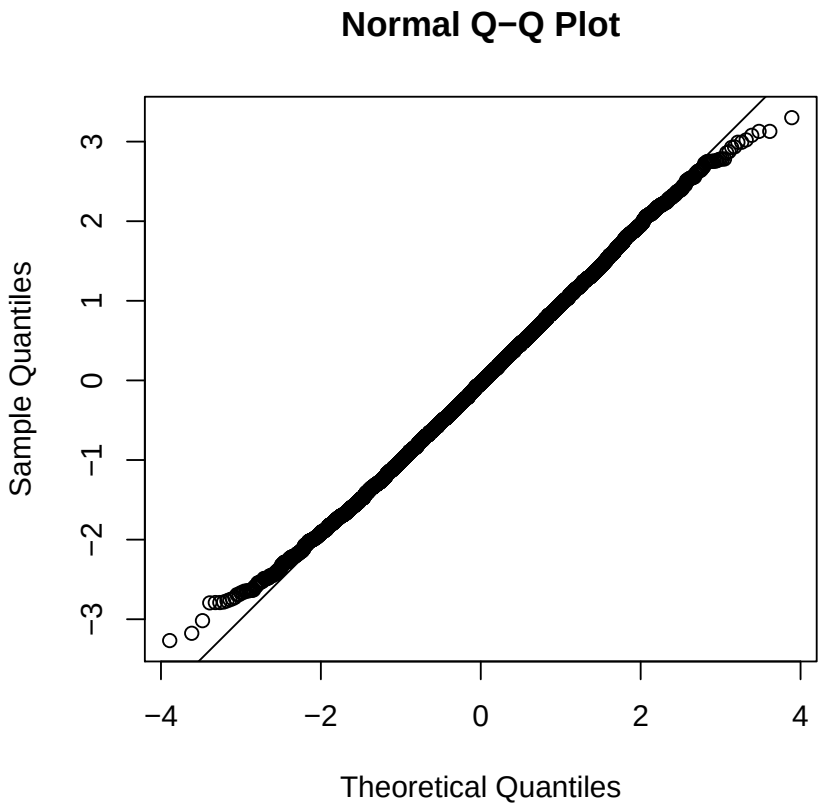
If g is symmetric ($g(\epsilon) = g(-\epsilon)$) then $q(\theta \mid \theta_t) = q(\theta_t \mid \theta)$ so that

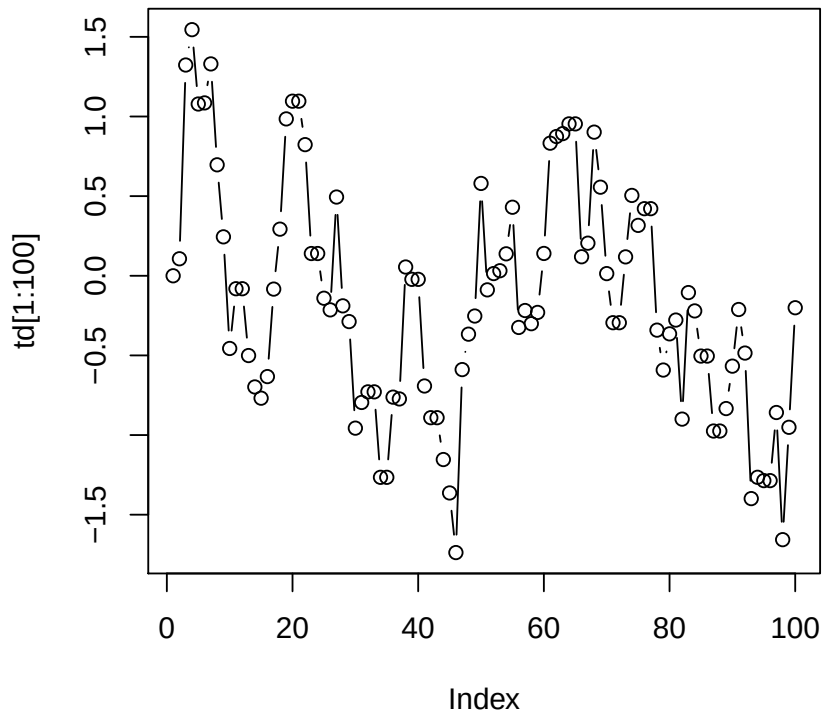
$$\alpha = \min \left\{ 1, \frac{p(\theta)}{p(\theta_t)} \right\}$$

The code below uses the random walk MH to draw from the standard normal. The proposal is the current value plus normal "error".

```
set.seed(99)
nd = 10000 #number of MH iterations (draws)
td = rep(0,nd) # storage for theta draws
sigma = .5 #q: theta_{t+1} = \theta_t + sigma*Z
for(t in 2:nd) {
   theta = td[t-1] + sigma*rnorm(1)
   alpha = min(1,dnorm(theta)/dnorm(td[t-1]))
   if(rbinom(1,1,alpha)) {
      td[t] = theta
   } else {
      td[t] = td[t-1]
   }
}
```

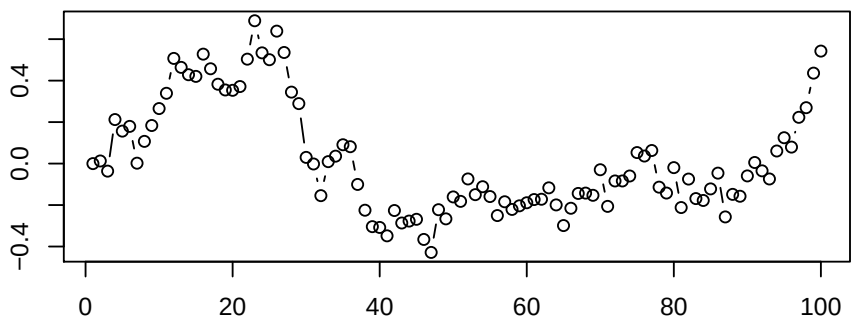
Here is a look at the draws (td).
On left is the normal QQ-plot and on right is the first 100 draws.



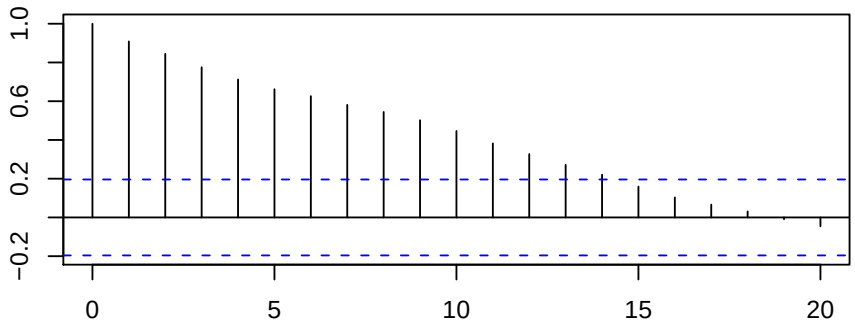


The role of σ:

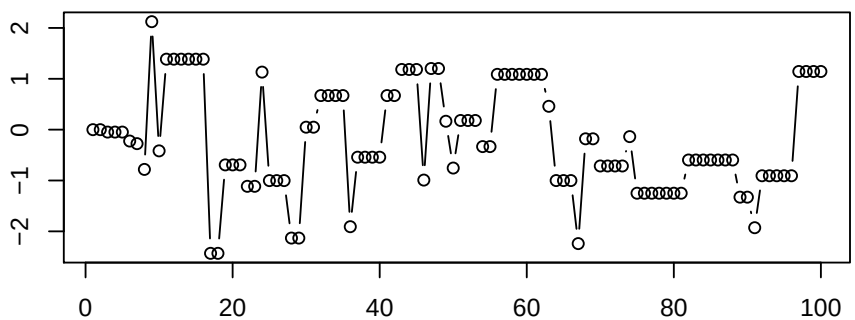
If σ is too small, we make a lot of small moves that are accepted.
If σ is too big, then we make bigger moves, but some are rejected.



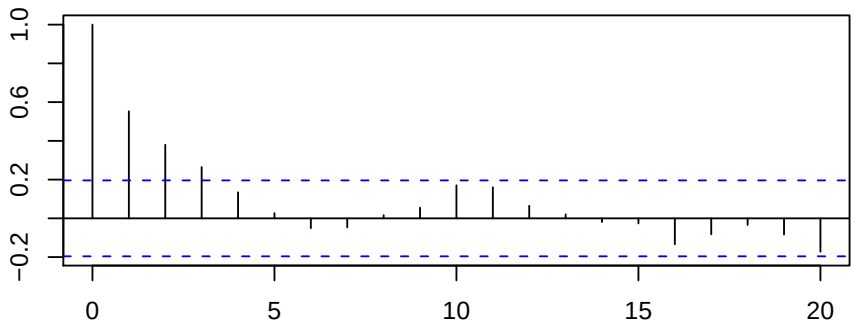
sigma=.1



sigma=.1



sigma=3.0



sigma=3.0

Independence Proposal

The other very popular proposal is iid draws.

$$q(\theta_{t+1} \mid \theta_t) = g(\theta_{t+1}).$$

Every proposal is just drawn independently from g.

This gives:

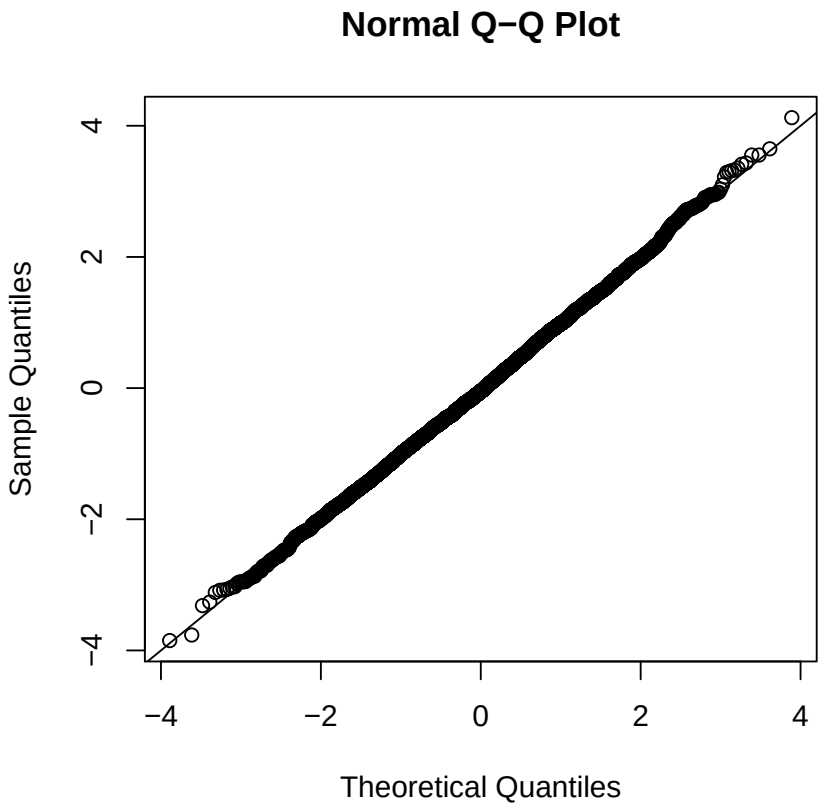
$$\alpha = \min\left\{1, \frac{p(\theta)\, g(\theta_t)}{p(\theta_t)\, g(\theta)}\right\} = \min\left\{1, \frac{w(\theta)}{w(\theta_t)}\right\}$$

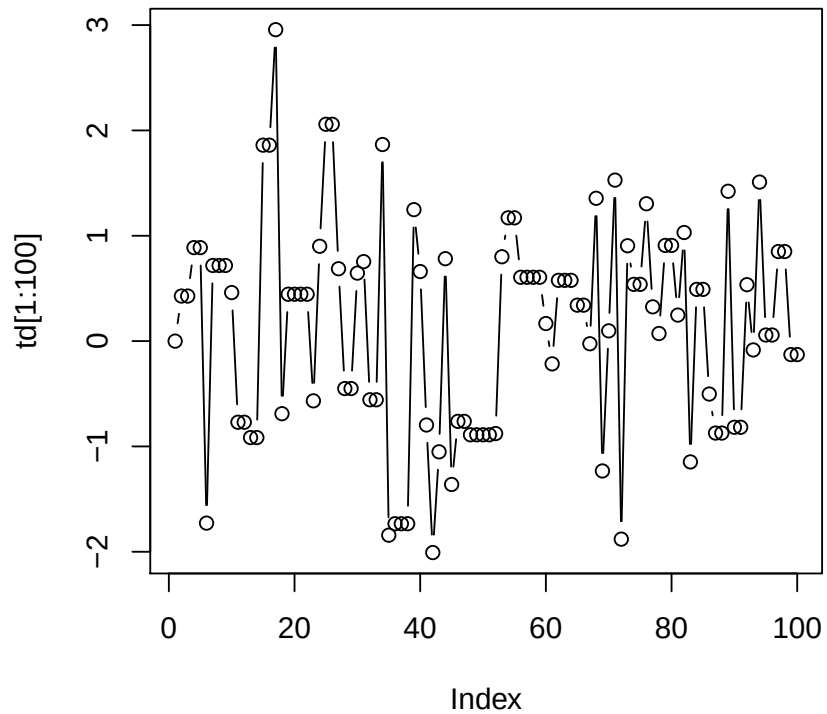
where $w(\theta) = \frac{p(\theta)}{g(\theta)}$.

Here is code to use the Metropolis algorithm to draw from the standard normal using a $N(0, \sigma^2)$ independence proposal.

```
set.seed(99)
nd = 10000 #number of MH iterations (draws)
td = rep(0,nd) # storage for theta draws
sigma = 2.0 #q: theta_{t+1} = sigma*Z
for(t in 2:nd) {
   theta =  sigma*rnorm(1)
   thetat = td[t-1]
   wtop=dnorm(theta)/dnorm(theta,sd=sigma)
   wbot=dnorm(thetat)/dnorm(thetat,sd=sigma)
   alpha = min(1,wtop/wbot)
   if(rbinom(1,1,alpha)) {
      td[t] = theta
   } else {
      td[t] = td[t-1]
   }
}
```

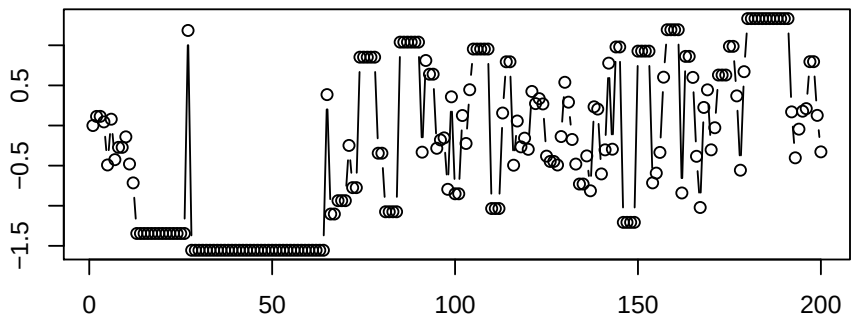
Here is a look at the draws (td).
On left is the normal QQ-plot and on right is the first 100 draws.



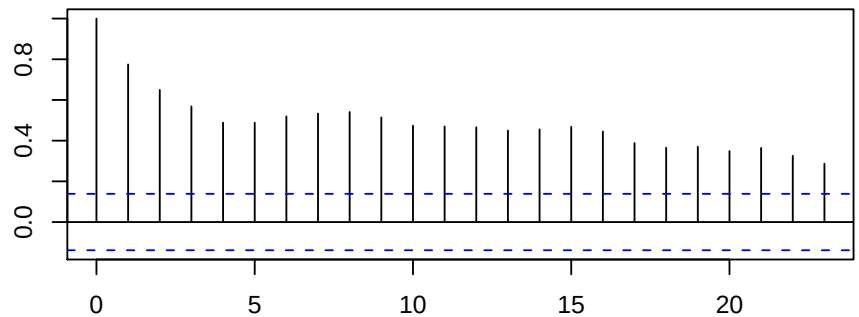


The role of σ:

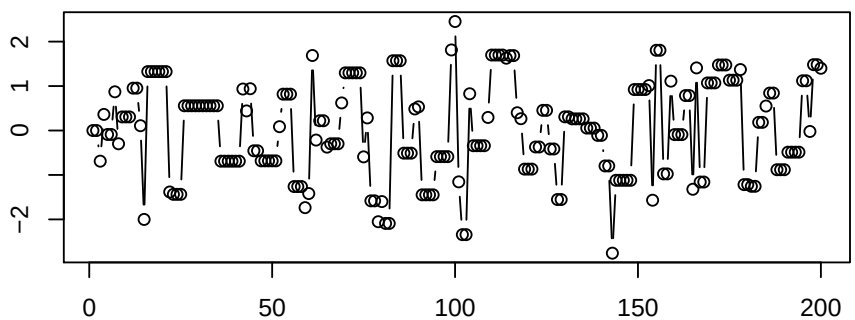
If σ is too small, we need to really build up the tails by repeating draws.
If σ is too big, we need to repeat in the center to build that up.



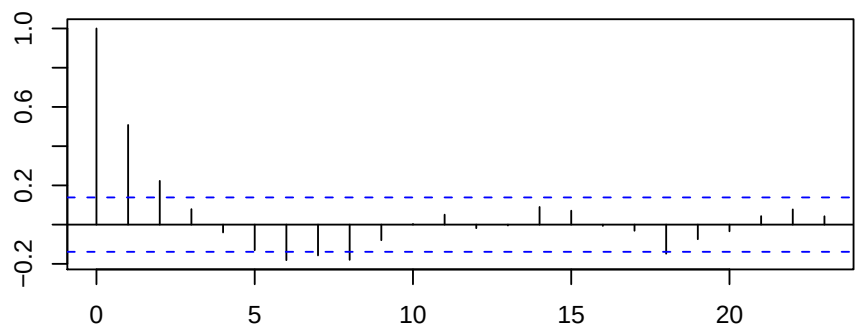
sigma=.5



sigma=.1



sigma=3.0



sigma=3.0

The Log Normal Prior for σ

The standard inverted-χ^2 prior for a normal variance has some drawbacks.

For small ν the right tail is extreme and the left tail has a "dead-area".

An obvious alternative prior is the log-normal:

$$\theta \equiv \log(\sigma) \sim N(\mu_\theta, \sigma_\theta^2).$$

We could work with either θ or σ.

Let's use θ.

So,

$$\sigma = e^{\theta},$$

and our model is:

$$\epsilon_i \sim N(0, (e^{\theta})^2), \text{ iid}, \quad \theta \sim N(\mu_\theta, \sigma_\theta^2).$$

Here we assume we observe the ϵ_i.

This function computes the log of the posterior at θ given data e and prior with mean mt and standard deviation st (μ_θ = mt, σ_θ=st).

```
lpost = function(theta,e,mt,st) {
#compute log posterior for e_i ~ N(0,exp(theta)^2), theta ~ N(mt,st^2)
n=length(e)
sigma = exp(theta)
s2 = sigma*sigma
S = sum(e^2)
llik = -n*theta -.5*S/s2
z = (theta-mt)/st
lpri = -log(st) -.5*z*z
return(llik+lpri)
}
```

Simulate some data and draw from the prior.

```
#simulate data
sigma = .5
nd=100
set.seed(99)
eps = rnorm(nd,sd=sigma)

#prior, theta ~ N(mt,st^2)
mt = log(sigma) #cheat by using true value
st = .2
sdpri = exp(rnorm(10000,mt,st)) #draws from prior
```

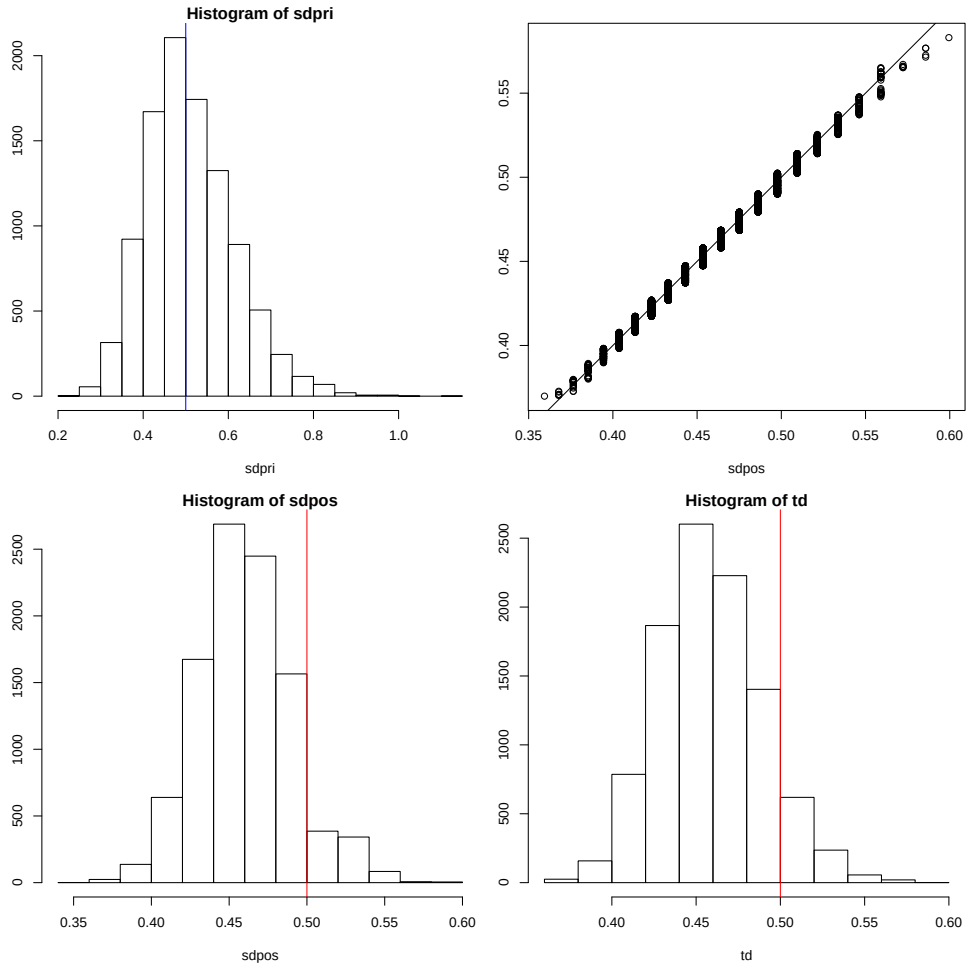
Draw from the posterior using a grid.

```
Lg = log(.1); Ug = log(1) #lower and upper end points of grid
ng = 100 # number of grid points
tg = seq(from=Lg,to=Ug,length.out=ng) #the grid
lp = rep(0,ng) #store evaluations of log post on the grid
for(i in 1:ng) lp[i] = lpost(tg[i],eps,mt,st)
lp = lp-max(lp)
postv = exp(lp); postv = postv/sum(postv)
sdpos = exp(sample(tg,size=10000,replace=T,prob=postv))
```

Draw from the posterior using random walk Metropolis.

```
sprop = .1 #standard deviation for q
nd=10000
td = rep(0,nd)
td[1] = log(sigma) # start at true value
oldlp = lpost(td[1],eps,mt,st)
for(t in 2:nd) {
   theta = td[t-1] + sprop*rnorm(1)
   newlp = lpost(theta,eps,mt,st)
   rat = exp(newlp-oldlp)
   alpha = min(1,rat)
   if(rbinom(1,1,alpha)) {
      td[t] = theta; oldlp = newlp
   } else {
      td[t] = td[t-1]
   }
}
td = exp(td)
```

Plot results.
(1,1): draws from prior, (1,2): qqplot of grid draws vs. MH draws
(2,1): hist of grid draws , (2,2): hist of MH draws.



Metropolis within Gibbs

Many Bayesian applications use both the Gibbs sampler and the MH.

Suppose we have the simple Gibbs structure:

$$\theta_1 \mid \theta_2, y, \quad \theta_2 \mid \theta_1, y.$$

We could, for example, replace the draw of θ_2 with an iteration of a MH Markov chain having stationary distribution $p(\theta_2 \mid \theta_1, y)$ where the value of θ_1 is simply our current value. We also use the current value of θ_2 in the MH chain.

Note:

Let's trace through MH within Gibbs to make sure the joint posterior is the stationary distribution of the Markov Chain.

Suppose (θ_1^t, θ_2^t) is a draw from $p(\theta_1, \theta_2 \mid y)$.

Replace θ_1^t with a draw from $\theta_1^{t+1} \sim p(\theta_1 \mid \theta_2^t, y)$.

Then $(\theta_1^{t+1}, \theta_2^t)$ is still a draw from $p(\theta_1, \theta_2 \mid y)$.

So, θ_2^t is a draw from $p(\theta_2 \mid \theta_1^{t+1}, y)$.

Now draw θ_2^{t+1} using a Markov chain (eg MH) having stationary distribution $p(\theta_2 \mid \theta_1, y)$ and previous value θ_2^t.

Then θ_2^{t+1} is a draw from $p(\theta_2 \mid \theta_1^{t+1}, y)$.

Then $(\theta_1^{t+1}, \theta_2^{t+1})$ is a draw from $p(\theta_1, \theta_2 \mid y)$.

Using the same logic, we can see that we can replace a draw from any conditional in a Gibbs setup with an iteration of an Markov chain having the conditional as its stationary distribution.

More generally, we can combine different chains in different ways.

Example:

We can use mixtures of Markov Chains.

We have Markov chains M_1 and M_2.

That is, we can draw $\theta_{t+1} \sim M_i(\theta_t)$.

We can make up a new mixture Markov chain

$$M = \alpha M_1 + (1 - \alpha) M_2.$$

This means with probability α draw using M_1, otherwise draw using M_2.

Regresssion with the Log-normal Prior

Let's write an MCMC for the model:

$$Y = X\beta + \epsilon, \epsilon \sim N(0, (e^{\theta})^2 I),$$

$$\beta \sim N(\bar{\beta}, A^{-1}),\ \theta \sim N(\mu_{\theta}, \sigma_{\theta}^2).$$

We'll draw $\beta \mid \theta$ and $\theta \mid \beta$ and use the MH to draw θ.

First, let's slightly rewrite the log-likelood function to depend only on the sufficient statistics since we don't want to compute these more often than we have to.

```
lpostS = function(theta,S,n,mt,st) {
#compute log posterior for e_i ~ N(0,exp(theta)^2), theta ~ N(mt,st^2)
#S = sum(e^2)
#n = length(e)
llik = -n*theta -.5*S*exp(-2*theta)
z = (theta-mt)/st
lpri = -log(st) -.5*z*z
return(llik+lpri)
}
```

Now we write the MH draw of $\theta = \log(\sigma)$ as a function.

```
dtheta = function(thetao,e,mt,st,tau) {
#function to do MH draw of theta = log(sigma)
#thetao: previous value
#eps: data eps_i ~N(0,\sigma^2), sigma = exp(theta)
#mt,st: theta ~ N(mt,st^2)
#tau: q is theta_{t+1} = theta_t + tau Z
S = sum(e^2)
n = length(e)
theta = thetao + tau*rnorm(1)
oldlp = lpostS(thetao,S,n,mt,st)
newlp = lpostS(theta,S,n,mt,st)
rat = exp(newlp-oldlp)
alpha = min(1,rat)
if(rbinom(1,1,alpha)) {
   return(theta)
} else {
   return(thetao)
}
}
```

Now we write the $\beta \mid \sigma$ draw as a function.

```
dbeta = function(n,xty,xtx,Ab,A,sigma) {
#draw beta | sigma
#n: number of obs
#xty: X'y, xtx: X'X
#beta ~ N(b,A^{-1})
p=length(xty)
V = solve(xtx/sigma^2 + A)
m = V %*% (xty/sigma^2 + Ab)
L = t(chol(V))
return(m+L %*% matrix(rnorm(p),ncol=1))
}
```

```
#simulate data
set.seed(99)
n=10
x=rnorm(n)
X = cbind(rep(1,n),x)
sigma = 1
beta=c(1,2)
y = X %*% beta + sigma*rnorm(n)
xty = t(X) %*% y
xtx = t(X) %*% X

#choose prior
b=rep(0,2)
A=diag(rep(.1,2))
Ab = A %*% b
mt = 0; st=.2
```

```
#run MCMC
nd = 5000
td = rep(0,nd) #draws of theta
bd = matrix(0.0,nd,ncol(X)) #draws of beta (rows)
bd[1,]=beta
tau = .1
for(t in 2:nd) {
e = y - X %*% bd[t-1,]
td[t] = dtheta(td[t-1],e,mt,st,tau)
bd[t,] = dbeta(n,xty,xtx,Ab,A,exp(td[t]))
}
```

Here is a plot of the results with n=100 and set.seed(99).