Lecture 4:

Markov Chain Monte Carlo

Overview

- Markov Chain Monte Carlo
  - Markov Chains
  - Metropolis-Hastings Algorithm
  - Gibbs sampling

- Advanced MCMC
  - Adaptive MCMC
  - Variable dimension techniques

Introduction

- Markov Chain Monte Carlo

- A combination between Monte Carlo sampling and time homogeneous Markov chains.

- The Monte Carlo approaches up to now have typically generated identically independently distributed (i.i.d) variables directly from the density of interest or indirectly in the case of importance sampling.

- The Markov property of a Markov chain introduces dependency, but only with a “one-step memory” between time points.

Markov Chains

Markov chain: a stochastic process in which future states are independent of past states given the present state.

Stochastic process: a *consecutive* set of *random countable* quantities defined on some known state space S.

- think of S as the parameter space.
- consecutive means a time component, indexed by *t*.

Consider one sample of $X^{(t)} = x^{(t)}$ to be a state at iteration *t*. The next draw $X^{(t+1)} = x$ is only dependent on the current sample, and not on any past draws.

- This satisfies the **Markov property:**

$$P\left(X^{(t+1)} = x \middle| X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \ldots, X^{(t)} = x^{(t)}\right) = P\left(X^{(t+1)} = x \middle| X^{(t)} = x^{(t)}\right)$$

Markov Chains

- So, the chain moves around in the parameter space, remembering only where it has been in the last state.
- What are the rules that determine how the chain jumps from one state to another at each period?
- The jumping rules are governed by a *transition probability matrix* (*kernel*), which is a mechanism that describes the probability of moving to some other state based on the current state.

Markov Chains

Transition kernel: For a discrete state space (s possible states): $\mathbf{P}$ is a $s \times s$ matrix of transition probabilities p_{ij}.

Example of San Francisco rainfall data from 1814 consecutive days:

	Wet today	Dry today
Wet yesterday	418	256
Dry yesterday	256	884

S has two states, "wet" and "dry". The rows of $\mathbf{P}$ sum to one and define a conditional (probability mass function) PMF, conditional on the current state.

$$\hat{\mathbf{P}} = \begin{bmatrix} \alpha & 1-\alpha \\ \beta & 1-\beta \end{bmatrix} = \begin{bmatrix} 0.620 & 0.380 \\ 0.224 & 0.775 \end{bmatrix}$$

Markov Chains

Using initial probabilities: if the probabilities of the initial states $\boldsymbol{\pi}^{(0)}$ are known, they can be used to infer future $\boldsymbol{\pi}^{(0)}\mathbf{P} = \boldsymbol{\pi}^{(1)}$. Moreover, this holds in general for all states and which can be inferred by simple matrix multiplication $\boldsymbol{\pi}^{(t)}\mathbf{P} = \boldsymbol{\pi}^{(t+1)}$.

Consider the rainfall data again and assume that the initial state $\boldsymbol{\pi}_{\mathbf{0}}^{T} = (0.5,0.5)$, what is the probability of wet and dry in $t = 1$:

$$\boldsymbol{\pi}^{(0)}\mathbf{P} = \boldsymbol{\pi}^{(1)} = (0.5,0.5)\begin{pmatrix} 0.620 & 0.380 \\ 0.224 & 0.775 \end{pmatrix} = (0.422,0.578)$$

and in $t = 2$:

$$\boldsymbol{\pi}^{(2)} = (0.422,0.578)\begin{pmatrix} 0.620 & 0.380 \\ 0.224 & 0.775 \end{pmatrix} = (0.391,0.608)$$

Markov Chains

Stationary distribution: a probability distribution $\boldsymbol{\pi}$ that satisfy $\boldsymbol{\pi}^{(t)}\mathbf{P} = \boldsymbol{\pi}^{(t)}$ is said to be *stationary* because the transition matrix doesn't change the probabilities of the process. If such a distribution exists, it is unique, and plays an essential role in the long term behavior of the process.

To obtain the stationary distribution it is necessry to perform calculations based on high powers. To compute these its is useful to work on the eigen-decomposition of the transition matrix

$$\mathbf{P} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1}$$

where $\mathbf{Q}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ the diagonal matrix with eigenvalues. It can be shown that

$$\mathbf{P}^{t} = \mathbf{Q}\boldsymbol{\Lambda}^{t}\mathbf{Q}^{-1}$$

which is computationally convenient because it is only needed to take the powers of the eigenvalues.

Markov Chains

Stationary distribution: Consider the following eigen-decomposition of the rainfall transition matix $\mathbf{P}$ and calculate the probabilities for $t = 10$

$$\mathbf{P^{10}} = \mathbf{Q\Lambda^{10}Q^{-1}} = \begin{matrix} 0.3694 & 0.6254 \\ 0.3687 & 0.6245 \end{matrix}$$

Multiply with $\mathbf{P}$ to check if it has become stationary at $t = 10$

$$\mathbf{P^{10}P} = \begin{matrix} 0.3691 & 0.6250 \\ 0.3684 & 0.6240 \end{matrix}$$

The marginal probabilities will then be (with some rounding errors)

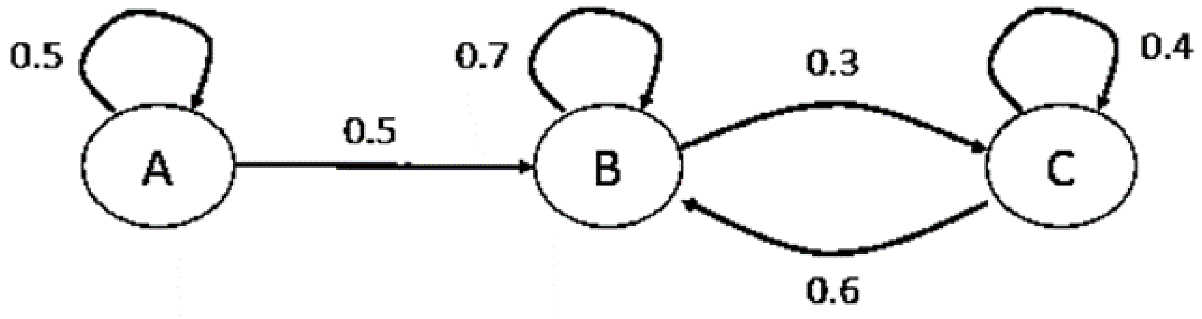
$$\boldsymbol{\pi} = \{\pi_1 = 0.37,\ \pi_2 = 0.63\}$$

and reversibility can be checked

$$\pi_1 p_{12} = \pi_2 p_{21} = 0.14$$

Markov Chains

Reducibility: A Markov chain is *irreducible* if it is possible go from any state to any other state (not necessarily in one step).

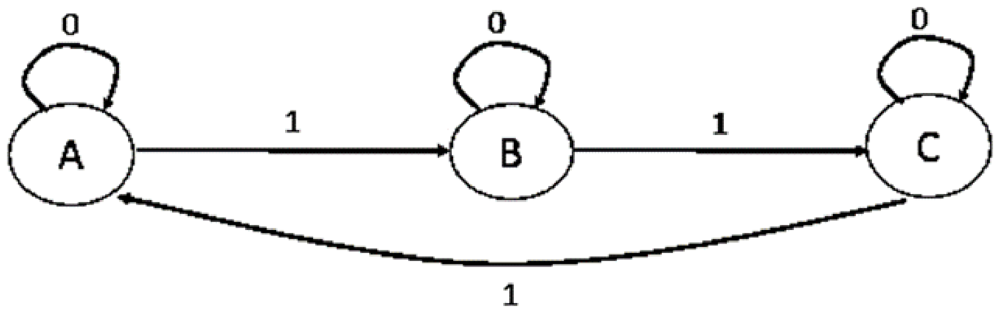


The chain is reducible because we cannot get (the probability of transitioning is zero) to A from B or C regardless of the number of steps we take.

Markov Chains

Periodicity: A state in a Markov chain is periodic if the chain can return to the state only at multiples of some integer larger than 1.

Let A, B and C denote the states in a 3-state Markov chain. The following chain is *periodic* with period 3, where the period is the number of steps that it takes to return to a certain state.



A Markov chain is *ergodic* if it is irreducible, aperiodic, and all its states are nonnull and recurrent.

Markov Chain Monte Carlo

- Let $\{\mathbf{X}^{(t)}\}$ denote a Markov chain for $t = 0,1,2 \ldots, T$, where $\mathbf{X}^{(t)} = \left(X_1^{(t)}, \ldots, X_p^{(t)}\right)$ and the state space is either discrete or continuous.

- Given a target distribution f, a Markov kernel $\mathbf{P}$ can be built having stationary distribution f from where the Markov chain can be generated for approximating integrals according to the ergodic theorem.

- There exists methods for deriving kernels that are universal in that they are theoretically valid for any density f.

- A popular application of MCMC methods is to facilitate Bayesian inference where f is a Bayesian posterior distribution for parameters $\mathbf{X}$.

Metropolis-Hastings

The Metropolis-Hastings is a very general algorithm for constructing a Markov chain. Given starting value $\mathbf{X}^{(t)} = \mathbf{x}^{(t)}$, the algorithm generates $\mathbf{X}^{(t+1)}$ as follows:

1. Sample a candidate value $\mathbf{X}^*$ from *proposal distribution* $q\left(\cdot\,|\mathbf{x}^{(t)}\right)$

2. Compute the *Metropolis-Hastings ratio*

$$R\left(\mathbf{x}^{(t)}|\mathbf{X}^*\right) = \frac{f(\mathbf{X}^*)q\left(\mathbf{x}^{(t)}|\mathbf{X}^*\right)}{f(\mathbf{x}^{(t)})q\left(\mathbf{X}^*|\mathbf{x}^{(t)}\right)}$$

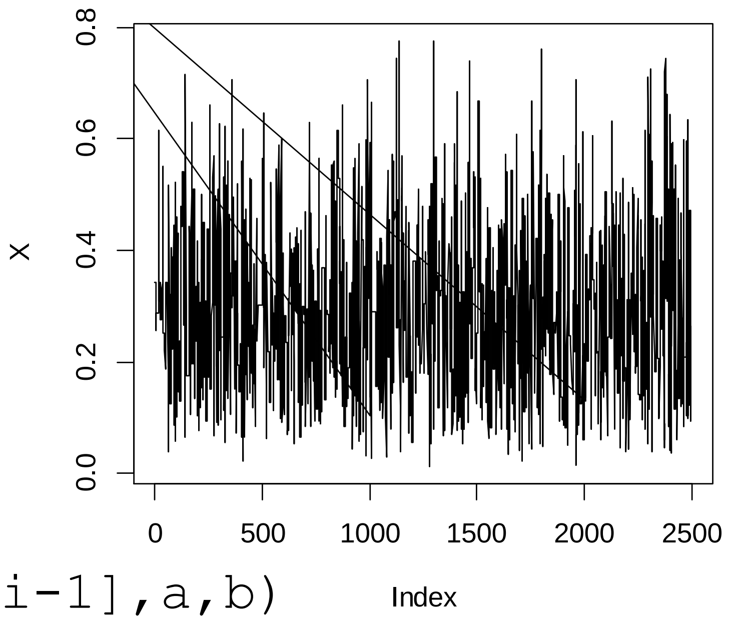
3. Accept a new value according to

$$\mathbf{X}^{(t+1)} = \begin{cases} \mathbf{X}^* & \text{with probability } \min\left\{R\left(\mathbf{x}^{(t)}|\mathbf{X}^*\right), 1\right\} \\ \mathbf{x}^{(t)} & \text{otherwise} \end{cases}$$

4. Increment *t* and return to 1.

R: Metropolis-Hastings

Simple example: Metropolis-Hastings sampling for simulating a sample from a Beta target with uniform candidate



```
Nsim <- 2500
a <- 2.7; b <- 6.3
X <- rep(runif(1),Nsim)
for (i in 2:Nsim){
  Xstar <- runif(1)
  R <- dbeta(Xstar,a,b)/dbeta(X[i-1],a,b)
  X[i] <- X[i-1] + (Xstar-X[i-1])*(runif(1)<R)
}
plot(X,type="l")
```

Metropolis-Hastings

Metropolis-Hastings is closely related to the Simulated Annealing algorithm, two major differences being:

1. The main goal of the SA algorithm is to maximize function $h(x)$ whereas to goal of the MH algorithm is to explore the support of the density f according to its probability.

2. The second difference is in their convergence properties. With the proper choice of a temperature schedule the SA algorithm converges to the maxima of function h, while the MH algorithm is converging to the distribution f itself. Modifying the proposal can have drastic consequences on the convergence pattern.

Metropolis-Hastings

Choice of proposal distribution q:

- When the proposal distribution is symmetric so that $q(\mathbf{x}^{(t)}|\mathbf{X}^*) = q(\mathbf{X}^*|\mathbf{x}^{(t)})$, the method is called the Metropolis algorithm.

- If a proposal distribution is too diffuse relative to the target distribution, the candidate values will be rejected frequently and the chain will require many iterations to explore to full space of the target distribution.

- If a proposal distribution is too narrow relative to the target distribution, then the chain will remain in one small region of the target distribution for many iterations.

Metropolis-Hastings

Independence chains:

- In the prior M-H algorithm the candidate distribution depends on the present state of the chain. The proposal distribution can instead be chosen such that it is independent of the current state for some fixed density g:

$$R\left(\mathbf{x}^{(t)}|\mathbf{X}^*\right) = \frac{f(\mathbf{X}^*)g\left(\mathbf{x}^{(t)}\right)}{f(\mathbf{x}^{(t)})g(\mathbf{X}^*)}$$

- This method is related to the rejection sampler since R can be expressed as the ratio of importance ratios $R\left(\mathbf{x}^{(t)}|\mathbf{X}^*\right) = w^*/w^{(t)}$.

- The rejection samples are i.i.d., while the M-H samples are not.

- The rejection algorithm requires the calculation of an upper bound, while the M-H doesn't.

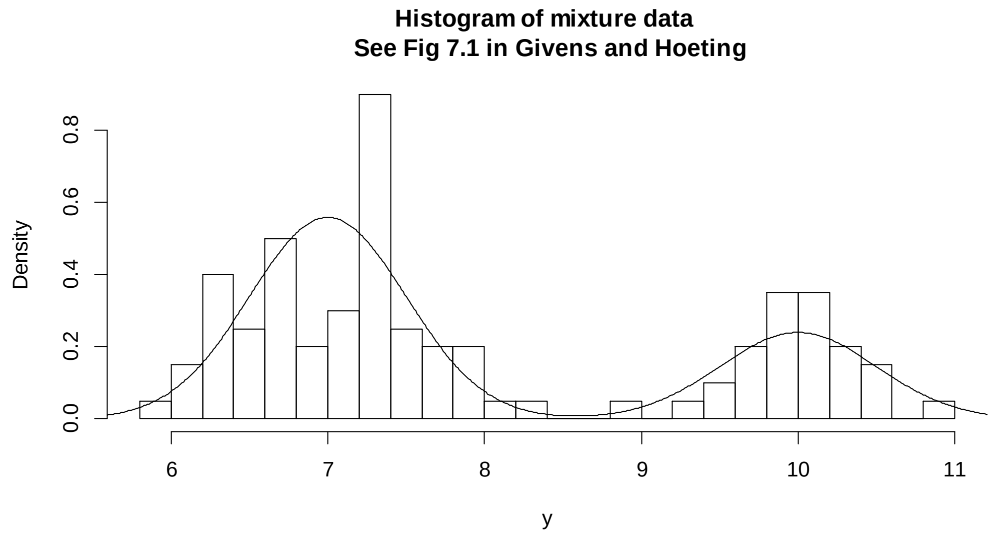
R: Metropolis-Hastings

Independence chains, Example 7.2 (Givens and Hoeting):

100 data points sampled from the mixture distribution:

$$y \sim \delta N(7, 0.5^2) + (1-\delta) N(10, 0.5^2)$$

where the mixture proportion is $\delta = 0.7$.



R: Metropolis-Hastings

Independence chains, Example 7.2 (Givens and Hoeting):

Evaluate the effect of two different Beta proposal densities on delta:

```
## INITIAL VALUES
Nsim <- 2500
x.val1 <- NULL
x.val2 <- NULL
set.seed(0)

## FUNCTIONS
f <- function(x){prod(x*dnorm(y,7,0.5) +
     (1-x)*dnorm(y,10,0.5))}
R <- function(xt,x){f(x)*g(xt)/(f(xt)*g(x))}
```

R: Metropolis-Hastings

Independence chains, Example 7.2 (Givens and Hoeting):

```
## M-H IND SAMP WITH BETA(1,1) PROPOSAL DENSITY
g <- function(x){dbeta(x,1,1)}
x.val1[1] <- rbeta(1,1,1)
for(i in 1:Nsim){
      xt <- x.val1[i]
      x <- rbeta(1,1,1)
      p <- min(R(xt,x),1)
      d <- rbinom(1,1,p)
      x.val1[i+1] <- x*d + xt*(1-d)
}
```

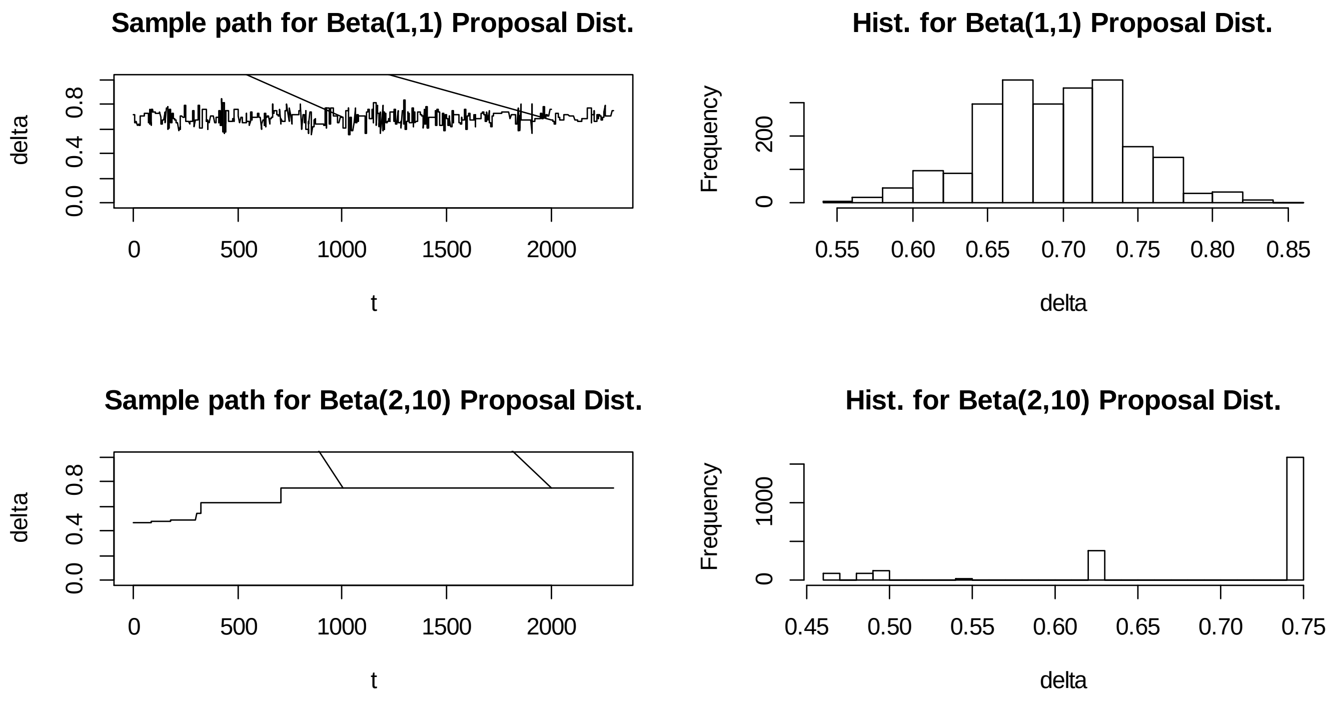
R: Metropolis-Hastings

Independence chains, Example 7.2 (Givens and Hoeting):

```
## M-H IND SAMP WITH BETA(2,10) PROPOSAL DENSITY
g <- function(x){dbeta(x,2,10)}
x.val2[1] <- rbeta(1,2,10)
for(i in 1:Nsim){
      xt <- x.val2[i]
      x <- rbeta(1,2,10)
      p <- min(R(xt,x),1)
      d <- rbinom(1,1,p)
      x.val2[i+1] <- x*d + xt*(1-d)
}
```

R: Metropolis-Hastings

Independence chains, Example 7.2 (Givens and Hoeting):



Metropolis-Hastings

Random walk chains:

- The random walk chain is a variant of the M-H algorithm where $\mathbf{X}^* = \mathbf{x}^{(t)} + \boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim h(\boldsymbol{\epsilon})$.
- It yields a proposal distribution of the form $g\left(\mathbf{x}^* \middle| \mathbf{x}^{(t)}\right) \sim h\left(\mathbf{x}^* - \mathbf{x}^{(t)}\right)$.
- Common choices for h include a uniform distribution centered at the origin, a scaled standard normal distribution and a scaled Student-t.

R: Metropolis-Hastings

Random walk chains, simple example:

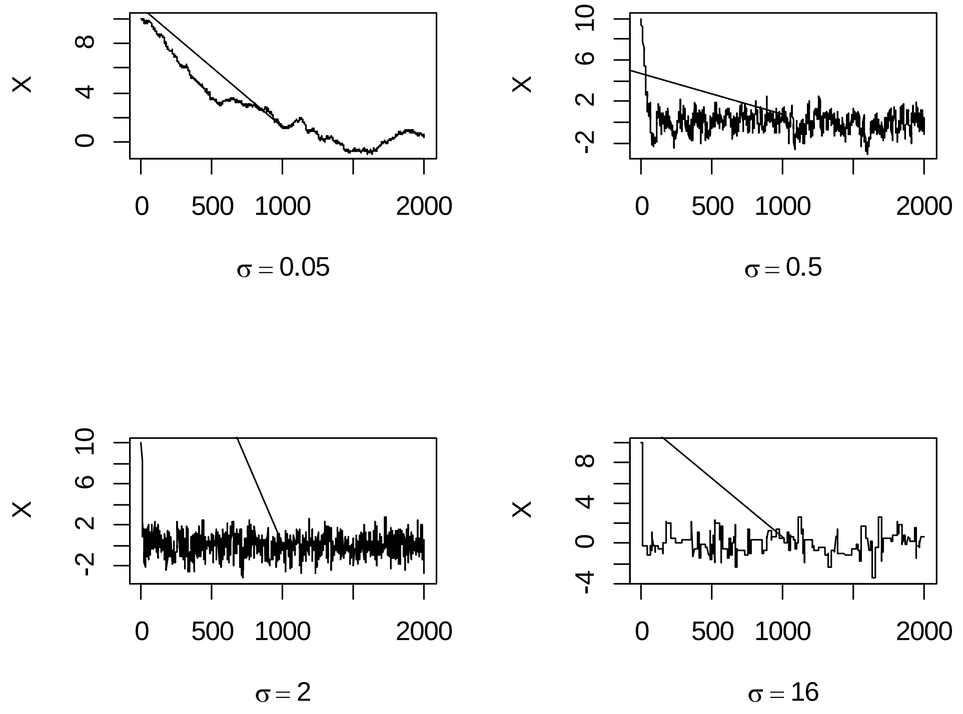
Simulating from a $N(0,1)$ to illustrate the effect of normal proposals with different variances:

```
MH.RW <- function(sigma, x0, Nsim) {
        x <- numeric(Nsim)
        x[1] <- x0
        u <- runif(Nsim)
        t <- 0
        for (i in 2:Nsim) {
           Xstar <- x[i-1]+rnorm(1,0,sigma)
           if (u[i] <= dnorm(Xstar,0,1)/dnorm(x[i-1],0,1))
                 x[i] <- Xstar
           else {
                 x[i] <- x[i-1]
                 t <- t+1
                }
            }
        return(list(x=x, t=t))
        }
```

R: Metropolis-Hastings

Random walk chains, simple example:

```
Nsim <- 20000 # number of iterations
sigma <- c(.05, .5, 2,  16) # sd proposal distribution
x0 <- 10 # initial mean value
rw1 <- MH.RW(sigma[1],
    x0, Nsim)
rw2 <- MH.RW(sigma[2],
    x0, Nsim)
rw3 <- MH.RW(sigma[3],
    x0, Nsim)
rw4 <- MH.RW(sigma[4],
    x0, Nsim)
```



R: Metropolis-Hastings

Random walk chains, Example 7.3 (Givens and Hoeting):

Mixture distribution, continued:

The model is reparameterized with a logit function $U = \text{logit}(\delta)$ and the random walk is performed in the *u*-space:

$U^* = u^{(t)} + \epsilon$ where $\epsilon \sim \text{Unif}(-1,1)$ or $\epsilon \sim \text{Unif}(-0.01,0.01)$.

```
#ESTABLISH INITIAL VALUES
set.seed(1)
Nsim <- 2500
u <- rep(0,Nsim)
u[1] <- runif(1,-1,1)
p <- rep(0,Nsim)
p[1] <- exp(u[1])/(1+exp(u[1]))
```

R: Metropolis-Hastings

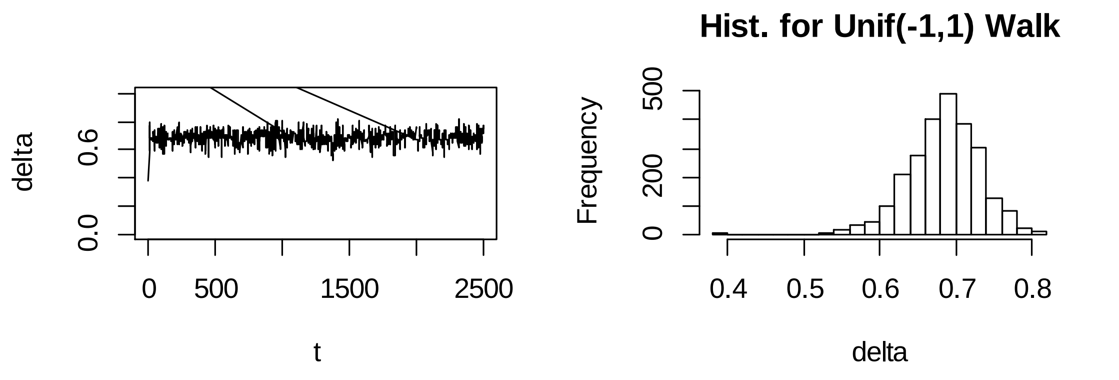
Random walk chains, Example 7.3 (Givens and Hoeting):

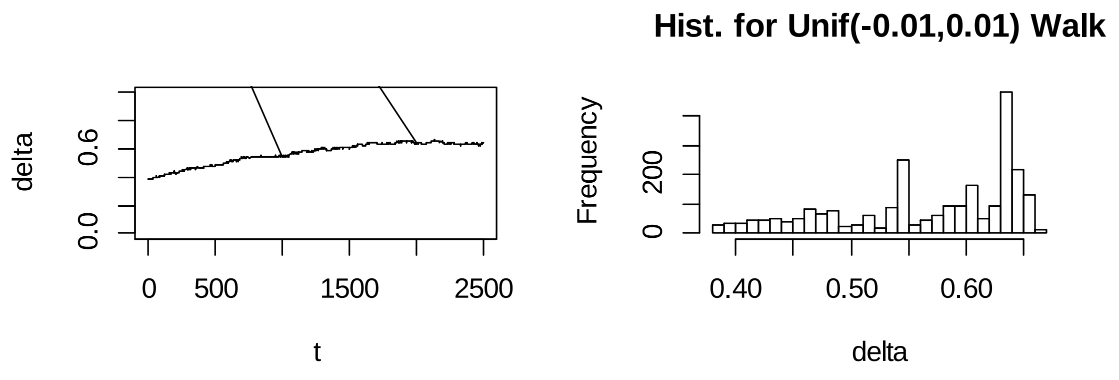
```
## FUNCTION LOG SCALE
log.like<-function(p,x) {
        sum(log(p*dnorm(y,7,.5)+(1-p)*dnorm(y,10,.5)))
}

## M-H RAND WALK SAMP WITH UNIF(-1,1) PROPOSAL DENSITY
for (i in 1:(Nsim-1)) {
      u[i+1] <- u[i]+runif(1,-1,1)
      p[i+1] <- exp(u[i+1])/(1+exp(u[i+1]))
      R <- exp(log.like(p[i+1],x)-
log.like(p[i],x))*exp(u[i])/exp(u[i+1])
      if (R<1)
             if(rbinom(1,1,R)==0)    {p[i+1] <- p[i];
u[i+1] <- u[i]}
}
```

R: Metropolis-Hastings

Random walk chains, Example 7.3 (Givens and Hoeting):





Gibbs sampling

- It is difficult to construct M-H samplers for high dimensional target functions.
- The appeal of the Gibbs sampler algorithm is that it first gather most of the calibration from the target density and then provides a consistent procedure for breaking up complex problems into a series of easier problems.
- The goal is still to construct a Markov chain whose stationary distribution—or some marginalization thereof—equals the target density.
- Sequentially samples from univariate conditional distributions, which needs to be available in closed form.

Gibbs sampling

<u>Two-stage Gibbs sampler:</u>

Creates a Markov chain from a joint distribution of two random variables *X* and *Y* according to:

Set the initial values to $X^{(0)} = x^{(0)}$ and $Y^{(0)} = y^{(0)}$.

For $t = 1,2 \ldots Nsim$

1. $Y^{(t)} \sim f_{Y|X}(\cdot\,|x_{t-1})$
2. $X^{(t)} \sim f_{X|Y}(\cdot\,|y_t)$

R: Gibbs sampling

Two-stage Gibbs sampler:

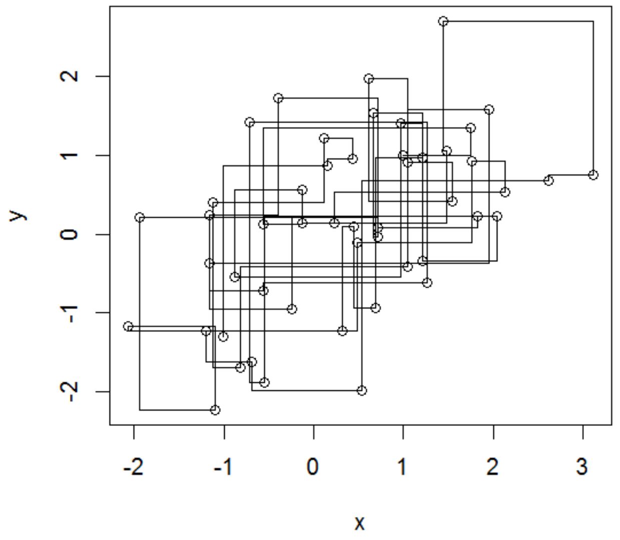
Simple example sampling from a joint bivariate normal density $f(x, y)$.

```
## Initial values
NSim <- 50
rho <- 0.5
x0 <- 1
y0 <- 1
x <- rep(0, NSim)
y <- rep(0, NSim)
x[1] <- x0
y[1] <- y0
```

R: Gibbs sampling

```
## Bivariate Gibbs sampler
for(i in 2:NSim) {
  ## sample the x direction conditional on (prior) y
  x[i] <- rnorm(1, rho*y[i-1], sqrt(1-rho^2))
  ## sample the y direction conditional on x
  y[i] <- rnorm(1, rho*x[i], sqrt(1-rho^2))
}

plot(x,y)
for(i in 2:NSim) {
 lines(c(x[i-1], x[i]),
   c(y[i-1], y[i-1]))
 lines(c(x[i], x[i]),
   c(y[i-1], y[i]))
}
```



Gibbs sampling

<u>Multi-stage Gibbs sampler:</u>

There is a natural extension from the two-stage to the multi-stage Gibbs sampler. Suppose $\mathbf{X} = (X_1, \ldots, X_p)$ and that it is possible to sample from the corresponding conditional densities $f_1, \ldots, f_p$:

Set the initial values to $\mathbf{X}^{(0)} = \left(x_1^{(0)}, \ldots, x_p^{(0)}\right)$.

For $t = 1,2 \ldots Nsim$

1. $X_1^{(t+1)} \sim f_1\left(x_1 \middle| x_2^{(t)}, \ldots, x_p^{(t)}\right)$

2. $X_2^{(t+1)} \sim f_2\left(x_2 \middle| x_1^{(t+1)}, x_3^{(t)} \ldots, x_p^{(t)}\right)$

P. $X_p^{(t+1)} \sim f_p\left(x_p \middle| x_1^{(t+1)}, \ldots, x_{p-1}^{(t+1)}\right)$

R: Gibbs sampling

Many options to do Gibbs sampling in R:

CRAN Task View Bayesian provides a good overview.

Packages that link R to other sampling engines include:

- R2WinBUGS which links to WinBUGS and OpenBUGS
- R2jags which links to JAGS
- RStan (not on CRAN)

There are also some less flexible general packages:

- MCMCpack
- MCMCglmm

R: Gibbs sampling

Example logistic fixed regression using MCMCpack:

```
library(MCMCpack)
data(birthwt)
posterior <-
MCMClogit(low~age+as.factor(race)+smoke,
  data=birthwt)
summary(posterior)
```

R: Gibbs sampling

Example logistic fixed regression using MCMCpack:

```
Iterations = 1001:11000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                      Mean     SD Naive SE Time-series SE
(Intercept)       -0.98190 0.9109 0.009109       0.038909
age               -0.03812 0.0341 0.000341       0.001403
as.factor(race)2   1.03851 0.5002 0.005002       0.020253
as.factor(race)3   1.08242 0.4185 0.004185       0.017995
smoke              1.12993 0.3922 0.003922       0.016334

2. Quantiles for each variable:

                      2.5%      25%      50%      75%   97.5%
(Intercept)       -2.72363 -1.60072 -1.00723 -0.31925 0.87166
age               -0.11025 -0.06131 -0.03675 -0.01468 0.02623
as.factor(race)2   0.06971  0.69756  1.03927  1.38325 2.01703
as.factor(race)3   0.30514  0.79286  1.06132  1.36351 1.94954
smoke              0.38584  0.85418  1.14129  1.39460 1.88826
```

Advanced MCMC

Many recent advancements in different areas of MCMC modelling:

Two particular notable innovations:

1. Adaptive MCMC where proposal distributions are adapted while the algorithm is running.

2. Variable dimension algorithms, particularly the Reversible Jump MCMC

Advanced MCMC

<u>Adaptive MCMC</u>

Random walk Metropolis-within-Gibbs sampling:

Useful when the (univariate) conditional density isn't available for some element of $\mathbf{X} = (X_1, \dots, X_p)$ in an ordinary Gibbs sampler. The goal of the AMCMC is to tune the variance of the proposal distribution so that it yields an optimal acceptance rate.

For the element that lacks closed conditional density:

- Metropolis-Hasting step
- Gibbs update of the other elements
- Adaptation step of the proposal

In R: `adaptMCMC()`

Advanced MCMC

Reversible Jump MCMC

Standard MCMC algorithms require that the dimensionality of $\mathbf{X}^{(t)}$ and the interpretation of the elements of $\mathbf{X}^{(t)}$ do not change with *t.*

In many applications, it may be of interest to develop a chain that allows for changes in the dimension *m* of the parameter space from one iteration to the next.

Examples where RJMCMC methods are useful:
– model selection
– selection of the number of components in a mixture distribution
– knot selection in nonparametric regression

Advanced MCMC

Reversible Jump MCMC

Consider constructing a Markov chain to explore a space of candidate models, each of which might be used to fit observed data *y*.

Let

- $\mathcal{M}_1, \ldots, \mathcal{M}_K$ be a collection of countable models.
- θ_m be the parameters for model *m.*
- p_m be the number of parameters in model *m.*

Random draws from the posterior distribution in a Bayesian RJMCMC model will then be guided by prior distributions of varying model and parameter dimensions

$\theta^{(t)}_{M^{(t)}}$ parameters drawn at iteration *t* for model $\mathcal{M}^{(t)}$.

$p^{(t)}_{M^{(t)}}$ dimension of $\theta^{(t)}_{M^{(t)}}$ at iteration *t.*

Advanced MCMC

Reversible Jump MCMC

The posterior distribution can be factorized

$$f(m, \theta_m | \boldsymbol{y}) = f(m|\boldsymbol{y}) f(\theta_m | m, \boldsymbol{y})$$

which leads to two major results

1. $f(m|\boldsymbol{y})$ is the posterior probability for the *m*th model, normalized over all models under consideration.

2. $f(\theta_m | m, \boldsymbol{y})$ is the posterior density of the parameters in the *m*th model.

R: Advanced MCMC

Example of Reversible Jump MCMC in R

Package `BMA` includes several options for Bayesian model selection. The function `MC3.REG()` is a version of RJMCMC called Markov Chain Monte Carlo Model Composition (MC3).

This function can be applied to the baseball.dat data that was analysed with simulated annealing and genetic algorithms for variable selection.

```
library(BMA)
data <- read.table("baseball.dat",header=TRUE)
pred <- data[,-1]
salary.log <- log(data$salary)
Mc3 <- MC3.REG(salary.log, pred,
  num.its=100000,rep(TRUE,27), outliers = FALSE)
```

R: Advanced MCMC

```
Example of Reversible Jump MCMC in R
 24058  models were selected
 Best  5  models (cumulative posterior probability =  0.2055 ):

                prob     model 1  model 2  model 3  model 4  model 5
variables
  average       0.03032   .        .        .        .        .
  obp           0.02760   .        .        .        .        .
  runs          0.65245   .        x        x        x        x
  hits          0.39126   x        .        .        .        .
  doubles       0.03751   .        .        .        .        .
  triples       0.03794   .        .        .        .        .
  homeruns      0.06455   .        .        .        .        .
  rbis          0.96889   x        x        x        x        x
  walks         0.07319   .        .        .        .        .
  sos           0.59122   .        x        x        x        x
  sbs           0.07823   .        .        .        .        .
  errors        0.07189   .        .        .        .        .
  freeagent     1.00000   x        x        x        x        x
  arbitration   1.00000   x        x        x        x        x
  runsperso     0.27751   .        x        .        .        x
  hitsperso     0.14713   .        .        .        .        x
  hrsperso      0.03212   .        .        .        .        .
  rbisperso     0.05997   .        .        .        .        .
  walksperso    0.02966   .        .        .        .        .
  obppererror   0.11122   .        .        .        .        .
  runspererror  0.15005   .        .        .        x        .
  hitspererror  0.19789   .        .        .        x        .
  hrspererror   0.02800   .        .        .        .        .
  soserrors     0.40525   x        .        .        .        .
  sbsobp        0.10225   .        .        .        .        .
  sbsruns       0.05768   .        .        .        .        .
  sbshits       0.05630   .        .        .        .        .

post prob                0.05902  0.04953  0.04858  0.02576  0.02261
```

Reading

- Book (Givens and Hoeting)

  Chapter: (1), 7, 8

- CRAN Task View: Bayesian Statistics