

Bayesian Approaches 2

Last week

Frequentist

- normal distribution
- linear models
- point estimates of parameter values

Bayesian

- any distribution
- any (including nonlinear) model
- rich characterization of posterior distribution over all possible model parameter values

Estimating the Posterior Distribution

- **Analytic**
 - use conjugate priors
 - posterior is defined using hyper-parameters

Estimating the Posterior Distribution

- **Numerical** : Grid approximation by discretizing the prior
- **Numerical** : Markov Chain Monte Carlo (MCMC)

This Week

- **Numerical**
 - an example of a grid approximation approach (by discretizing the prior)
 - Markov Chain Monte Carlo (MCMC)

Grid Approximation

- numerically approximating the **posterior** distribution by defining the **prior** distribution over a **fine grid** of parameter values
- we don't need a mathematical equation (function) to define the prior
- we can specify any shape we want
- no need for formulas, calculus, etc

Bayes' Theorem for Discrete Values

- Binomial example: let's say we want to find the posterior distribution of the binomial parameter θ , which corresponds to the probability of being correct on a single trial
- posterior is:
$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\sum_{\theta} p(D|\theta)p(\theta)}$$
- we decide on a fixed set of possible values of θ to compute the posterior

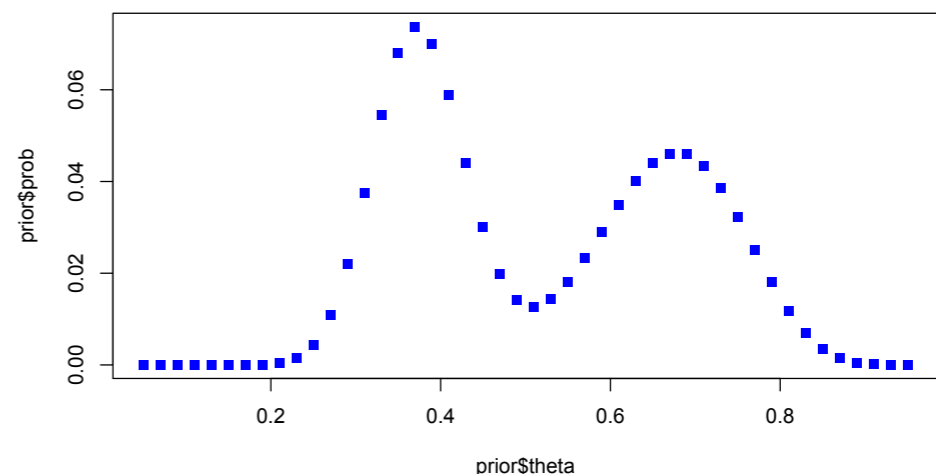
Discrete Prior

- instead of specifying a prior as a mathematical equation (e.g. a Beta function) we can instead just list the discrete values

Computing the Posterior

$$p(\theta_i|D) = \frac{p(D|\theta_i)p(\theta_i)}{\sum_{\theta} p(D|\theta)p(\theta)}$$

- for each value of θ , we compute the posterior
- let's say our data D is $k=20$ successes in $n=40$ trials
- let's say our Prior on θ is defined over a grid of 46 discrete points and is in an R data frame called **prior** with columns **theta** and **prob**



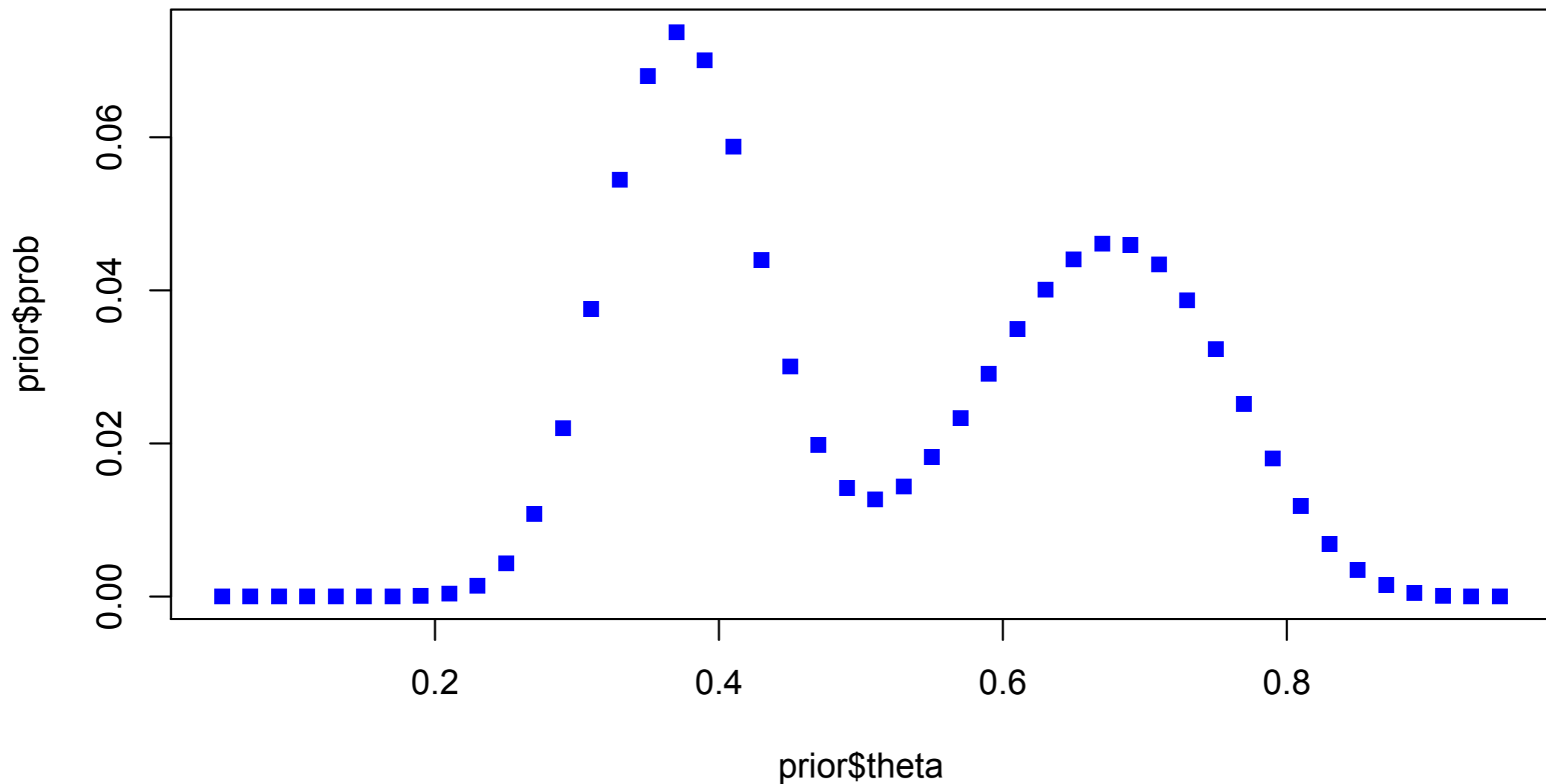
our discrete prior

```
> prior
   theta prob
1  0.05 0.000
2  0.07 0.000
3  0.09 0.000
4  0.11 0.000
5  0.13 0.000
6  0.15 0.000
7  0.17 0.000
8  0.19 0.000
9  0.21 0.000
10 0.23 0.001
11 0.25 0.004
12 0.27 0.011
13 0.29 0.022
14 0.31 0.038
15 0.33 0.054
16 0.35 0.068
17 0.37 0.074
18 0.39 0.070
19 0.41 0.059
20 0.43 0.044
21 0.45 0.030
22 0.47 0.020
23 0.49 0.014
24 0.51 0.013
25 0.53 0.014
26 0.55 0.018
27 0.57 0.023
28 0.59 0.029
29 0.61 0.035
30 0.63 0.040
31 0.65 0.044
32 0.67 0.046
33 0.69 0.046
34 0.71 0.043
35 0.73 0.039
36 0.75 0.032
37 0.77 0.025
38 0.79 0.018
39 0.81 0.012
40 0.83 0.007
41 0.85 0.004
42 0.87 0.002
43 0.89 0.001
44 0.91 0.000
45 0.93 0.000
46 0.95 0.000
```

our discrete prior

$$p(\theta_i|D) = \frac{p(D|\theta_i)p(\theta_i)}{\sum_{\theta} p(D|\theta)p(\theta)}$$

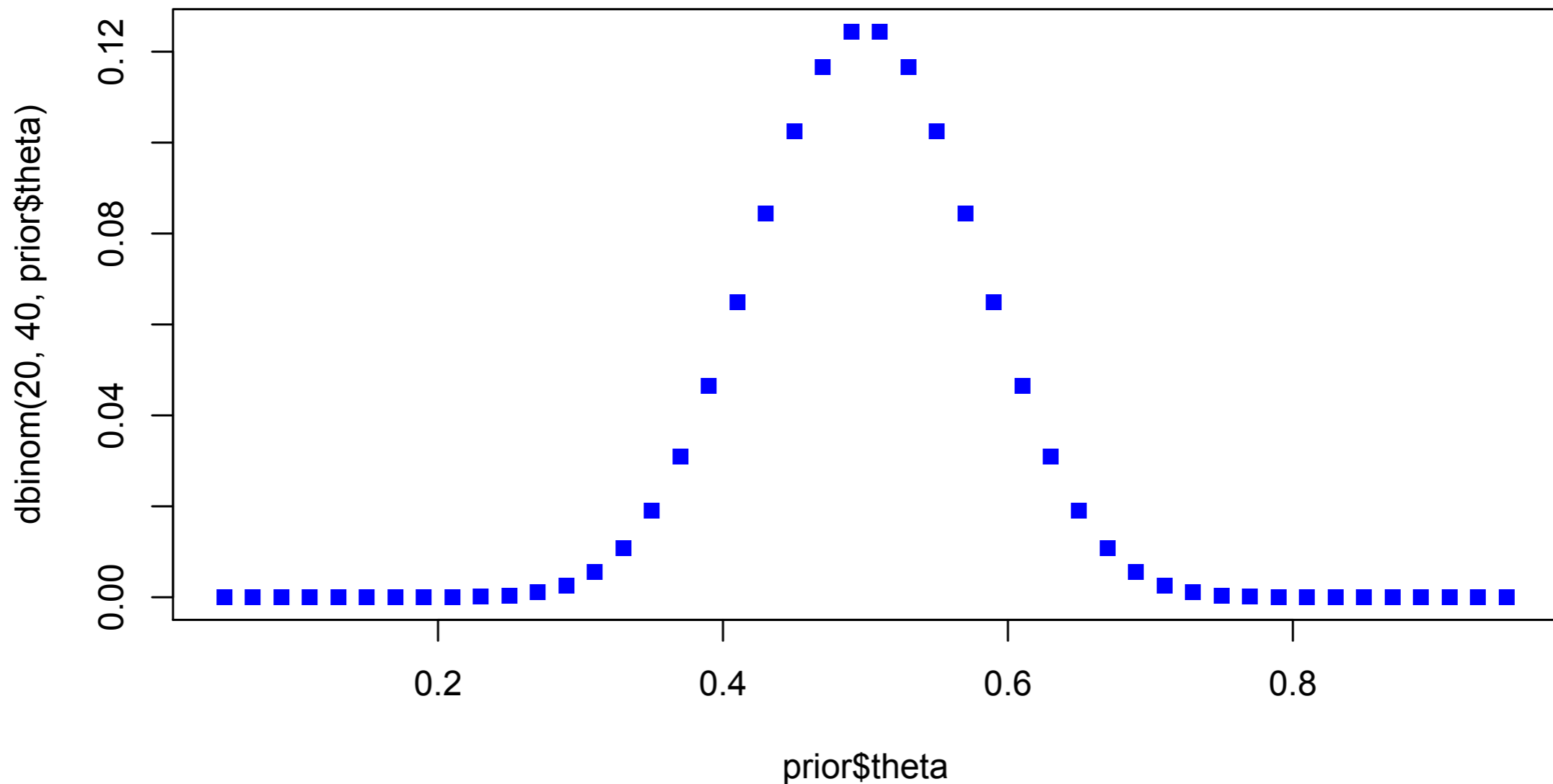
```
> plot(prior$theta, prior$prob, type="p", col="blue", pch=15)
```



the likelihood

$$p(\theta_i | D) = \frac{p(D | \theta_i) p(\theta_i)}{\sum_{\theta} p(D | \theta) p(\theta)}$$

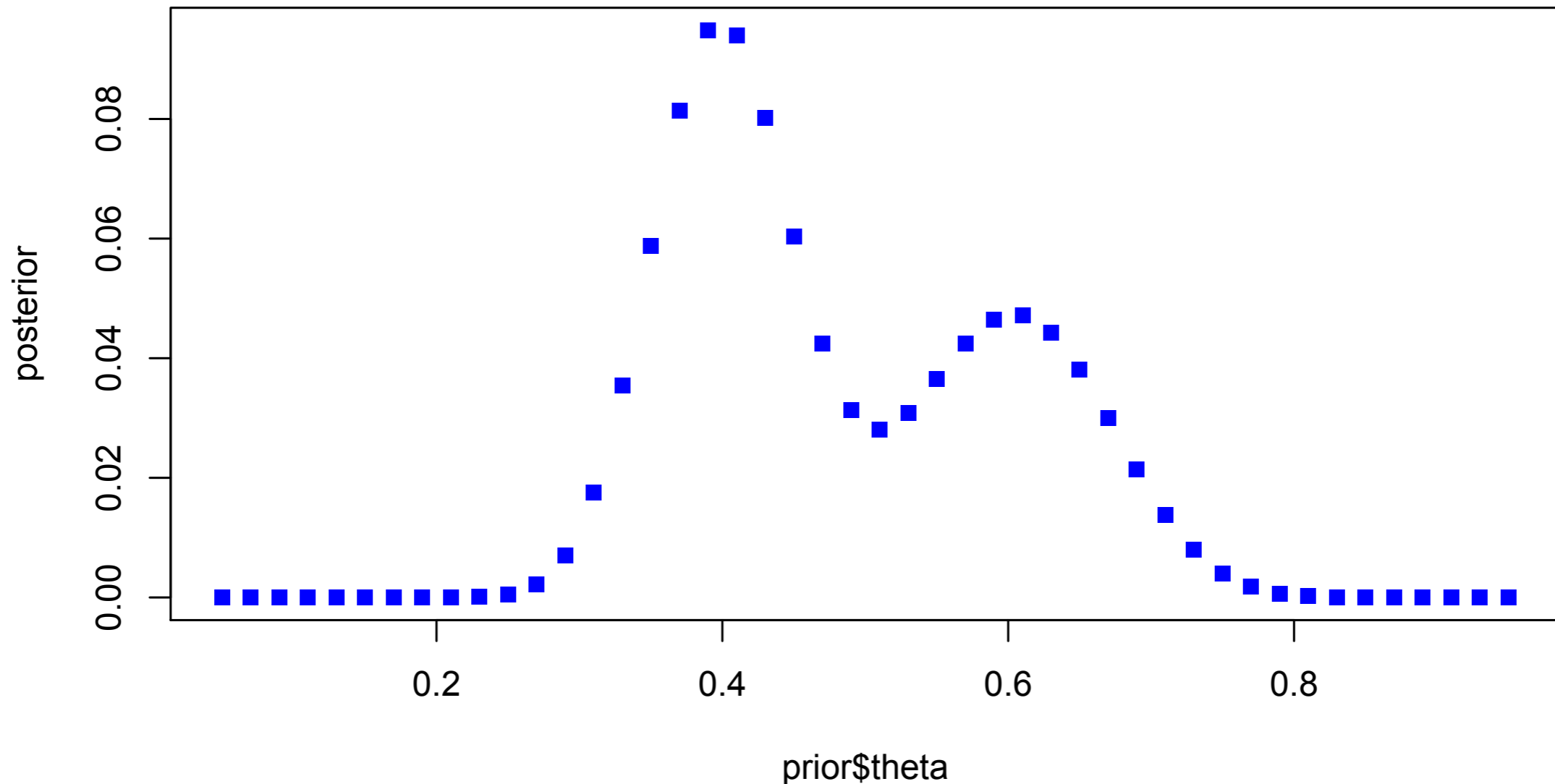
```
> plot(prior$theta, dbinom(20, 40, prior$theta), type="p", col="blue", pch=15)
```



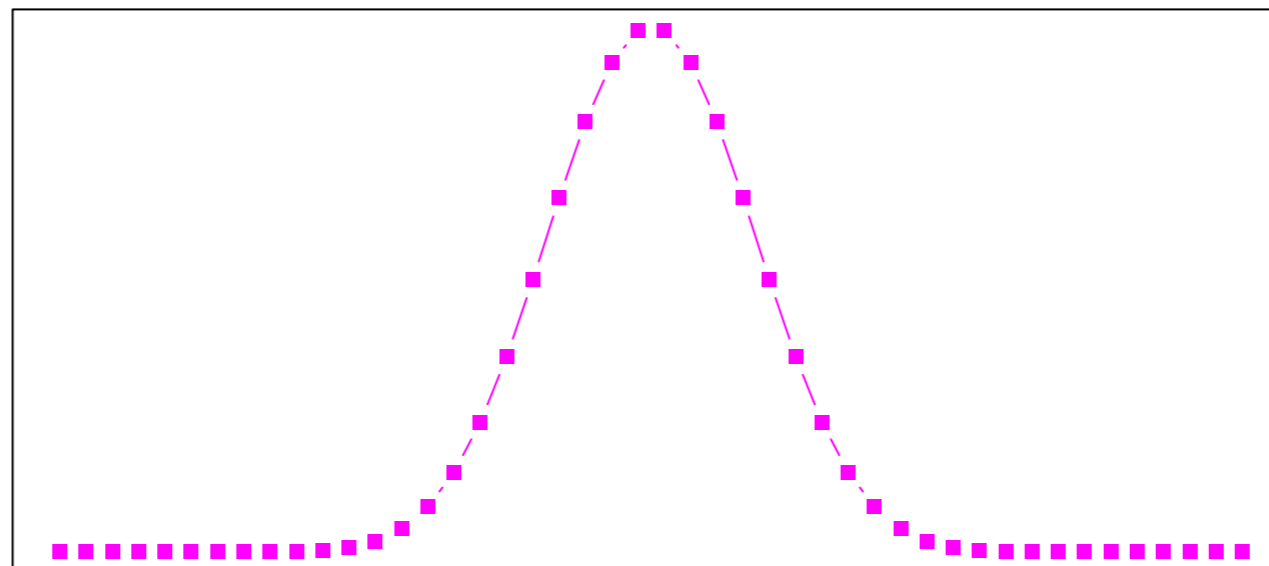
the posterior

$$p(\theta_i|D) = \frac{p(D|\theta_i)p(\theta_i)}{\sum_{\theta} p(D|\theta)p(\theta)}$$

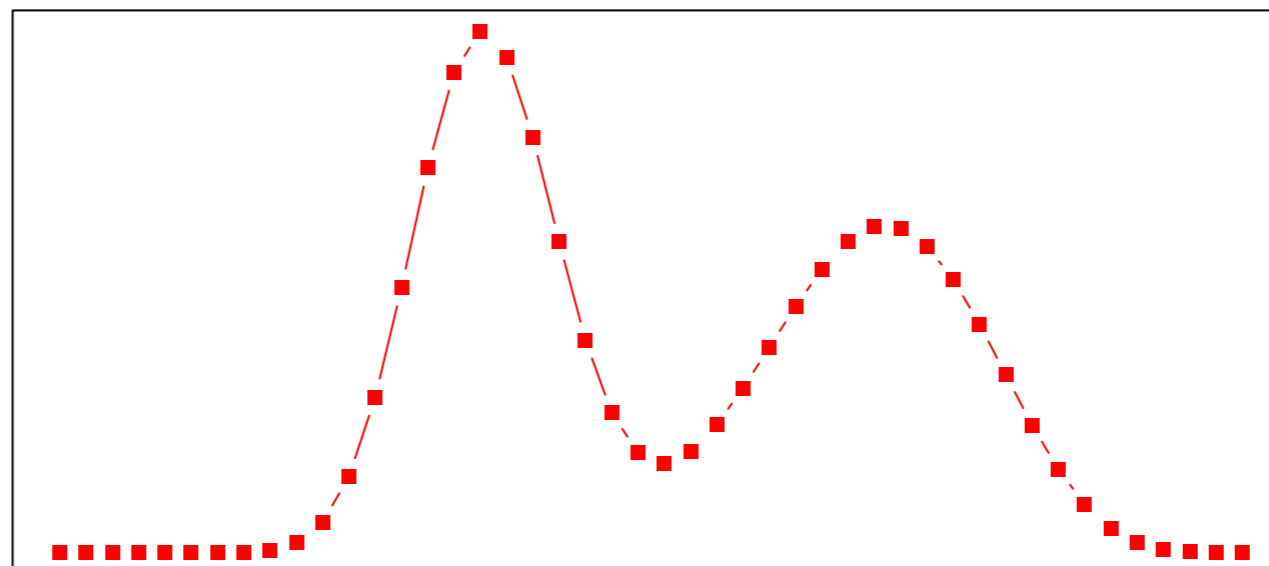
```
> post_num <- dbinom(20, 40, prior$theta) * prior$prob  
> posterior <- post_num / sum(post_num)  
> plot(prior$theta, posterior, type="p", col="blue", pch=15)
```



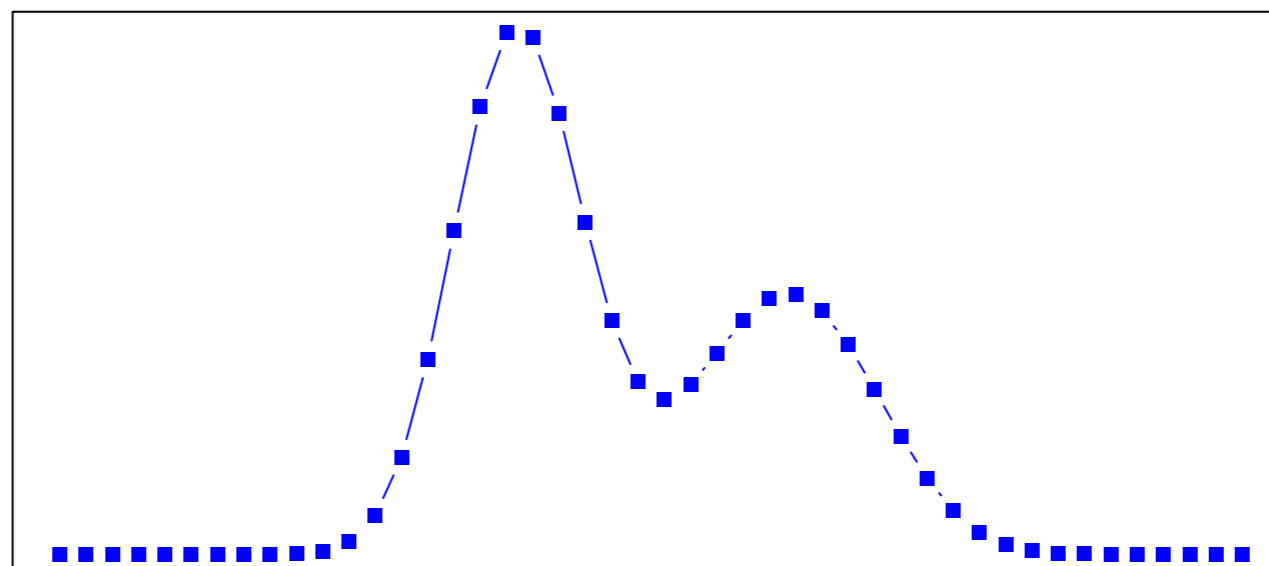
likelihood



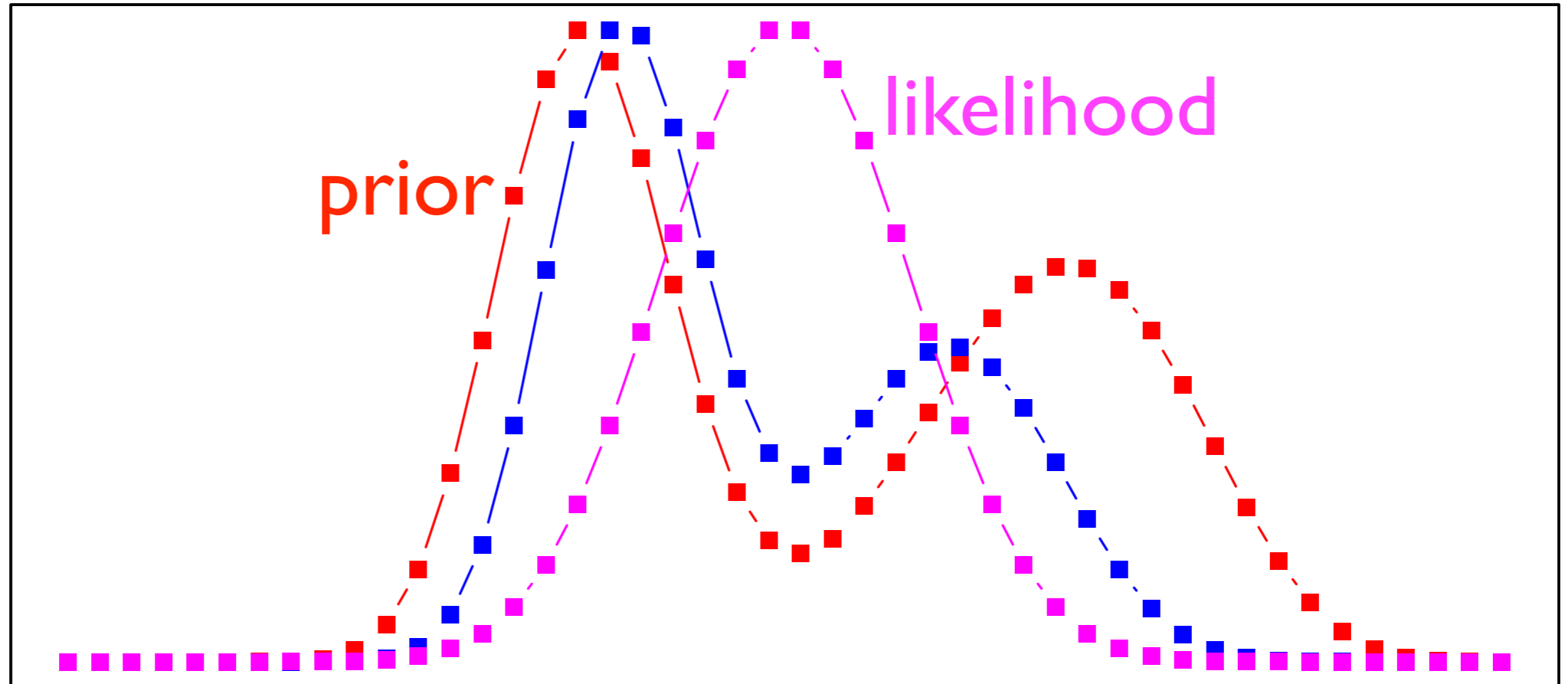
prior



posterior



posterior



Grid Approximation

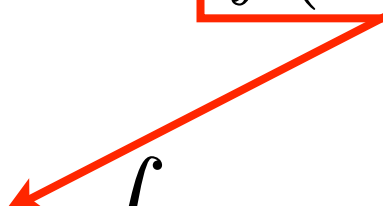
- we can use a discrete prior
- we could (although we didn't here) also use a discrete likelihood
- then just apply Bayes' Theorem to compute the posterior

Grid Approximation

- the finer / coarser the grid, the better / worse the approximation
- values in between the grid are not given

Markov Chain Monte Carlo (MCMC)

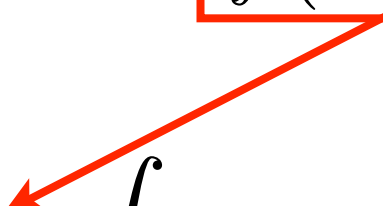
$$f(\theta|data) = \frac{f(data|\theta)f(\theta)}{f(data)}$$

$$f(data) = \int f(data|\theta)f(\theta)d\theta$$


- marginal probability involves an integral
- calculus ninjas can do the algebra to come up with solutions for conjugate priors
- for many distributions (esp multivariate) integrals may not be easy (or even possible) to compute

Markov Chain Monte Carlo (MCMC)

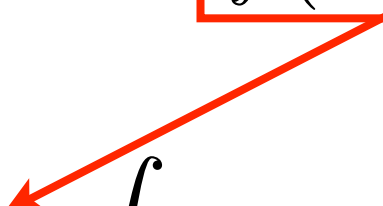
$$f(\theta|data) = \frac{f(data|\theta)f(\theta)}{f(data)}$$

$$f(data) = \int f(data|\theta)f(\theta)d\theta$$


- e.g. if we had a beta prior distribution on the variance of a normal distribution likelihood, the posterior distribution for the variance would not have a known form (so say the calculus ninjas)
- what to do?

Markov Chain Monte Carlo (MCMC)

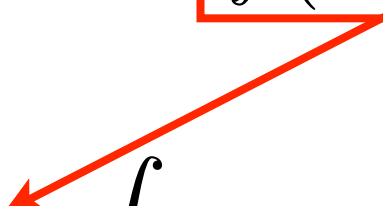
$$f(\theta|data) = \frac{f(data|\theta)f(\theta)}{f(data)}$$

$$f(data) = \int f(data|\theta)f(\theta)d\theta$$


- Sampling methods (like MCMC)
- generate (**simulate**) a **sample** of size n from the **posterior** distribution
- check Lynch CH4 for inversion sampling, rejection sampling

Markov Chain Monte Carlo (MCMC)

$$f(\theta|data) = \frac{f(data|\theta)f(\theta)}{f(data)}$$

$$f(data) = \int f(data|\theta)f(\theta)d\theta$$


- MCMC facilitates sampling from complex, multivariate distributions for which there may not be a closed form solution to the calculus/algebra
- MCMC simulates sampling from multivariate densities by breaking them down into more manageable univariate densities

MCMC

- **Markov Chain**
 - the process of sampling a new value from the posterior distribution, given the previous value
- **Monte Carlo**
 - refers to the random simulation process
 - like a **random walk**

MCMC

- Two kinds of random walks in MCMC
- **Gibbs sampling**
- **Metropolis-Hastings sampling**

Gibbs Sampling

- let's say we're trying to estimate a **multivariate posterior density** for the parameter vector $\Theta = (\theta_1, \theta_2, \dots, \theta_k)$

1. assign starting values S to $\Theta^{j=0} = S$

2. $j = j + 1$

3. sample $(\theta_1^j | \theta_2^{j-1}, \theta_3^{j-1}, \dots, \theta_k^{j-1})$

4. sample $(\theta_2^j | \theta_1^j, \theta_3^{j-1}, \dots, \theta_k^{j-1})$

5. ...

6. sample $(\theta_k^j | \theta_1^j, \theta_2^j, \dots, \theta_{k-1}^j)$

7. return to step 2

Gibbs Sampling

- Gibbs sampling involves ordering the parameters and sampling from the conditional distribution for each parameter, given the current value of all the other parameters, and then repeatedly cycling through this update process
- Each loop through the parameter vector is an “iteration” of the Gibbs sampler

Gibbs Sampling

- e.g. let's say $\Theta = (\theta_1, \theta_2) = (\mu, \sigma)$

1. choose starting values $\Theta^{j=0} = (1.0, 1.0)$

2. $j = j + 1$

3. Sample $(\mu^j | \sigma^{j-1}, data)$

4. Sample $(\sigma^j | \mu^j, data)$

5. return to step 2



Gibbs Sampling

- the trick then is to take a **multivariate density**, for which there may be no closed form solution to the integral (and for which random sampling from it is not feasible), and break it down into **univariate densities**, for which we can do either or both

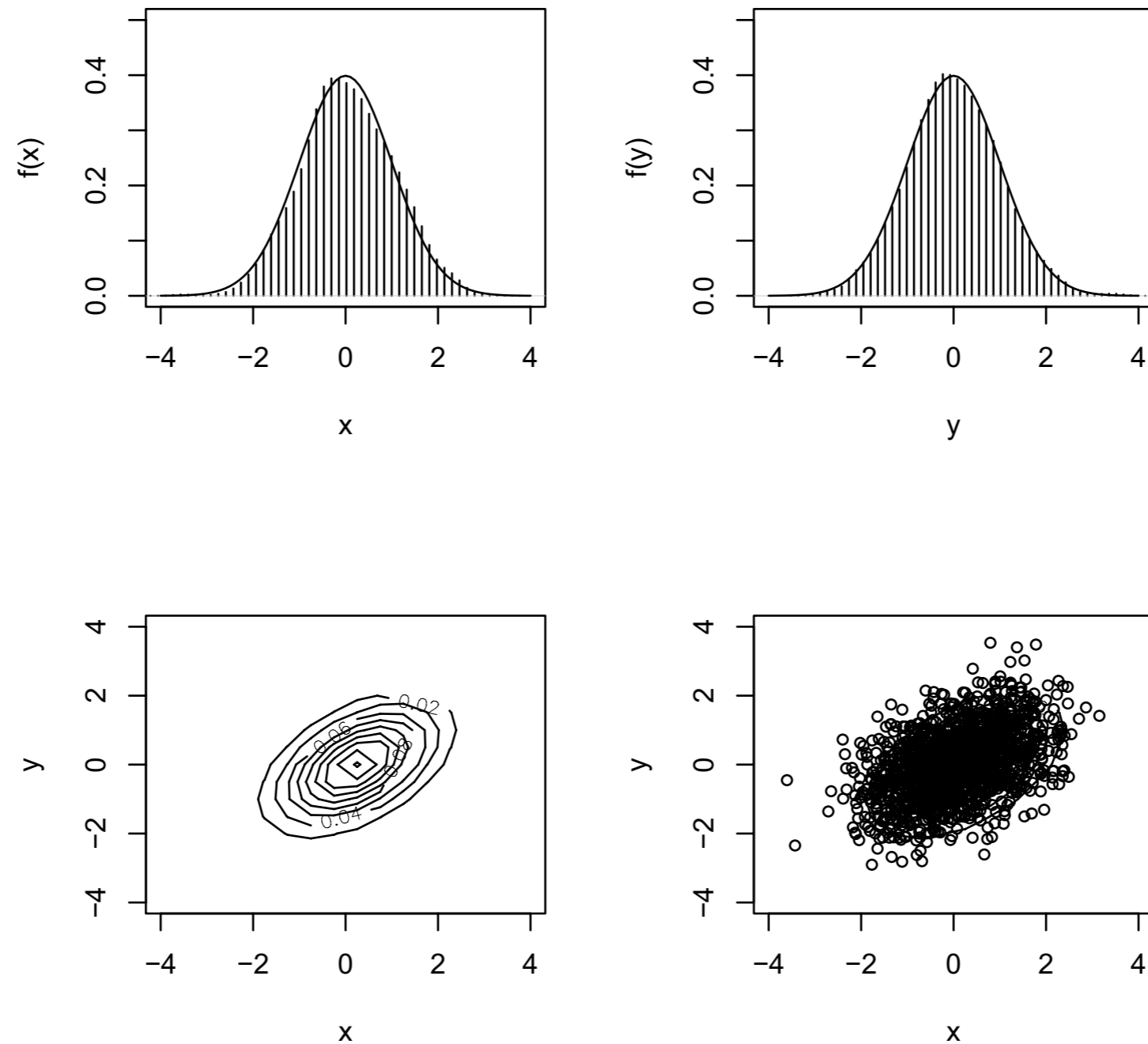


Fig. 4.10. Results of Gibbs sampler for standard bivariate normal distribution: Upper left and right graphs show marginal distributions for x and y (last 1,500 iterations); lower left graph shows contour plot of true density; and lower right graph shows contour plot of true density with Gibbs samples superimposed.

See Lynch Ch 4 for much more detail

Gibbs Sampling

- Gibbs sampling although simple has limitations
- conditional distributions may not be so easily derived or simulated
- Gibbs sampling can be slow / inefficient

MH Sampling

- Metropolis-Hastings (MH) sampling turns out to be more generally useful
- also a “random walk” but steps taken are more cleverly decided upon
- see Lynch Ch 5, Kruschke Ch 7, for details

Software for MCMC

- **JAGS**
 - + “rjags” package in R
 - <http://mcmc-jags.sourceforge.net/>
- **BUGS (& OpenBUGS)**
- <http://www.mrc-bsu.cam.ac.uk/bugs/>
 - “BRugs” package in R
- “mcmc” package in R
 - <http://www.stat.umn.edu/geyer/mcmc/library/mcmc/doc/demo.pdf>

R/JAGS

1. install JAGS for your computer

- <http://mcmc-jags.sourceforge.net/>

2. install the “rjags” package in R

- `install.packages(“rjags”)`
- `library(rjags)`

R/JAGS

- uses a special meta-language (BUGS) to specify your model, in terms of
 - stochastic elements
 - deterministic elements
 - your data

R/JAGS

- JAGS model declaration in a file (.bug)
- in R, prepare data, pass data and model syntax to JAGS via the `jags.model` command
- JAGS exploits conjugacy when it can, and will use sampling (Gibbs, MH, etc) when a conditional distribution is not recognizable in closed form etc

An Example

- from:
<http://www.johnmyleswhite.com/notebook/2010/08/20/using-jags-in-r-with-the-rjags-package/>

An Example

- let's create some fictitious data using a normal distribution

```
> N <- 100
```

```
> x <- rnorm(N, 0, 5)
```

An Example

- specify our model of the data in JAGS syntax
- put this into a plain text file and save as `example1.bug`

```
model {  
  for (i in 1:N) {  
    x[i] ~ dnorm(mu, tau)  
  }  
  mu ~ dnorm(0, .0001)  
  tau <- pow(sigma, -2)  
  sigma ~ dunif(0, 100)  
}
```

An Example

```
model {  
  for (i in 1:N) {  
    x[i] ~ dnorm(mu, tau)  
  }  
  mu ~ dnorm(0, .0001)  
  tau <- pow(sigma, -2)  
  sigma ~ dunif(0, 100)  
}
```

i indexes your data points

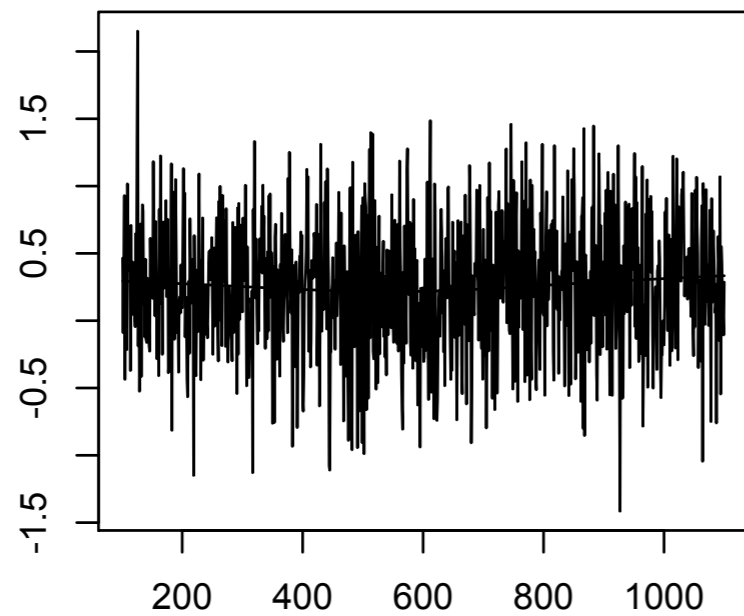
we're saying x is normally distributed with mean mu and precision tau (tau = 1/variance)

we define flat priors on mu and tau

mu is normally distributed with mean 0 and precision .0001

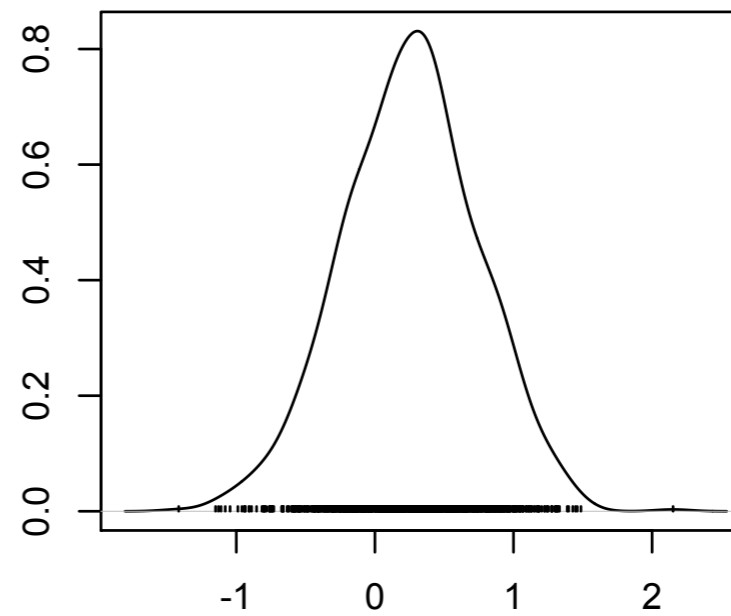
tau is $1/\text{sigma}^2$ where sigma is uniformly distributed between 0 and 100

Trace of mu



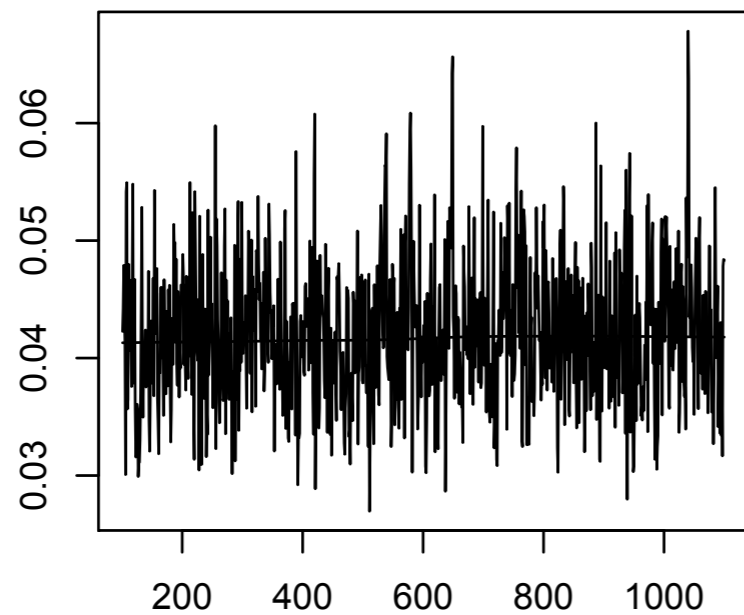
Iterations

Density of mu



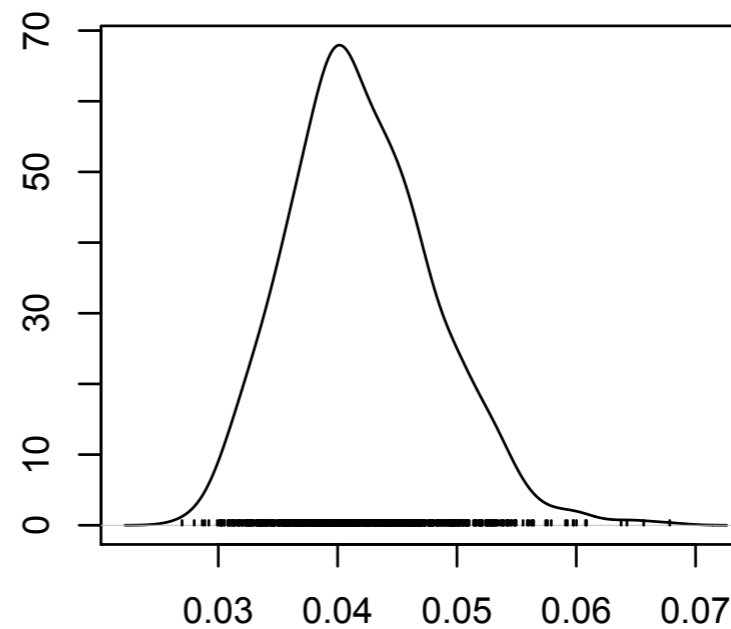
N = 1000 Bandwidth = 0.129

Trace of tau



Iterations

Density of tau

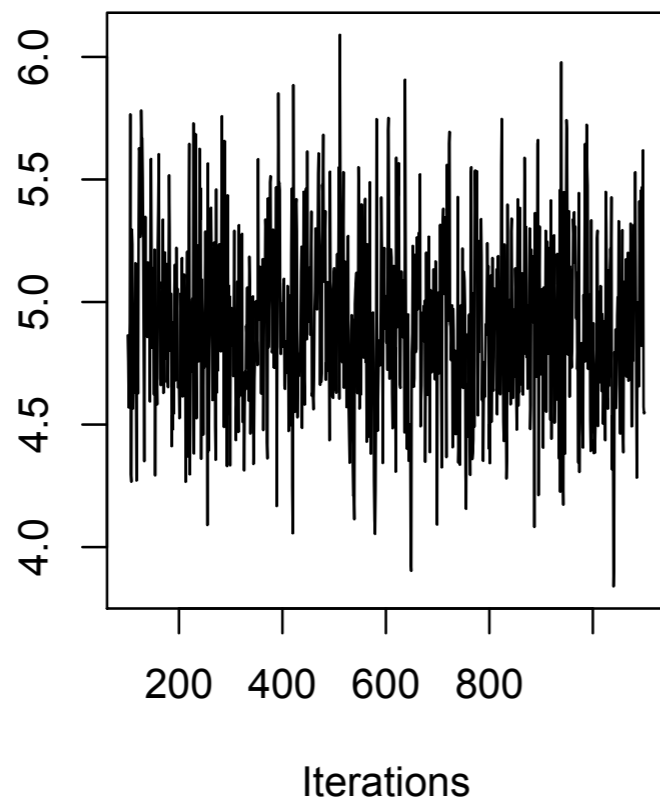


N = 1000 Bandwidth = 0.001592

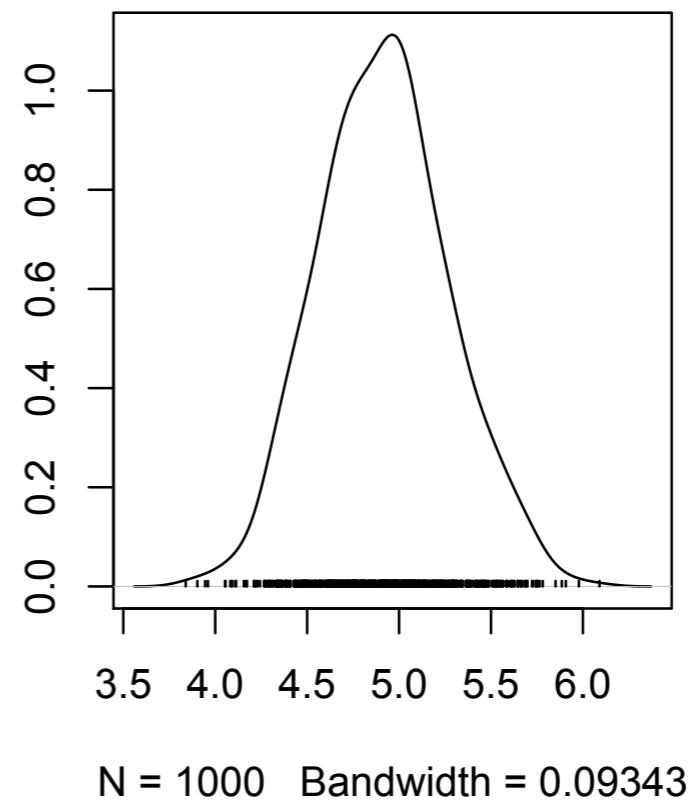
An Example

```
> sigma = sqrt(1/posterior[[1]][,2])  
> plot(sigma)
```

Trace of var1



Density of var1



models

```
model {  
  for (i in 1:N) {  
    x[i] ~ dnorm(mu, tau)  
  }  
  mu ~ dnorm(0, .0001)  
  tau <- pow(sigma, -2)  
  sigma ~ dunif(0, 100)  
}
```

- here we had a simple model with 2 parameters (mu, tau)
- we can specify **any model** we want
- nonlinear, multi-level, mixture, whatever
- MCMC will attempt to sample the posterior

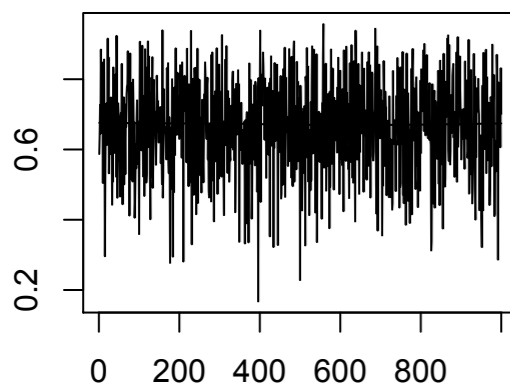
e.g. Illy vs Lavazza

```
> N <- 10  
> y <- c(1,1,1,0,1,1,0,0,1,1)  
> jags <- jags.model('coffee.bug',  
data=list('y'=y, 'N'=N),  
n.chains=1, n.adapt=100)  
> post <- coda.samples(jags, c('theta'),  
1000)  
> plot(post)
```

coffee.bug

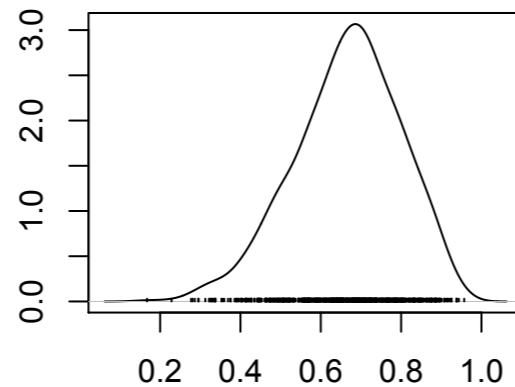
```
model {  
  for (i in 1:N) {  
    y[i] ~ dbern(theta)  
  }  
  # prior  
  theta ~ dbeta(priorA, priorB)  
  priorA <- 1  
  priorB <- 1  
}
```

Trace of theta



Iterations

Density of theta



N = 1000 Bandwidth = 0.0352

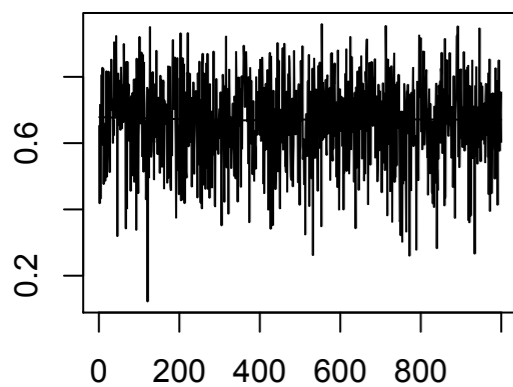
e.g. Illy vs Lavazza

```
> N <- 10
> y <- c(1,1,1,0,1,1,0,0,1,1)
> jags <- jags.model('coffee2.bug',
data=list('y'=y, 'N'=N),
n.chains=1, n.adapt=100)
> post <- coda.samples(jags, c('theta'),
1000)
> plot(post)
```

coffee2.bug

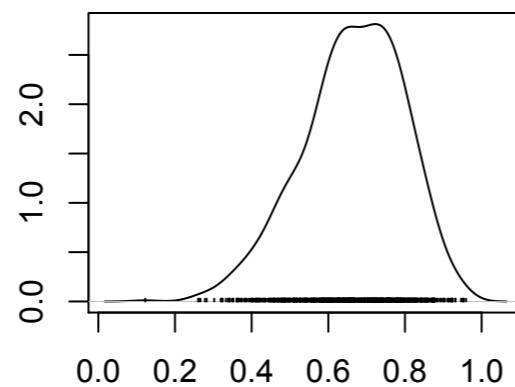
```
model {
  for (i in 1:N) {
    y[i] ~ dbern(theta)
  }
  # prior
  theta ~ dunif(0,1)
}
```

Trace of theta



Iterations

Density of theta



N = 1000 Bandwidth = 0.03529

MCMC

- you can specify any crazy model you want
- multi-level models
- hierarchical models
- nonlinear models
- MCMC will attempt to sample the posterior

MCMC

- diagnostics
- burn-in period
- stability
- chains
- lots of details, check the texts if you are interested in pursuing this approach