BMEG 802 – Advanced Biomedical Experimental Design and Analysis

Markov Chain Monte Carlo (MCMC)

Joshua G. A. Cashaback, PhD

Recap

- Bayesian
  - Any Distribution
  - Any Model (Linear, nonlinear)
  - Rich characterization of posterior distribution over all possible model parameter values
- Frequentist
  - normal distribution
  - linear models
  - point estimates of parameter values

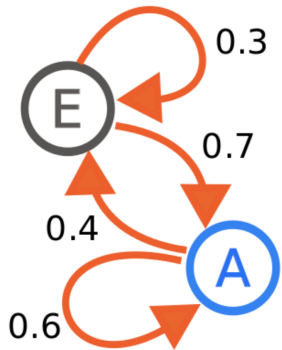
Today

Markov Chain Monte Carlo (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

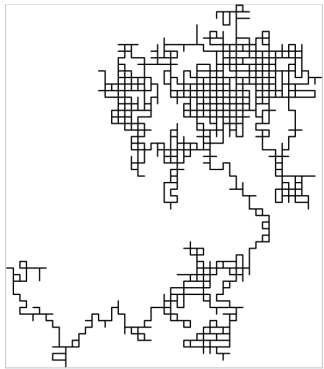
Lynch textbook
https://link.springer.com/content/pdf/10.1007%2F978-0-387-71265-9.pdf

Markov Chain



Transition Graph of a two-state Markov Chain. Values denote the probability of staying or jumping between states.

Monte Carlo



Monte Carlo: repeated random sampling. In the example above, randomly selecting N,E,S,W direction changes leads to “random walks”.

Why MCMC?

$$p(\theta|y) = \frac{p(y|\theta) \cdot p(\theta)}{\int_a^b p(y|\theta) \cdot p(\theta) d\theta}$$

- Marginal Integral often does not have an analytic solution
- MCMC facilitates sampling from complex, multivariate distributions
  - for which there may not be a closed form solution to the calculus/algebra to find conjugate priors
- MCMC simulates sampling from multivariate densities by breaking them down into more manageable univariate densities

Sampling

Two common kinds of random walks in MCMC (There are others)

1. Gibbs sampling
2. Metropolis-Hastings sampling

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

Let's say we're trying to estimate a multivariate posterior density

$$\Theta = (\theta_1, \theta_2, ..., \theta_k)$$

1. Assign starting values to $\Theta^{j=0}$
2. $j = j + 1$
3. sample $(\theta_1^j | theta_2^{j-1}, theta_3^{j-1}, ..., theta_k^{j-1})$
4. sample $(\theta_2^j | theta_1^{j-1}, theta_3^{j-1}, ..., theta_k^{j-1})$
5. ...
6. sample $(\theta_k^j | theta_1^{j-1}, theta_2^{j-1}, ..., theta_{k-1}^{j-1})$
7. return to step 2

Gibbs Sampling - Binomial Distribution

Lets come back to our coffee example, where I make the claim I can differentiate between Little Goat and Starbucks. Remember, I correctly identified 16 out of 20 samples.

Prior: $Beta(\alpha, \beta)$

Likelihood: $Binomial(w|n, k)$

Posterior: $Beta(\alpha_h, \beta_h)$, s.t.,

$\alpha_h = k + \alpha$

$\beta_h = n - k + \beta$

Gibbs Sampling - Binomial Distribution

```
K = 16
N = 20
n = 10000 # number of samples to perform
alpha = 1
beta = 1
WEIGHT = array(NA,n)
WEIGHT[1] = 0.5
for (i in 2:n) {
  WEIGHT[i] = rbeta(1, (K + alpha), (N - K + beta), ncp = 0)
}
```

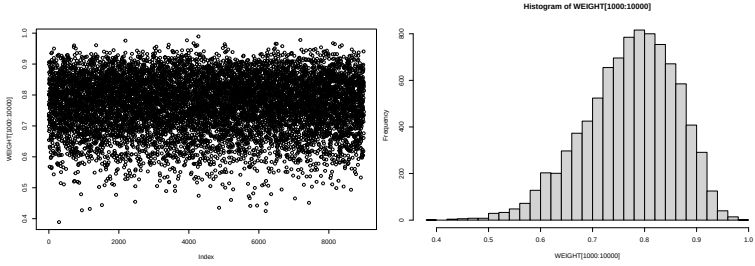
Gibbs Sampling - Binomial Distribution

Plotting the Posterior Probability Distribution

```
plot(WEIGHT[1000:10000])
freq <- hist(WEIGHT[1000:10000],40)$counts
midpoint = hist(WEIGHT[1000:10000],40)$mids
mode = midpoint[which.max(freq)]
mode
```

```
## [1] 0.79
```

```
# allowing for a 1000 sample "burn-in" period
```



Here we find the mode to find the Maximum a Posteriori (MAP) estimate. Median = least absolute error, Mean = least squares error. True solution = 0.8

Gibbs Sampling - Normal Distribution

We have the following data: x = [105.01510, 97.05776 106.70943, 98.73814, 95.81697, 92.36068, 76.02600, 112.06383, 109.26824 108.18182]

Let's pretend this data was drawn from a normal distribution, $\mathcal{N}(\mu, \sigma^2)$.

We want to estimate μ and σ using MCMC.

Gibbs Sampling - Normal Distribution

Assuming an uninformative prior, the conditional probability distributions are:

$$p(\sigma^2|\mu, Y) = \frac{1}{\Gamma\left(\frac{n}{2}, \frac{1}{2}\frac{\Sigma_{i=1}^{n}(Y_i-\mu)}{2}\right)}$$

$$p(\mu|\sigma^2, Y) = \mathcal{N}\left(\bar{Y}, \frac{\sigma^2}{n}\right)$$

Γ is the Gamma function [rgamma() in R].

Derivations for conditional probability distributions in Lynch (Chapter 3).

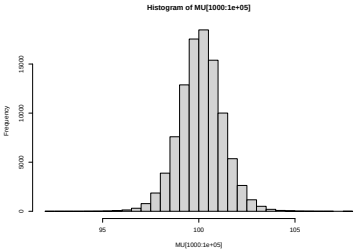
Gibbs Sampling - Normal Distribution

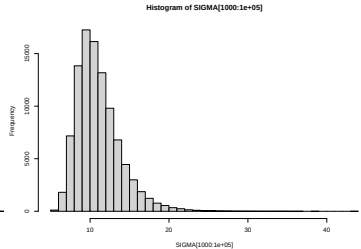
```r
x <- c(105.01510, 97.05776, 106.70943, 98.73814, 95.81697,
       92.36068, 76.02600, 112.06383, 109.26824, 108.18182)
n = 100000 # number of samples to perform
MU = array(NA,n) # initialize arrays
SIGMA = array(NA,n)
MU[1] = 90 # initial guess
SIGMA[1] = 5
for (i in 2:n) {
  # finding sigma (NOT variance as in the equations)
  SIGMA[i] = (1/rgamma(1,length(x)/2,sum((x-MU[i-1])^2)/2))^(1/2)
  MU[i] = rnorm(1,mean(x),SIGMA[i]/length(x))
}
```

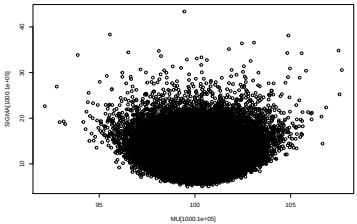
Gibbs Sampling - Normal Distribution

Marginal and Joint Posterior Probabilities

```
hist(MU[1000:100000],40) # allowing for a 1000 sample "burn-in" period
hist(SIGMA[1000:100000],40)
plot(MU[1000:100000],SIGMA[1000:100000])
```

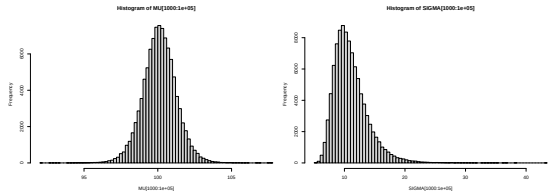






Gibbs Sampling - Normal Distribution

```
MUfreq <- hist(MU[1000:100000],101)$counts
MUmidpoint = hist(MU[1000:100000],101)$mids
MUmode = MUmidpoint[which.max(MUfreq)]
SIGfreq <- hist(SIGMA[1000:100000],101)$counts
SIGmidpoint = hist(SIGMA[1000:100000],101)$mids
SIGmode = SIGmidpoint[which.max(SIGfreq)]
```



Again, we can characterize (mean, median, mode, confidence intervals, etc.) the posterior distibution of the parameters given the data.

Gibbs Sampling - Normal Distribution

```
MUmode
```

```
## [1] 100.1
```

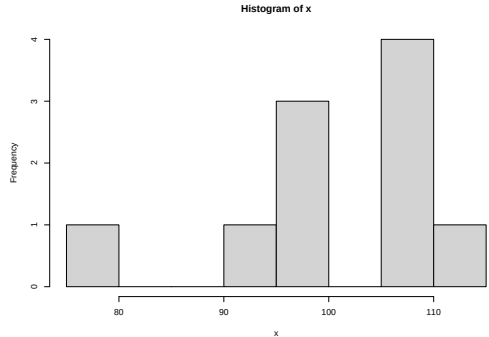
```
SIGmode
```

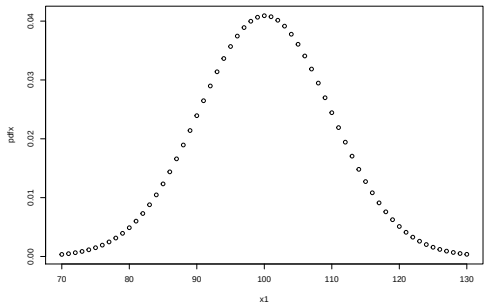
```
## [1] 9.75
```

Here we used the mode (MAP estimate). Data was sampled from $\mathcal{N}(\mu = 100, \sigma = 10)$

Gibbs Sampling - Plots

```
x1 = seq(70.0, 130, 1.0)
pdfx = (1/(SIGmode * sqrt(2 * pi)))*exp(-(1/2) * ((x1 - MUmode)/SIGmode)^2)
hist(x,10)
plot(x1,pdfx)
```





Gibbs Sampling - Pros and Cons

Pros

- Gibbs sampling is simple
- Because you are working with the conditional distribution, you always accept the new sample:
  - http://gregorygundersen.com/blog/2020/02/23/gibbs-sampling/

Cons

- Conditional distributions may not be so easily derived or simulated
- Gibbs sampling can be slow / inefficient

Gibbs is a special case of Metropolis-Hastings

Metropolis-Hastings

- Metropolis-Hastings sampling turns out to be more generally useful.
- also a “random walk” but steps taken are more cleverly decided upon

Metropolis-Hastings

Existence of stationary distribution: there must exist a stationary distribution $\pi(x)$. A sufficient but not necessary condition is detailed balance, which requires that each transition $x \to x'$ is reversible: for every pair of states x, x', the probability of being in state x and transitioning to state x' must be equal to the probability of being in state x' and transitioning to state x, $\pi(x)P(x' \mid x) = \pi(x')P(x \mid x')$.

Uniqueness of stationary distribution: the stationary distribution $\pi(x)$ must be unique. This is guaranteed by ergodicity of the Markov process, which requires that every state must (1) be aperiodic—the system does not return to the same state at fixed intervals; and (2) be positive recurrent—the expected number of steps for returning to the same state is finite.

Metropolis-Hastings

$$P(x' \mid x)P(x) = P(x \mid x')P(x'),$$

which is re-written as

$$\frac{P(x' \mid x)}{P(x \mid x')} = \frac{P(x')}{P(x)}.$$

Metropolis-Hastings

The approach is to separate the transition in two sub-steps; the proposal and the acceptance-rejection. The proposal distribution $g(x' \mid x)$ is the conditional probability of proposing a state x' given x, and the acceptance distribution $A(x', x)$ is the probability to accept the proposed state x'. The transition probability can be written as the product of them:

$$P(x' \mid x) = g(x' \mid x)A(x', x).$$

Inserting this relation in the previous equation, we have

$$\frac{A(x', x)}{A(x, x')} = \frac{P(x')}{P(x)} \frac{g(x \mid x')}{g(x' \mid x)}.$$

Metropolis-Hastings

The next step in the derivation is to choose an acceptance ratio that fulfills the condition above. One common choice is the Metropolis choice: $A(x', x) = \min\left(1, \frac{P(x')}{P(x)} \frac{g(x|x')}{g(x'|x)}\right)$.

For this Metropolis acceptance ratio A, either $A(x', x) = 1$ or $A(x, x') = 1$ and, either way, the condition is satisfied.

Metropolis-Hastings

The Metropolis–Hastings algorithm thus consists in the following:

1. Initialize

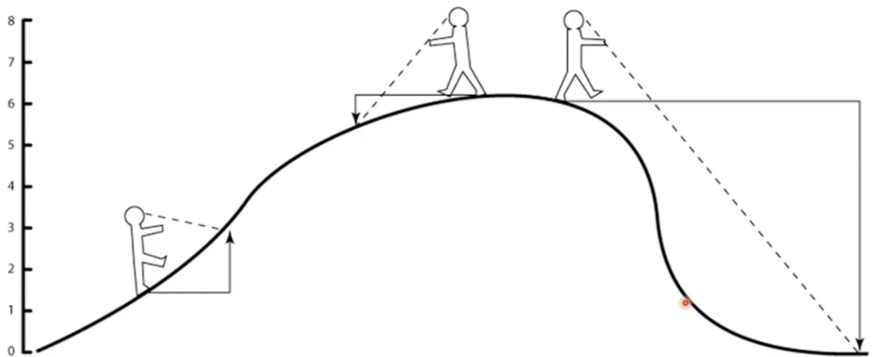
   - Pick an initial state x_0.
   - Set $t = 0$.

2. Iterate

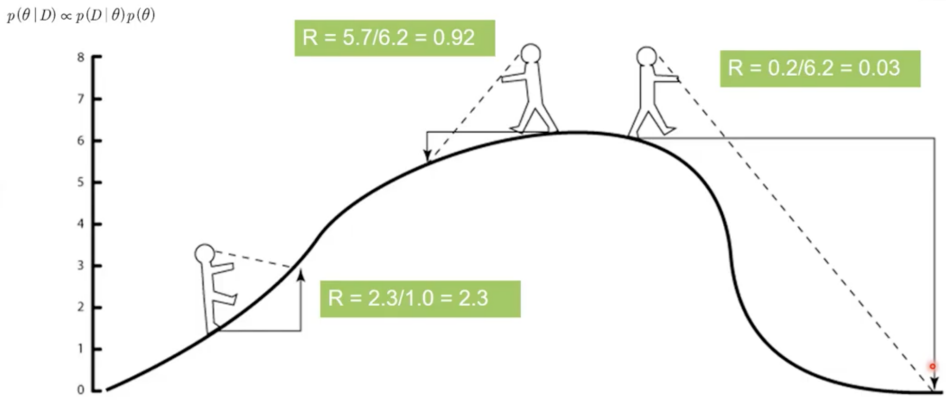
   - Generate a random candidate state x' according to $g(x' \mid x_t)$.
   - Calculate the acceptance probability $A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t|x')}{g(x'|x_t)}\right)$.
   - Accept or reject:
     - generate a uniform random number $u \in [0, 1]$;
     - if $u \leq A(x', x_t)$, then ''accept" the new state and set $x_{t+1} = x'$;
     - if $u > A(x', x_t)$, then ''reject" the new state, and copy the old state forward $x_{t+1} = x_t$.
   - Increment: set $t = t + 1$.

Metropolis-Hastings

$$p(\theta \mid D) \propto p(D \mid \theta) p(\theta)$$

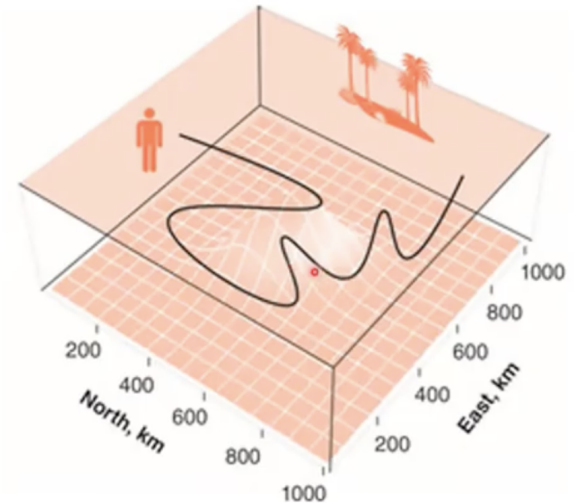


Metropolis-Hastings



Note: all R over 1 are accepted

Metropolis-Hastings



Metropolis-Hastings - Example

Lets use our Little Goat and Starbucks example again.

Metropolis-Hastings - Example

Prior: Beta($\alpha = 1, \beta = 1$)

Likelihood: Binomial($n = 20, k = 16$);

Posterior: $p(w|n, k, \alpha, \beta) = \frac{1}{B(\alpha_h, \beta_h)} w^{\alpha_h - 1}(1 - w)^{\beta_h - 1}$, s.t.

$$\alpha_h = k + \alpha$$

$$\beta_h = n - k + \beta$$

Thus, $p(w|k, n, \alpha, \beta) \propto w^{k+\alpha-1}(1 - w)^{n-k+\beta-1}$

Metropolis-Hastings - Example

Our posterior is

$$p(w|k,n,\alpha,\beta) \propto w^{k+\alpha-1}(1-w)^{n-k+\beta-1}$$

To construct a random-walk Metropolis algorithm, we choose the proposal

$$w' = g(w|w_t) \sim \mathcal{N}(w_t, 0.4^2)$$

and accept, i.e. $w^{(t+1)} = w'$ with probability min{1, A(w', w_t)} where

$$A(w', w_t) = \frac{P(w'|k,n,\alpha,\beta)}{P(w_t|k,n,\alpha,\beta)} = \frac{(w')^{k+\alpha-1}(1-w')^{n-k+\beta-1}}{(w_t)^{k+\alpha-1}(1-w_t)^{n-k+\beta-1}}$$

otherwise, $w^{(t+1)} = w^t$

Note, that all w within [0,1]

Metropolis-Hastings - Example

It is often convenient to deal with the logs of probability distributions.

Let's take the natural log of our ratio:

$$ln[A(w', w_t)] = ln\left[\frac{(w')^{k+\alpha-1}(1-w')^{n-k+\beta-1}}{(w_t)^{k+\alpha-1}(1-w_t)^{n-k+\beta-1}}\right]$$

which leads to

$$ln[A(w', w_t)] = [(k+\alpha-1)\cdot ln(w')+(n-k+\beta-1)\cdot ln(1-w')]-[(k+\alpha-1)\cdot ln(w_t)+(n-k+\beta-1)\cdot ln(1-w_t)]$$

Doing the same for the acceptance rule:

Accept if $ln(u) \leq ln[A(w', w_t)]$

Metropolis-Hastings - Example

```
n = 100000
log_q = function(w, k=16, n=20) {
  if (w<0 | w>1) return(-Inf)
  alpha = 1
  beta = 1
  (k+alpha-1)*log(w)+(n-k+beta-1)*log(1-w)
}
current = 0.5 # Initial guess of w
samps = rep(NA,n)
for (i in 1:n) {
  proposed = rnorm(1, current, 0.4) # tuning parameter is 0.4
  logr = log_q(proposed)-log_q(current)
  if (log(runif(1)) <= logr) current = proposed
  samps[i] = current
}
acc_rate <- length(unique(samps))/n # acceptance rate
```

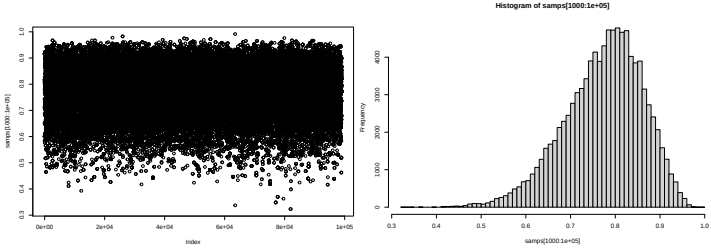
Metropolis-Hastings

Posterior Probability

```
plot(samps[1000:100000])
freq <- hist(samps[1000:100000],50)$counts
midpoint = hist(samps[1000:100000],50)$mids
mode = midpoint[which.max(freq)]
mode
```

```
## [1] 0.805
```

```
# allowing for a 1000 sample "burn-in" period
```



JAGS

Built in R package that will carry out MCMC for you.

JAGS

Install JAGS version 4.3.0 (JAGS-4.3.0.dmg) from Martyn Plummer's:
https://sourceforge.net/projects/mcmc-jags/files/JAGS/

1.Download the disk image from the JAGS website.

2.Double click on the disk image to mount (this may not be required).

3.Double click on the 'JAGS-4.3.0.mpkg' file within the mounted disk image.

4.Follow the instructions in the installer. If you receive a warning that this software cannot be installed because it comes from an unidentified developer, you need to go to System Preferences > Security & Privacy, and authorize the installation there before proceeding.

5.Authenticate as the administrative user. The first user you create when setting up Mac OS X has administrator privileges by default.

Note: I haven't used a Windows OS since 2007 so please google for installation instructions

JAGS

install.packages(c("R2jags", "coda", "R2WinBUGS", "lattice", "rjags"))

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

JAGS - Binomial Example

```
{
sink("binomial_model.R") # This is the file name for the jags code
cat("
model{
  # likelihood
  for (i in 1:N) {
    y[i] ~ dbern(theta)
  }
  # priors
  theta ~ dbeta(priorA, priorB)
  priorA <- 1
  priorB <- 1
}
",fill = TRUE)
sink()
}
```

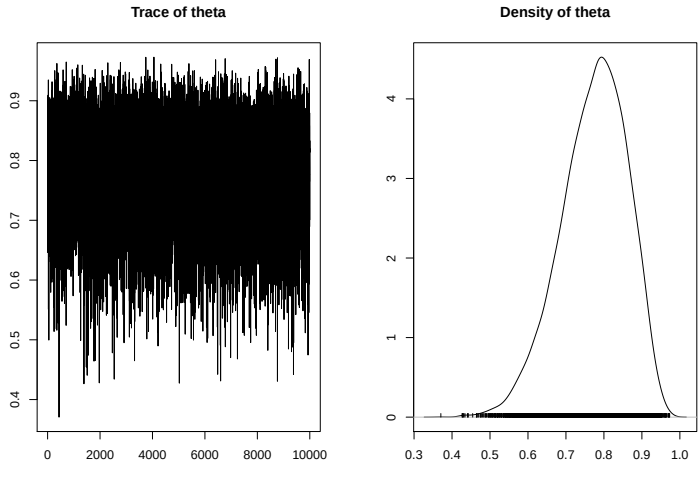
JAGS - Binomial Example

```
N <- 20
y <- c(1,1,1,0,1,1,0,1,1,1,0,1,1,1,1,1,1,1,0,1)
jags <- jags.model('binomial_model.R', data=list('y'=y, 'N'=N),
                   n.chains=1, n.adapt=100)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 20
##    Unobserved stochastic nodes: 1
##    Total graph size: 23
##
## Initializing model
```

JAGS - Binomial Example

```
plot(post)
```



JAGS - Further Examples

Some nice examples here:

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

Warm Starts

- Grid approximation as initial guess
- LSE as initial guess

MCMC

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

MCMC

- burn-in periods
- diagnostics (convergence)
- stability
- chains
- see Lynch for more details

Next Class

Bootstrapping