Statistical Computing with R – MATH 6382[1,*] Set 8 (Markov Chain Monte Carlo (MCMC))

Tamer Oraby

UTRGV

tamer.oraby@utrgv.edu

[1] Based on textbook.

[*] Last updated December 12, 2016

Introduction (Bayesian Computations)

Bayesian statistical analysis

Bayesian statistical analysis is used to

- estimate parameters
- test hypothesis
- select models

Bayesian statistical analysis

By Bayes' theorem for random variables:

$$f_{\theta|x_1,\ldots,x_n}(\theta) = \frac{L(\theta|x_1,\ldots,x_n)f_\theta(\theta)}{\int_\Theta L(\theta|x_1,\ldots,x_n)f_\theta(\theta)d\theta}$$

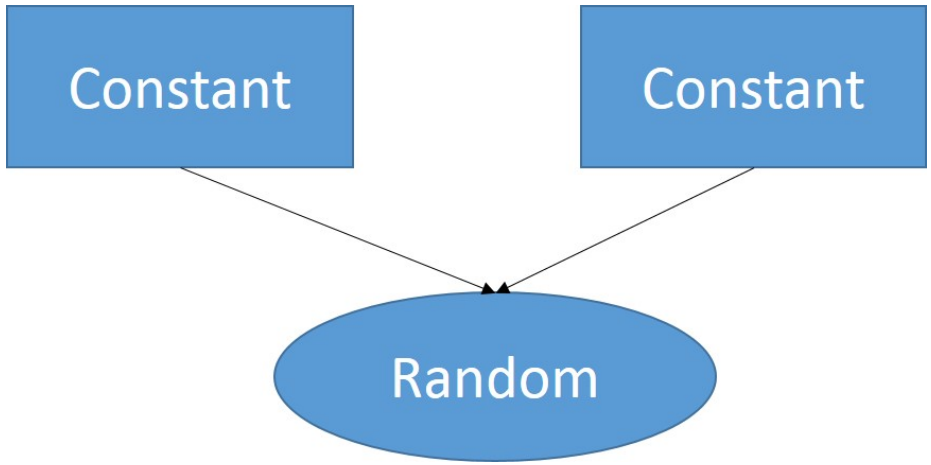
or simply

$$\textit{posterior} \propto \textit{likelihood} \times \textit{prior}$$

with constant of proportionality c where $1/c = \int_\Theta L(\theta|x_1,\ldots,x_n)f_\theta(\theta)d\theta$

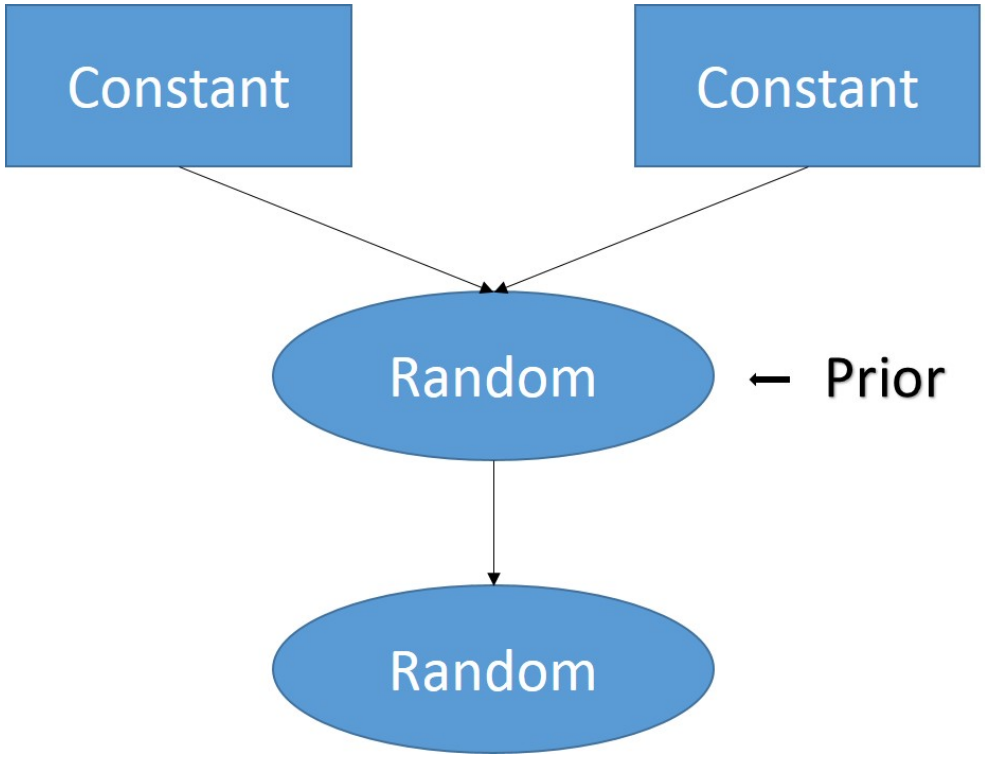
Graphical Models

Acyclic diagrams:



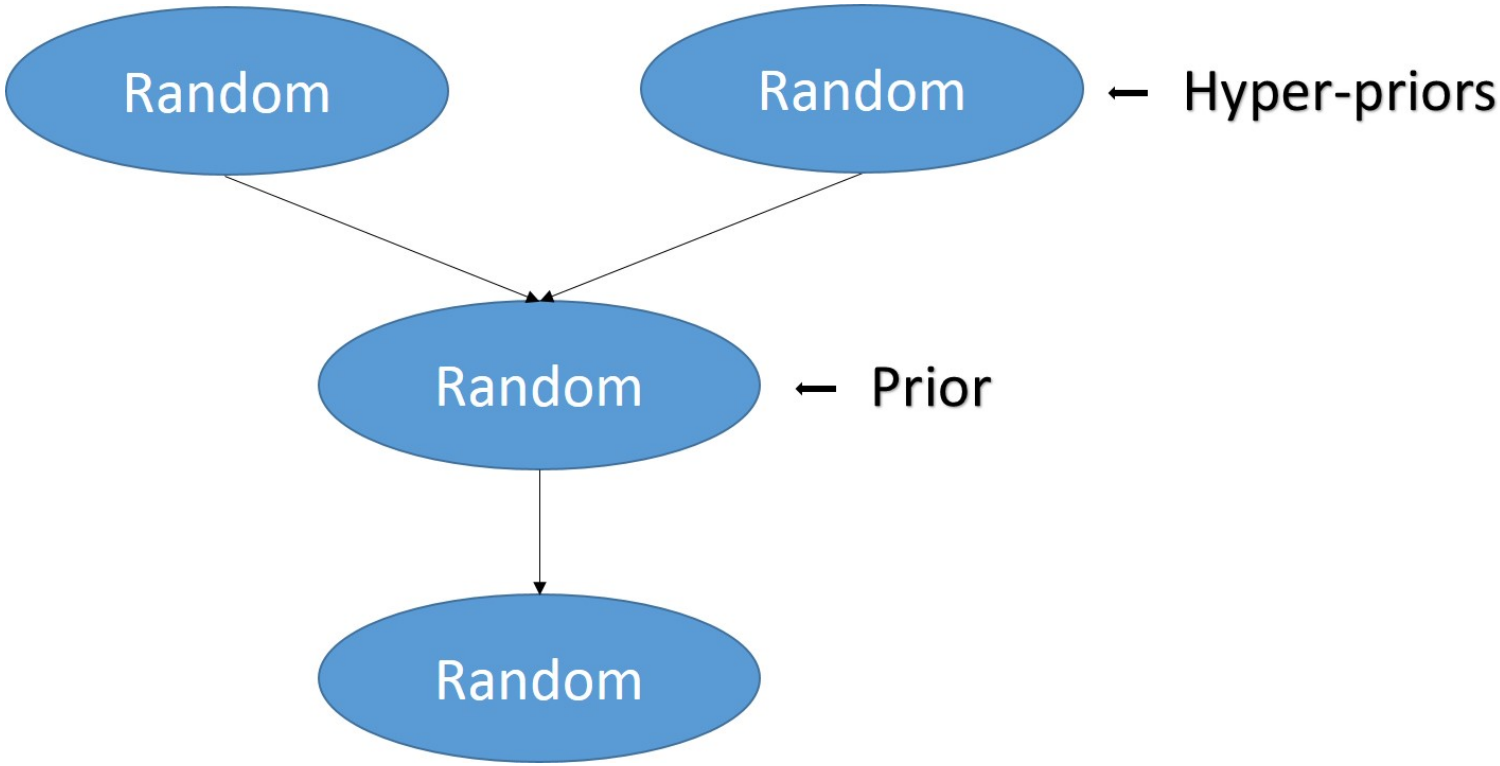
Graphical Models

Hierarchical Model:



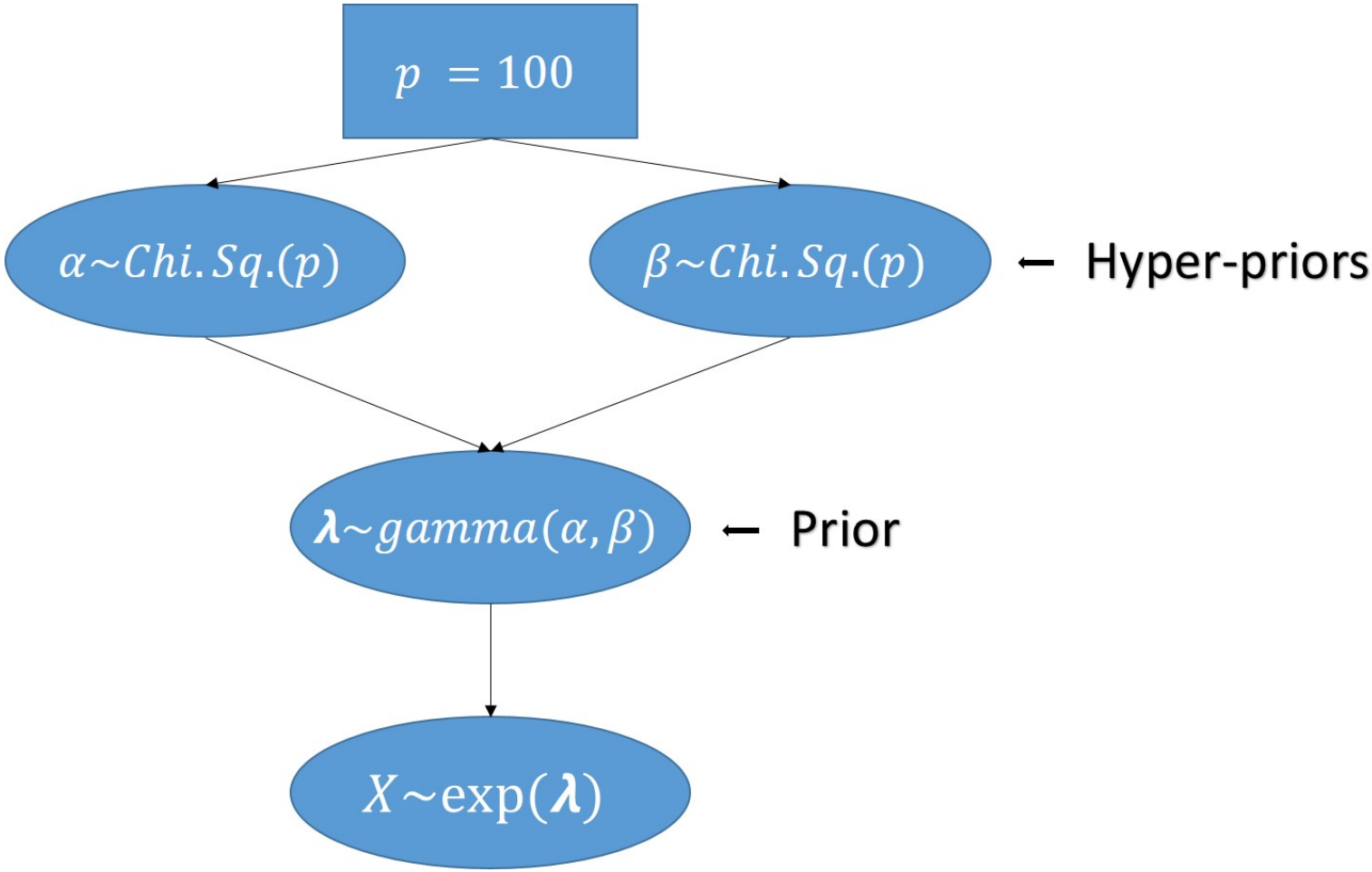
Graphical Models

Hierarchical Model:



Graphical Models

Example (Hierarchical Model):



Graphical Models

Example (Hierarchical Model):

$$[X|\lambda] \sim exp(\lambda)$$

$$[\lambda|\alpha,\beta] \sim gamma(\alpha,\beta)$$

$$[\alpha|p] \sim Chi.Sq.(p)$$

$$[\beta|p] \sim Chi.Sq.(p)$$

$$p = 100$$

The gamma distribution is a prior and the Chi-square is a hyper-prior.
Meanwhile, λ is a parameter and α, β are hyper-parameters.
By Bayes' theorem for random variables:

$$f_{\lambda|x;\alpha,\beta}(\lambda) \propto L(\lambda|x) f_{\lambda|\alpha,\beta}(\lambda) f_{\alpha}(\alpha) f_{\beta}(\beta)$$

Bayesian statistical analysis

- A point estimate $\hat{\theta}$ of a parameter $\theta = (\theta_1, \ldots, \theta_k)$ is given by

$$\hat{\theta} = \mathit{argmin}_{\alpha \in \Theta} \int_{\Theta} \mathit{Loss}(\theta, \alpha) f_{\theta|x_1,\ldots,x_n}(\theta) d\theta$$

.

- If $\mathit{Loss}(x, y) = |x - y|^2$ then

$$\hat{\theta} = \int_{\Theta} \theta f_{\theta|x_1,\ldots,x_n}(\theta) d\theta$$

and in general

$$h(\hat{\theta}) = \int_{\Theta} h(\theta) f_{\theta|x_1,\ldots,x_n}(\theta) d\theta$$

Bayesian statistical analysis

Testing

$$H_0 : \theta \in \Theta_0 \text{ vs } H_1 : \theta \in \Theta_1$$

where the parameters space $\Theta = \Theta_0 \cup \Theta_1$ and $\Theta_0 \cap \Theta_1 = \phi$.
It involves the calculation of $Pr(\theta \in \Theta_0 | x)$ which is comparable to the p-value in the frequentist approach but not the same.
Where, for $i = 0, 1$

$$Pr(\theta \in \Theta_i | x) = \int_{\Theta_i} f_{\theta | x_1, \ldots, x_n}(\theta) d\theta$$

where the posterior is found using a prior f_θ

Bayesian statistical analysis

Bayes factor is defined by

$$B^f_{0,1}(x) = \frac{Pr(\theta \in \Theta_0|x)}{Pr(\theta \in \Theta_1|x)} \frac{\int_{\theta\in\Theta_1} f_\theta(\theta)d\theta}{\int_{\theta\in\Theta_0} f_\theta(\theta)d\theta}$$

If $\int_{\theta\in\Theta_1} f_\theta(\theta)d\theta = \int_{\theta\in\Theta_0} f_\theta(\theta)d\theta$ which is usually assumed in tests of hypothesis then

$$B^f_{0,1}(x) = \frac{Pr(\theta \in \Theta_0|x)}{Pr(\theta \in \Theta_1|x)}$$

and so

$$Pr(\theta \in \Theta_0|x) = \frac{1}{1 + B^f_{0,1}(x)}$$

Bayes factor is used to decide which hypothesis is accepted.

Bayesian statistical analysis

$100(1-\alpha)\%$ Bayesian confidence intervals are based on the posterior distribution and can be found from the highest posterior region

$$\{\theta : f_{\theta|x_1,\ldots,x_n}(\theta) \geq \ell(x)\}$$

where $\ell(x)$ is such that

$$Pr^f(f_{\theta|x_1,\ldots,x_n}(\theta) \geq \ell(x)|x) = \alpha$$

which is also the $1-\alpha$ quantile of the random variable $f_{\theta|x_1,\ldots,x_n}(\theta)$.

Bayesian statistical analysis

And so they all require evaluating

$$\mathbf{E}(h(\theta)) = \int_{\Theta} h(\theta) f_{\theta|x_1,\ldots,x_n}(\theta) d\theta$$

with cases like $h(\theta) = \theta$, $h(\theta) = (\theta - \mathbf{E}(\theta))^2$, $h(\theta) = I(\theta \in \Theta_0)$ ... etc.

But how can we find that last integral?

Bayesian statistical analysis

Using **Monte Carlo** (MC) integration, generate $y_1, \ldots, y_m$ random numbers from the posterior $f_{\theta|x_1,\ldots,x_n}(\theta)$

$$\frac{1}{m}\sum_{i=1}^{m} h(y_i) \to \mathbf{E}(h(\theta)) = \int_{\Theta} h(\theta) f_{\theta|x_1,\ldots,x_n}(\theta) d\theta$$

as $m \to \infty$, due to SLLN.

But we need the proportionality constant c which is many cases is difficult to find.

Bayesian statistical analysis

- The idea is to make use of ergodic **Markov chains** (MC) that has a "unique" stationary (invariant) distribution

$$\pi(\theta) := f_{\theta|x_1,\ldots,x_n}(\theta)$$

- Run the Markov chain for long time and throw away a burn-in period of time, first part of the sample path

- Use the last m realization of the sample path $y_1, \ldots, y_m$ (which are dependent but have the same distribution as π) as the sample of random points for which

$$\frac{1}{m}\sum_{t=1}^{m} h(Y_t) \to \mathbf{E}(h(\theta)) = \int_{\Theta} h(\theta)\pi(\theta)d\theta$$

as $m \to \infty$

Markov chains

Markov chains

- Fix a probability space $(\Omega, \mathcal{B}, Pr)$
- Consider a discrete-time continuous-state space space Markov chain

$$\{Y_t(\omega) \in \mathcal{R} : t \in T, \omega \in \Omega\}$$

where $T = \{0, 1, 2, \ldots\}$ and $\mathcal{R}$ is uncountable which is basically Θ.

- Transitions are governed by a transition kernel

$$P(x, A) = Pr(Y_{t+1} \in A | Y_t = x)$$

for $x \in \mathcal{R}$ and $A \subset \mathcal{R}$ due to the Markov property, only the present counts.

- Thus,

$$P(x, A) = \int_A P(x, dy)$$

Markov chains

- Let the function

$$p(.,.) : \mathcal{R} \times \mathcal{R} \to \mathbb{R}^+$$

such that

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy)$$

- where $\delta_x(A) = 1$ only if $x \in A$ and zero otherwise
- and $r(x) = 1 - \int_{\mathcal{R}} p(x, y)dy$ is the probability of transition from x to x

Markov chains

- The n-step transition probability is then

$$P^{(n)}(x, A) = \int_{\mathcal{R}} P(x, dy) P^{(n-1)}(y, A)$$

and

- The stationary distribution π^* with density π with respect to Lebesgue measure is then

$$\pi^*(dy) = \pi(y)dy = \int_{\mathcal{R}} P(x, dy)\pi(x)dx$$

- So, if π^* is the distribution of Y_t then it is also the distribution of Y_{t+1}

Markov chains

- A Markov chain is said to be reversible if there exists a function $f : \mathcal{R} \to \mathbb{R}^+$ such that

$$f(x)p(x,y) = f(y)p(y,x) \text{ Balanced equation}$$

- A reversible MC with a function f implies that $f \equiv \pi$ and so it is the stationary density

Markov chains

$$
\begin{aligned}
\int_{\mathcal{R}} P(x,A)f(x)dx &= \int_{\mathcal{R}}\int_{A} P(x,dy)f(x)dx \\
&= \int_{\mathcal{R}}\int_{A} (p(x,y)dy + r(x)\delta_x(dy))f(x)dx \\
&= \int_{\mathcal{R}}\int_{A} p(x,y)dyf(x)dx + \int_{\mathcal{R}}\int_{A} r(x)\delta_x(dy)f(x)dx \\
&= \int_{A}(\int_{\mathcal{R}} p(x,y)f(x)dx)dy + \int_{\mathcal{R}} r(x)f(x)\delta_x(A)dx \\
&= \int_{A}(\int_{\mathcal{R}} p(y,x)f(y)dx)dy + \int_{A} r(x)f(x)dx \\
&= \int_{A}(\int_{\mathcal{R}} p(y,x)dx)f(y)dy + \int_{A} r(x)f(x)dx \\
&= \int_{A}(1-r(y))f(y)dy + \int_{A} r(x)f(x)dx = \int_{A} f(y)dy
\end{aligned}
$$

Hence, f is a stationary density.

Markov chains

- We can guarantee that the MC is irreducible if the state space $\mathcal{R}$ (parameter space Θ) is connected and the transition function $p(x, y)$ is positive and continuous in y for all x.
- If $r(x) > 0$ then it is aperiodic.
- SLLN:

$$\hat{h}_m := \frac{1}{m} \sum_{t=1}^{m} h(Y_t) \to \int_{\mathcal{R}} h(y)\pi(y)dy$$

almost surely, as $m \to \infty$

- If it is in addition positive recurrent and geometrically ergodic, then besides the SLLN

$$\frac{\hat{h}_m - \mu_h}{\sigma_h/\sqrt{m}} \to N(0, 1) \text{ in distribution}$$

Markov chains

where

$$\mu_h = \int_{\mathcal{R}} h(y)\pi(y)dy$$

and

$$\begin{aligned}
\mathbf{V}(\hat{h}_m) = \frac{1}{m}\sigma_h^2 &= \frac{1}{m^2}\sum_{i,j=1}^{m} Cov(h(Y_i), h(Y_j)) \\
&= \frac{s^2}{m^2}\sum_{i,j=1}^{m} Corr(h(Y_i), h(Y_j)) \\
&= \frac{s^2}{m^2}\sum_{i,j=1}^{m} \rho_{|i-j|} \text{ where } \rho_\ell \text{ is autocorrelation at lag } \ell \\
&= \frac{s^2}{m^2}(m + \sum_{i,j=1;i\neq j}^{m} \rho_{|i-j|}) = \frac{s^2}{m}(1 + 2\sum_{\ell=1}^{m}(1 - \frac{\ell}{m})\rho_\ell))
\end{aligned}$$

Metropolis–Hastings methods (M–H)

Metropolis–Hastings methods (M–H)

- Metropolis (1953) – hydrogen bomb project – M algorithm
- Hastings (1970) – M–H algorithm
- Chib and Greenberg (1995) – multiple block M–H algorithm
- Besag (1974), Geman and Geman (1984), Tanner and Wong – Gibbs sampling algorithm

Metropolis–Hastings methods (M–H)

H–M algorithm has special cases

- Metropolis sampler
- independence sampler
- random walk sampler
- Gibbs sampler

The idea is to build a Markov chain $\{X_t : t \geq 0\}$

Metropolis–Hastings methods (M–H)

H–M algorithm: To have a sample from the target distribution f

1. Choose X_0
2. Choose a proposal distribution $g(\cdot|p)$ with parameter p
3. Repeat the following till convergence to a stationary distribution, for each t such that $t \geq 0$

(i) Generate a random number Y from $g(.|X_t)$

(ii) Generate a uniform random number U from $unif(0,1)$

(iii) Calculate $\alpha(X_t, Y) = \min\{1, \frac{f(Y)g(X_t|Y)}{f(X_t)g(Y|X_t)}\}$

(iv) $X_{t+1} = Y$ if $U \leq \alpha$ and $X_{t+1} = X_t$ if not.

Metropolis–Hastings methods (M–H)

H–M algorithm makes a Markov chain with transition kernel

$$P(x, dy) = \underbrace{\alpha(x, y)g(y|x)}_{p(x,y)}\, dy + r(x)\delta_x(dy)$$

where $r(x) = 1 - \int \alpha(x, y)g(y|x)dy$.
Note: if $\alpha(x, y) = 1$ then transition to y is certain as $r(x) = 0$
Note that

$$\frac{\alpha(x, y)}{\alpha(y, x)} = \frac{f(y)g(x|y)}{f(x)g(y|x)}$$

but then

$$f(x)\underbrace{\alpha(x, y)g(y|x)}_{p(x,y)} = f(y)\underbrace{\alpha(y, x)g(x|y)}_{p(y,x)} \text{ balanced equation}$$

Therefore, f is the stationary distribution of the M–H induced MC.

Special cases of M–H ... Metropolis Sampler

If $g(x|p) = g(|x - p|)$ is symmetric then $g(x|y) = g(y|x)$ and

$$\alpha(x, y) = \min\{1, \frac{f(y)}{f(x)}\}$$

An example for the proposal distribution is $N(p, \sigma^2)$ which is symmetric about p since it involves $(x - p)^2$. Note that the scale parameter plays a role in the speed of convergence.

Special cases of M–H ... Random Walk Metropolis Sampler

Same as in M sampler, but generate the new increment Z using $g(\cdot|0)$

$$Y_{t+1} = \begin{cases} Y_t + Z & \text{if } U \leq \alpha, \\ Y_t & \text{if } U > \alpha, \end{cases}$$

An example for the proposal distribution is $N(0, \sigma^2)$.
Large σ^2 results in large jumps that might be rejected :(

&

Small σ^2 results in small jumps that might be accepted :((a real RW)

Monitor *acceptance rate* and try to have it in [.15, .5]

Special cases of M–H ... Independence Sampler

If $g(y|X_t) = g(y)$ is independent of the previous state X_t

$$\alpha(X_t, y) = \min\{1, \frac{f(y)g(X_t)}{f(X_t)g(y)}\}$$

Works well if g is close to f.

Metropolis–Hastings methods (M–H)

Example: Find the Bayesian point estimate of the rate $\lambda \in \Theta = \mathbb{R}^+$ in an exponential model given the data $x_1, \ldots, x_n$.

```
sample<-c(4.679781, 9.325474, 2.491352, 0.841366,
2.982121)
```

$$L(\lambda|x_1, \ldots, x_n) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i} = \lambda^n e^{-n\lambda \bar{x}}$$

and

$$f_{\lambda|x_1,\ldots,x_n}(\lambda) \propto L(\lambda|x_1, \ldots, x_n) f_\lambda(\lambda)$$

$$= \lambda^n e^{-n\lambda \bar{x}} f_\lambda(\lambda)$$

Use a diffuse prior $gamma(r = .001, \beta = .001)$.

Metropolis–Hastings methods (M–H)

Example: Note that it is the conjugate prior for which the posterior is also $gamma(n + r, n\bar{x} + \beta)$ since

$$f_{\lambda|x_1,\ldots,x_n}(\lambda) \propto \lambda^{n+r-1} e^{-(n\bar{x}+\beta)\lambda}$$

And then the mean of the posterior is

$$\frac{n + r}{n\bar{x} + \beta} = \frac{5 + .001}{5 * 4.064019 + .001} = 0.2460989$$

is the Bayesian point estimate for λ.

Metropolis–Hastings methods (M–H)

On the other hand, implementing M–H algorithm using a proposal distribution of $lognormal(X_t, \sigma^2)$

$$\alpha(X_t, Y) = \min\{1, \frac{Y^{n+r-1} e^{-(n\bar{x}+\beta)Y} * (1/X_t) e^{-(\log X_t - Y)^2/2\sigma^2}}{X_t^{n+r-1} e^{-(n\bar{x}+\beta)X_t} * (1/Y) e^{-(\log Y - X_t)^2/2\sigma^2}}\}$$

Notice that we can ignore the irrelevant constants of g

$$\alpha(X_t, Y) = \min\{1, (Y/X_t)^{n+r} e^{-(n\bar{x}+\beta)(Y-X_t)} e^{-[(\log X_t - Y)^2 - (\log Y - X_t)^2]/2\sigma^2}\}$$

Metropolis–Hastings methods (M–H)

```
sample<-c(4.679781, 9.325474, 2.491352, 0.841366,
2.982121)
n<-length(sample)
r<-beta<-.001
sigma<-2
alpha<-function(x,y){ratio<-((y/x)^(n+r))
*exp(-(n*mean(sample)+beta)*(y-x))
*exp((-1/(2*sigma^2))*((log(x)-y)^2-(log(y)-x)^2))
return(min(1,ratio))}
m<-10000
accept<-0
X<-c(.1)
```

Metropolis–Hastings methods (M–H)

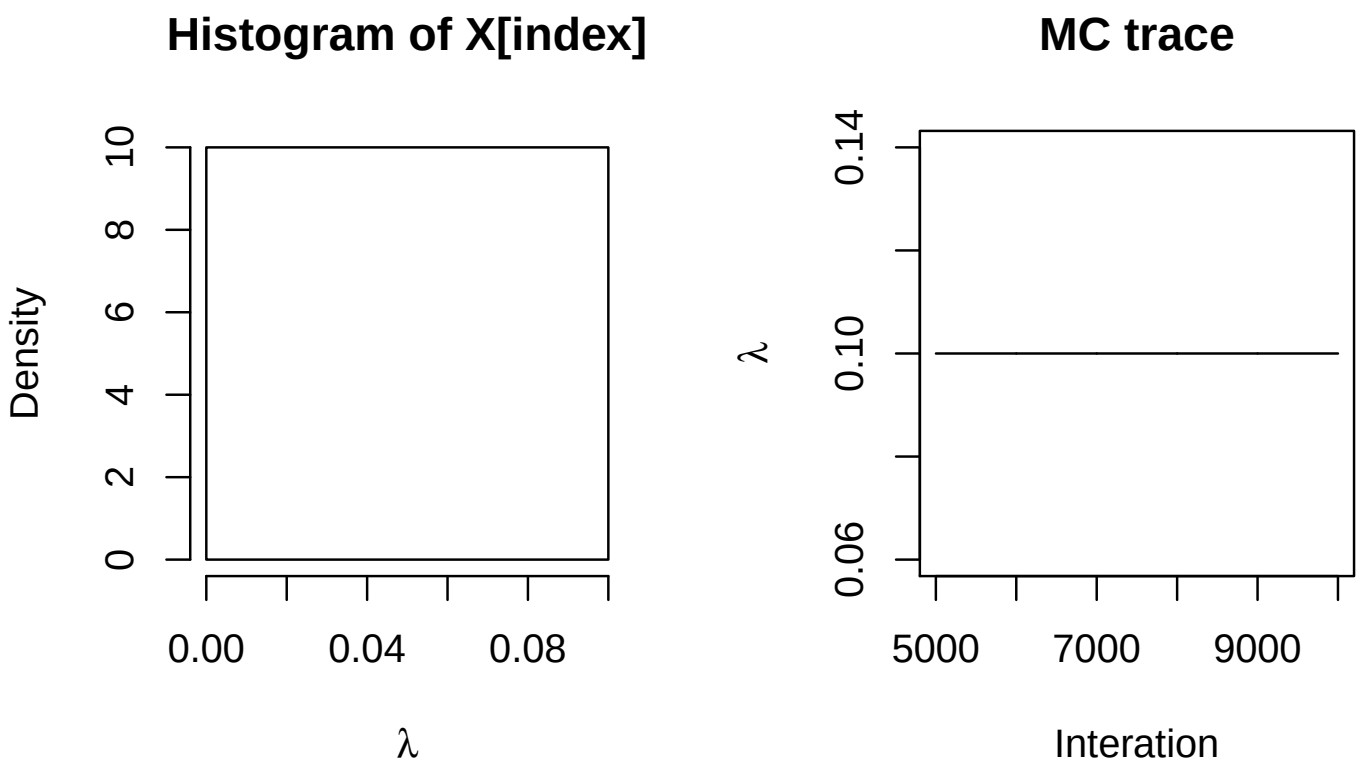
```
for (i in 2:m){
 Y<-rlnorm(1,X[i-1],sigma);U<-runif(1)
 if (U<=alpha(X[i-1],Y)){
 X[i]<-Y
 accept<-accept+1}
 else {
 X[i]<-X[i-1] }}
```

Metropolis–Hastings methods (M–H)

```
acceptancerate<-accept/m
burnin<-m-5000
index<-(burnin+1):m
par(mfrow=c(1,2))
hist(X[index],prob=T,xlab=expression(lambda))
plot(index,X[index],type="l",main="MC
trace",xlab="Interation",ylab=expression(lambda))
acceptancerate
mean(X[index])
median(X[index])
var(X[index])
```

Metropolis–Hastings methods (M–H)

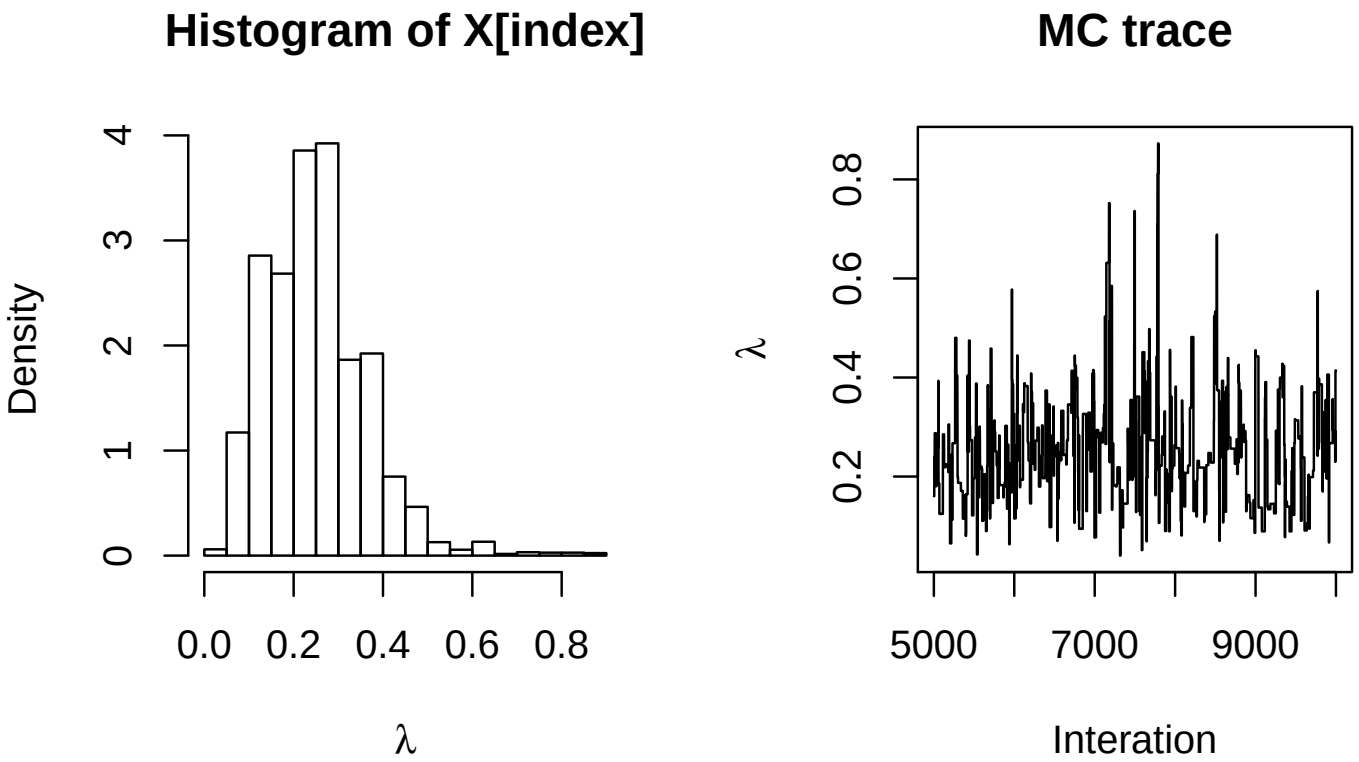
$$X_0 = .1, \sigma = .1$$



acceptance rate=0, mean=.1, variance=0

Metropolis–Hastings methods (M–H)

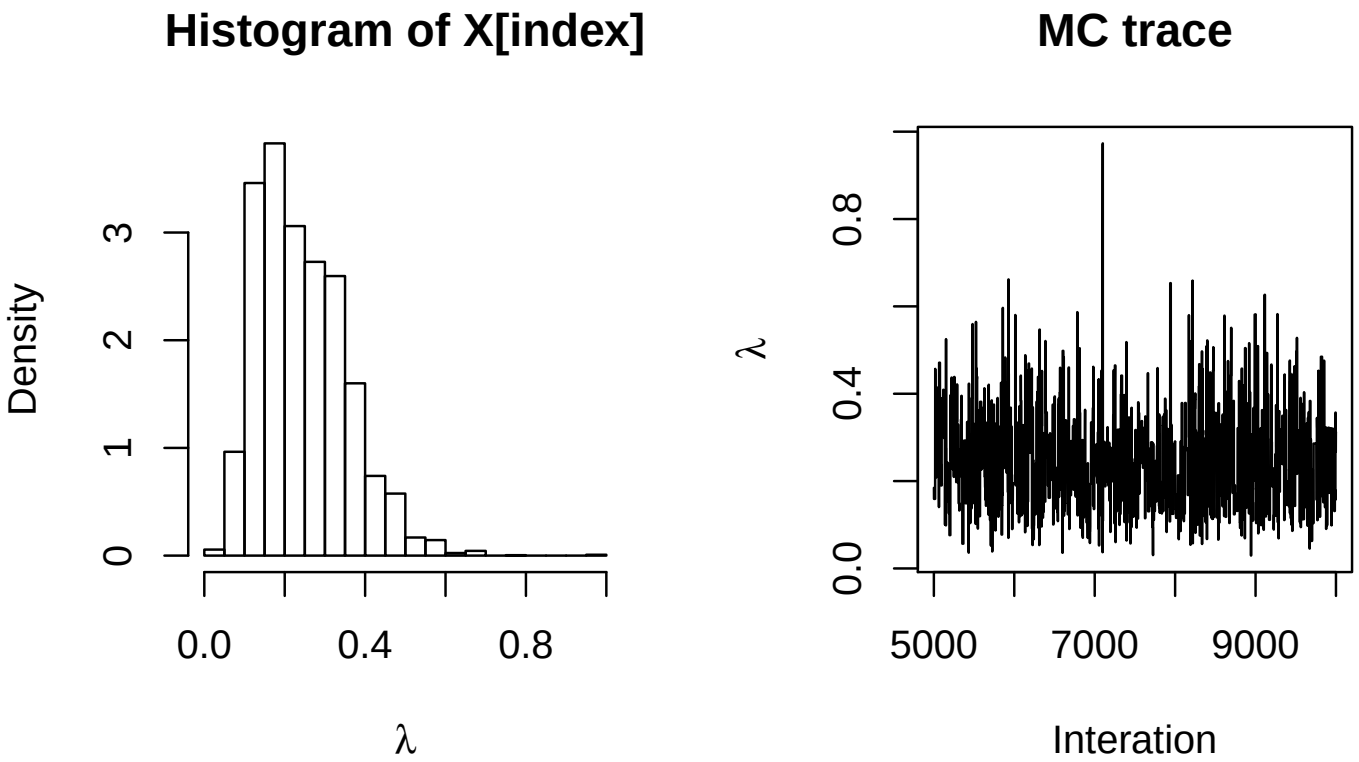
$$X_0 = .1, \sigma = 1$$



acceptance rate=0.0931, mean=0.2518239, variance=0.01250856

Metropolis–Hastings methods (M–H)

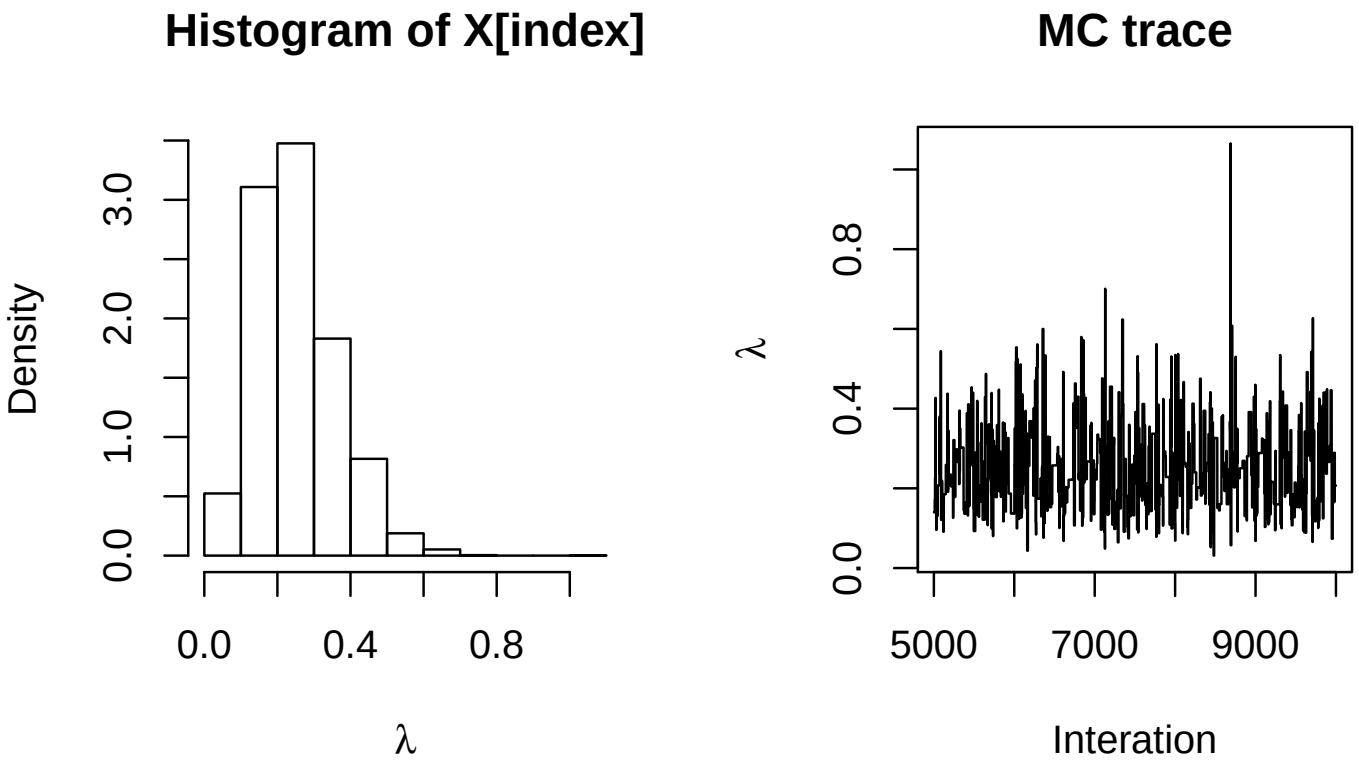
$$X_0 = .1, \sigma = 2$$



acceptance rate=0.2003, mean=0.2442261, variance=0.01218672

Metropolis–Hastings methods (M–H)

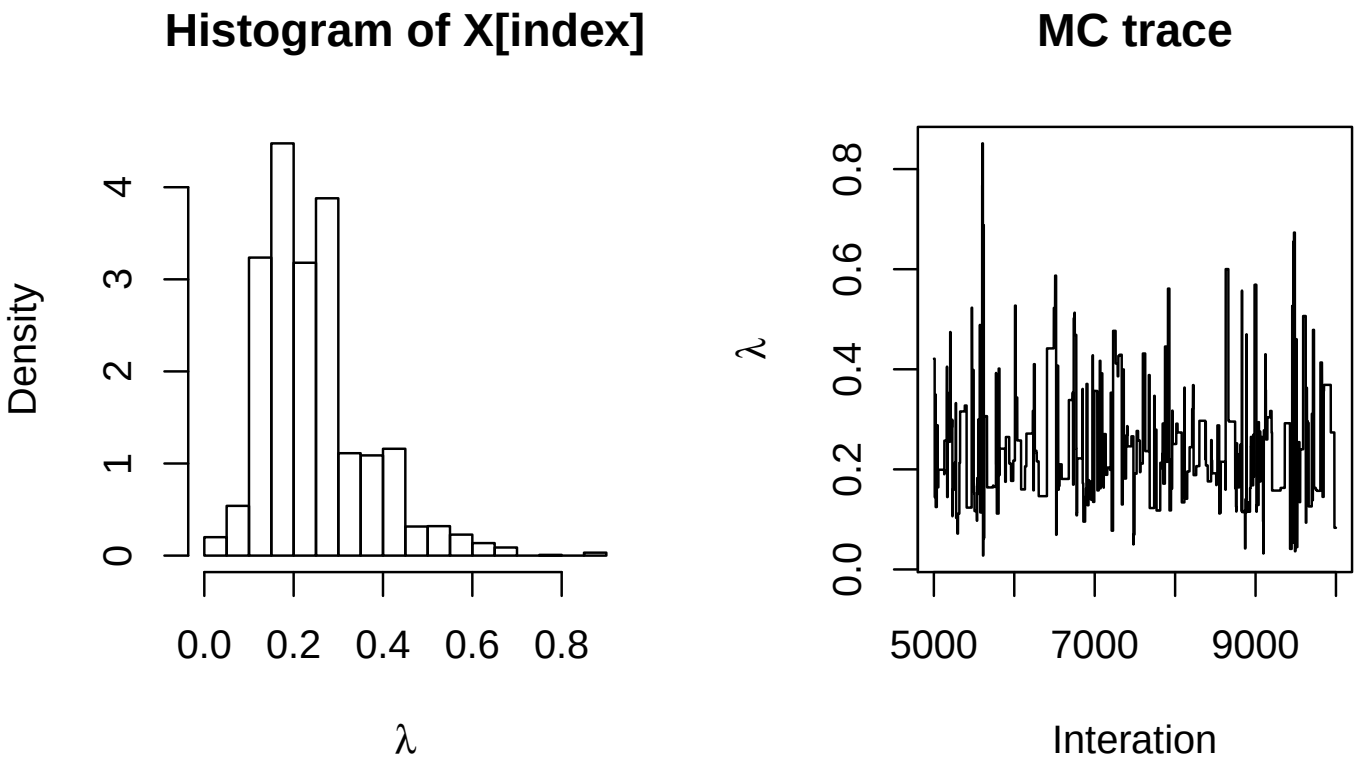
$$X_0 = .1, \sigma = 5$$



acceptance rate=0.1078, mean=0.2487472, variance=0.01192377

Metropolis–Hastings methods (M–H)

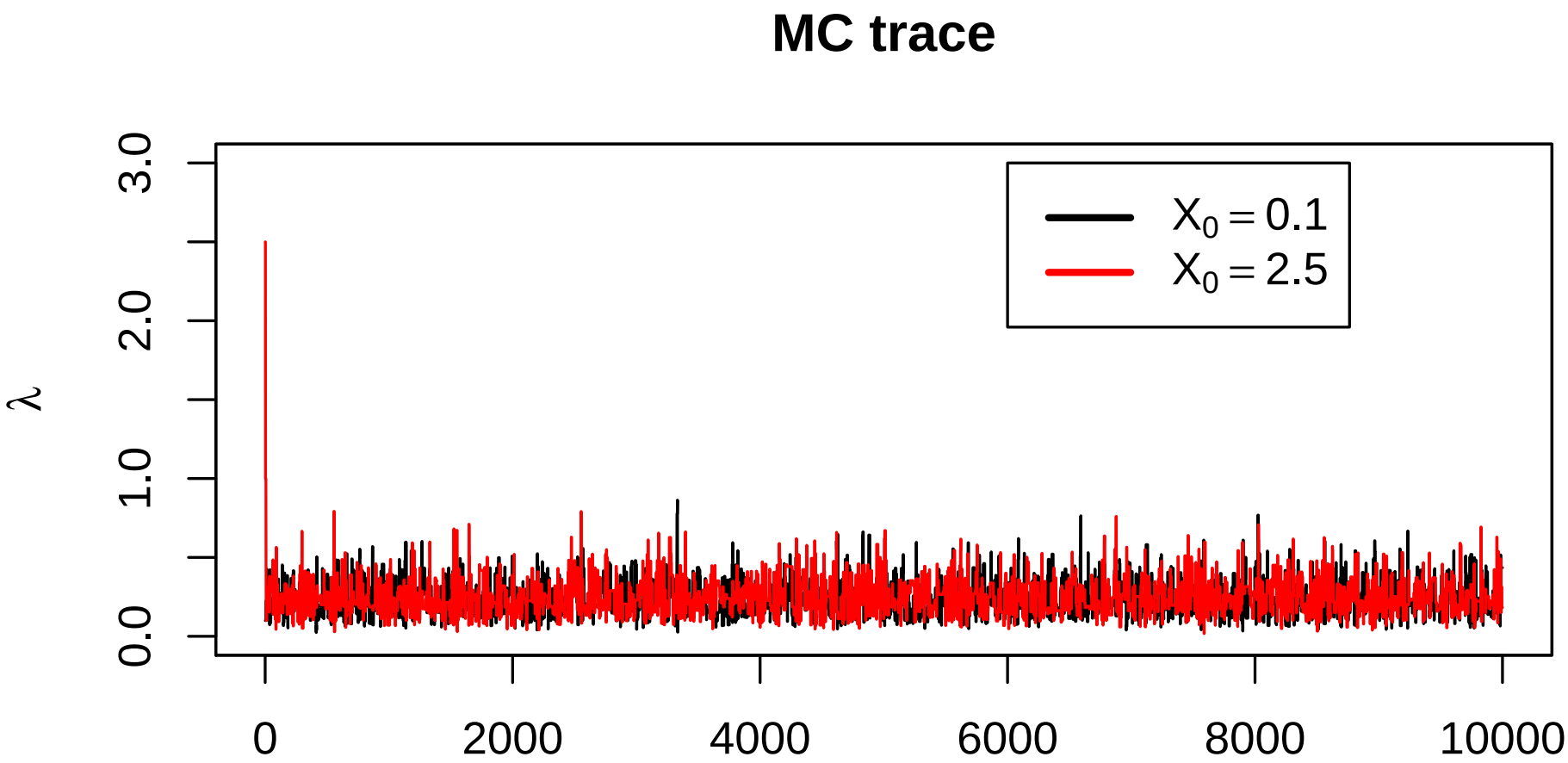
$$X_0 = .1, \sigma = 10$$



acceptance rate=0.0569, mean=0.2447697, variance=0.01365697

Metropolis–Hastings methods (M–H)

$$\sigma = 2$$



Metropolis–Hastings methods (M–H)

On the other hand, implementing M–H algorithm using a proposal distribution of $normal(X_t, \sigma^2)$

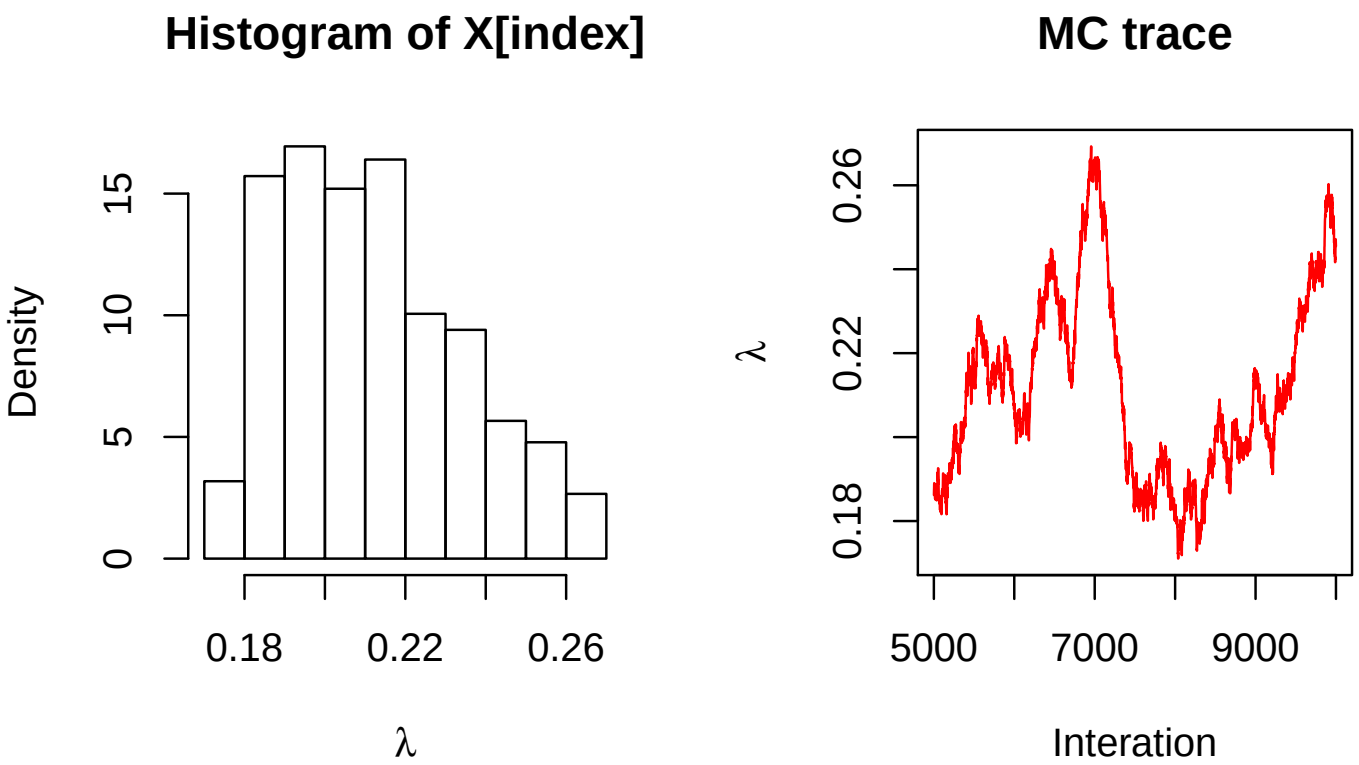
$$\alpha(X_t, Y) = \min\{1, \frac{Y^{n+r-1} e^{-(n\bar{x}+\beta)Y} * e^{-(X_t-Y)^2/2\sigma^2}}{X_t^{n+r-1} e^{-(n\bar{x}+\beta)X_t} * e^{-(Y-X_t)^2/2\sigma^2}}\}$$

Notice that we again ignore the irrelevant constants of g but it is just an M sampler

$$\alpha(X_t, Y) = \min\{1, (Y/X_t)^{n+r-1} e^{-(n\bar{x}+\beta)(Y-X_t)}\}$$

Metropolis–Hastings methods (M–H)

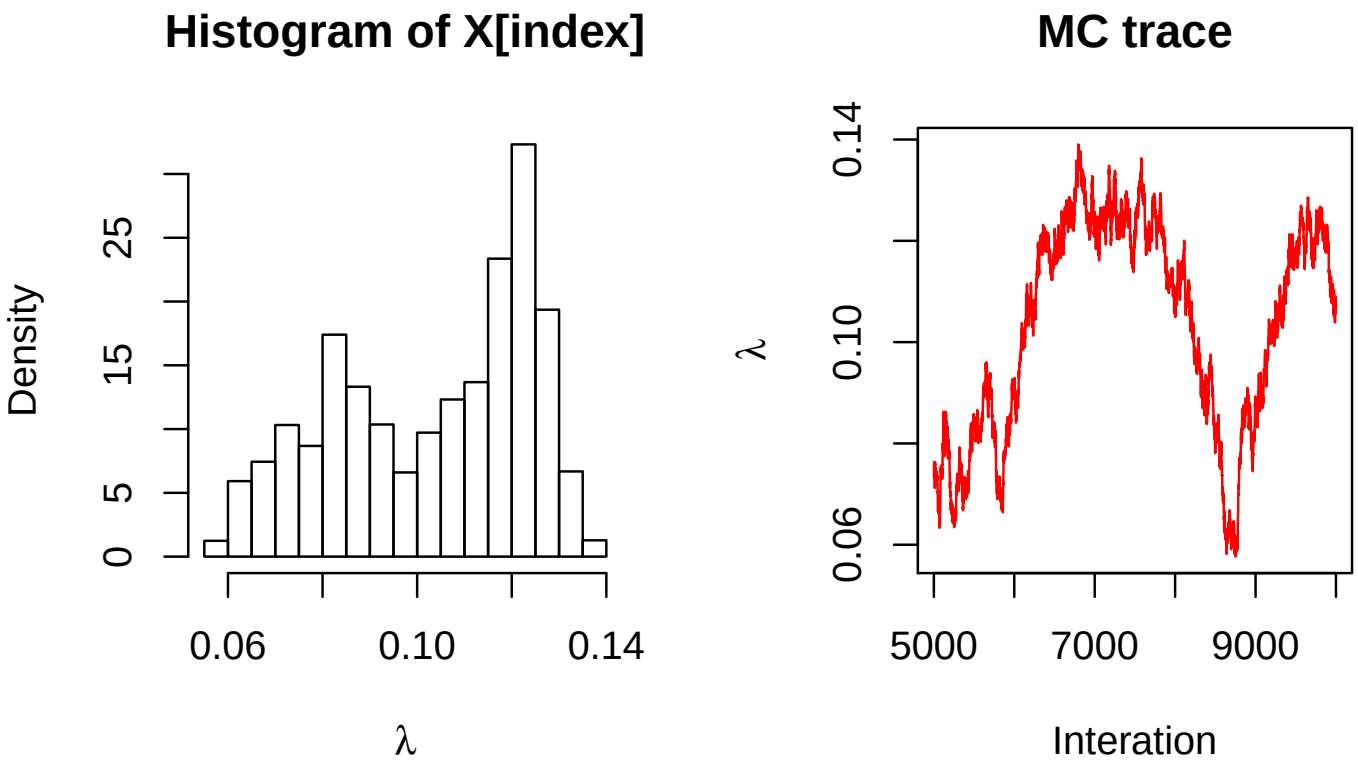
$$X_0 = .1, \sigma = .001$$



acceptance rate=0.997, mean=0.2117715, variance=0.0004963982

Metropolis–Hastings methods (M–H)

same every thing $X_0 = .1, \sigma = .001$



acceptance rate=0.9923, mean=0.1032699, variance=0.0004300882

Metropolis–Hastings methods (M–H)

$$X_0 = .1, \sigma = .1$$

acceptance rate=4e-04, mean=NA, variance=NA

Metropolis–Hastings methods (M–H)

Example: If the sample is given by

$$4.679781, 9.325474, 2.491352, 0.841366, 2.982121$$

(FYI: I used $rexp(5, .373)$)
Test

$$H_0 : \lambda \leq .373 \; vs \; H_a : \lambda > .373$$

Using the Bayes factor

$$B^f_{0,1}(x) = \frac{Pr(\lambda \leq .373|x)}{Pr(\lambda > .373|x)} \frac{\int_{\lambda > .373} f_\lambda(\lambda) d\lambda}{\int_{\lambda \leq .373} f_\lambda(\lambda) d\lambda}$$

Metropolis–Hastings methods (M–H)

Let us use a prior symmetric about .373 like $N(.373, 10)$ so that

$$\int_{\lambda > .373} f_\lambda(\lambda) d\lambda = \int_{\lambda \leq .373} f_\lambda(\lambda) d\lambda$$

and

$$B^f_{0,1}(x) = \frac{Pr(\lambda \leq .373|x)}{Pr(\lambda > .373|x)} = \frac{1}{Pr(\lambda > .373|x)} - 1$$

Metropolis–Hastings methods (M–H)

$$f_{\lambda|x_1,\ldots,x_n}(\lambda) \propto \lambda^n e^{-n\bar{x}\lambda} e^{-\frac{1}{2}(\frac{\lambda-.373}{10})^2}$$

By implementing M–H algorithm using a proposal distribution of $lognormal(X_t, \sigma^2)$

$$\alpha(X_t, Y) = \min\{1, \frac{Y^n e^{-n\bar{x}Y} e^{-\frac{1}{2}(\frac{Y-.373}{10})^2} * (1/X_t) e^{-(\log X_t - Y)^2/2\sigma^2}}{X_t^n e^{-n\bar{x}X_t} e^{-\frac{1}{2}(\frac{X_t-.373}{10})^2} * (1/Y) e^{-(\log Y - X_t)^2/2\sigma^2}}\}$$

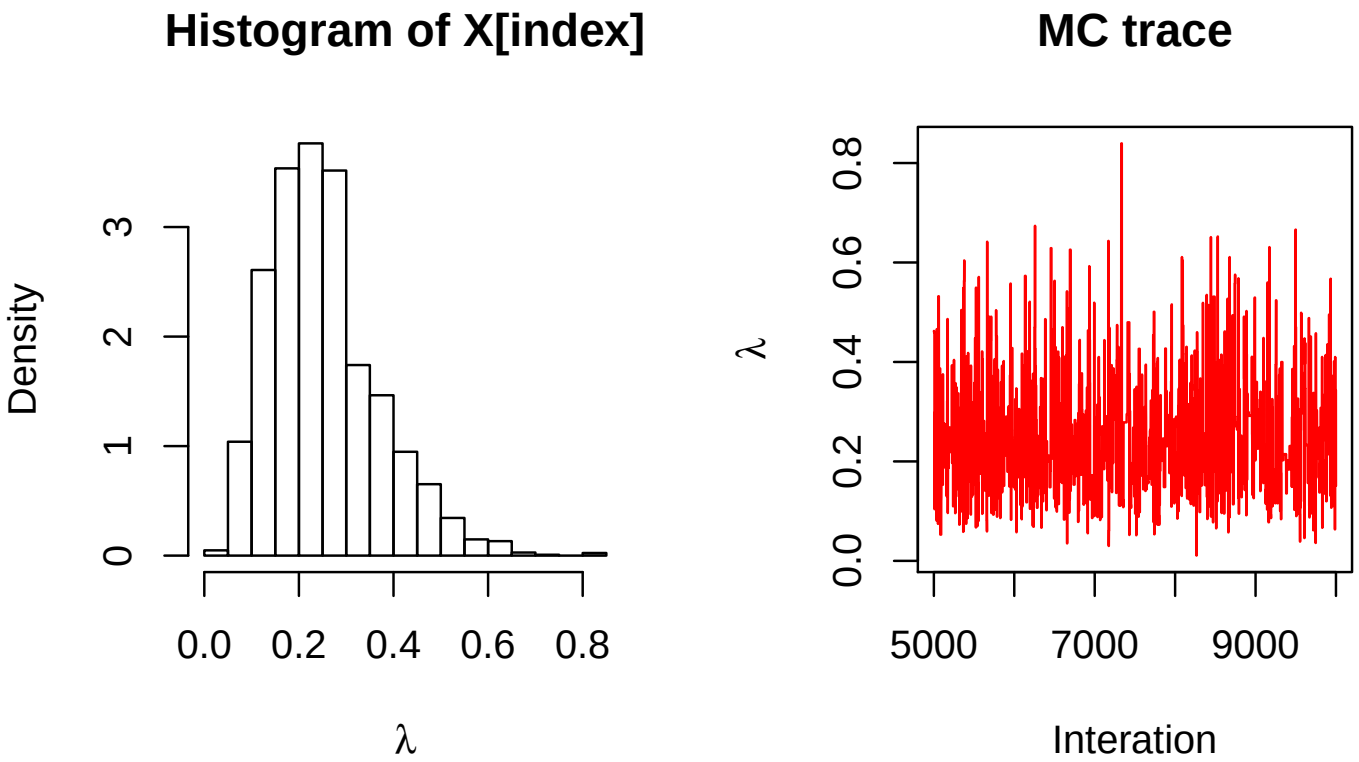
which is

$$\alpha(X_t, Y) = \min\{1, (Y/X_t)^n e^{-n\bar{x}(Y-X_t)}$$

$$e^{-[(Y-.373)^2-(X_t-.373)^2]/200} e^{-[(\log X_t - Y)^2-(\log Y - X_t)^2]/2\sigma^2}\}$$

Metropolis–Hastings methods (M–H)

$$X_0 = .1, \sigma = 2$$



acceptance rate=0.1988, mean=0.252539, variance=0.01329633

Metropolis–Hastings methods (M–H)

```
pro<-as.integer(X[index]>.373)
mean(pro)
[1] 0.1462
```

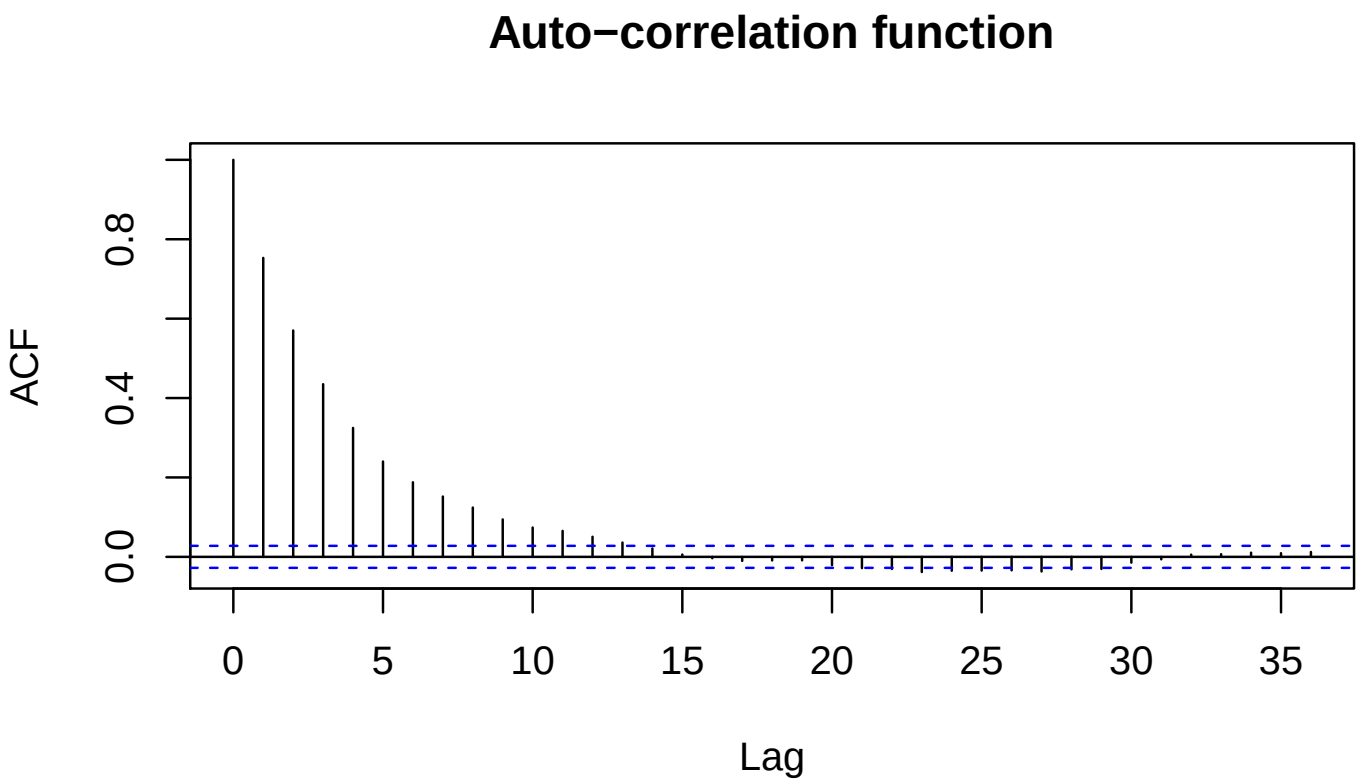
$$B^f_{0,1}(x) = \frac{Pr(\lambda \leq .373|x)}{Pr(\lambda > .373|x)} = \frac{1}{0.1462} - 1 = 5.839945$$

We don't reject H_0.

Metropolis–Hastings methods (M–H)

The auto-correlation with different lags ℓ of the retained states is the correlation between $X_1, \ldots, X_{n-\ell}$ and $X_{\ell+1}, \ldots, X_n$

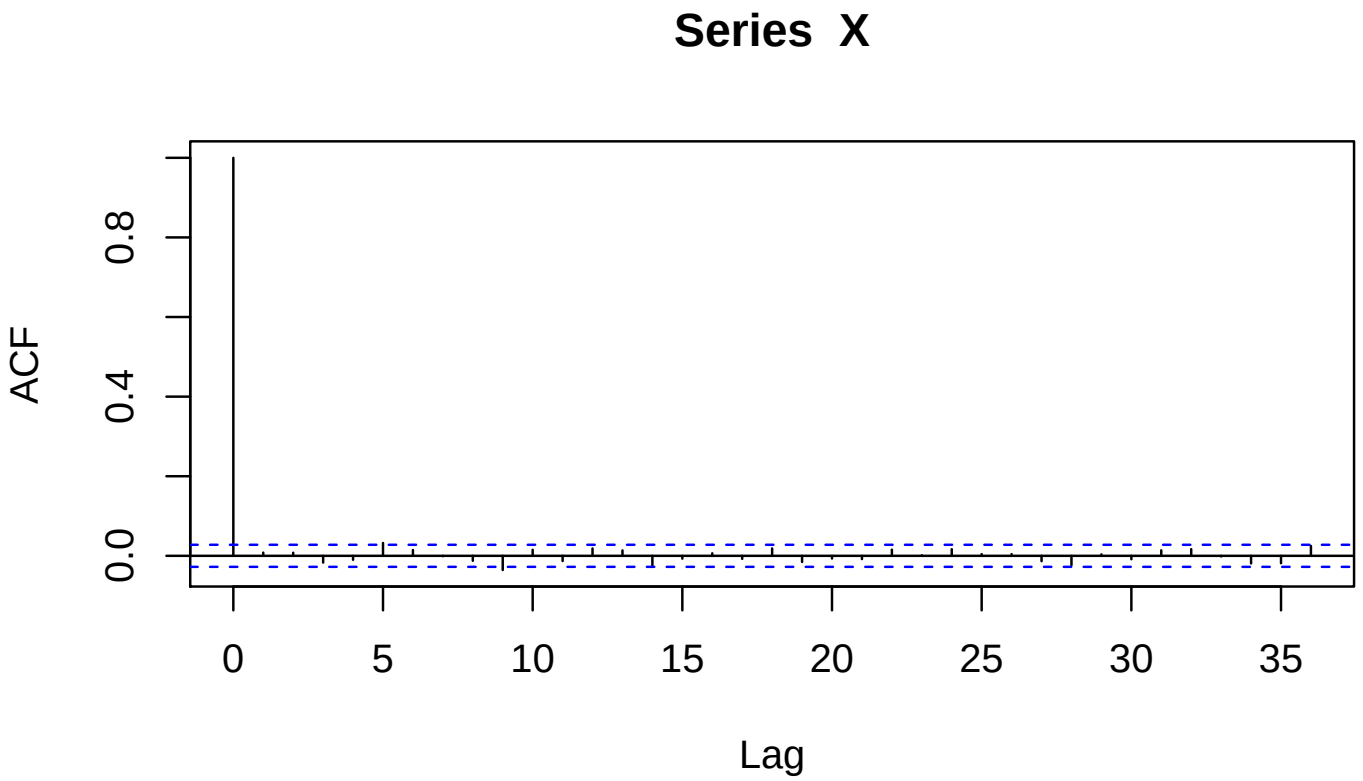
```
acf(X[index],main="Auto-correlation function")
```



Metropolis–Hastings methods (M–H)

The auto-correlation of the thinned states by lag 15 is (which might require an increase of the total number of runs by a multiple of 15)

```
acf(X[index],main="Auto-correlation function")
```



Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Convergence diagnostics are either using

- Single-chain diagnostics: Geweke test
- Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Single-chain diagnostics: Geweke test

- Let m be the number of retained states after the burn-in period of length b.
- Split the retained part of the MC into $m_1 = 10\%m$ (first 10%) and $m_2 = 50\%m$ (last 50%).
- Then calculate

$$\bar{Y}_1 = \frac{1}{m_1} \sum_{i=b+1}^{b+m_1} \Delta_i \text{ and } \bar{Y}_2 = \frac{1}{m_1} \sum_{i=b+m+1-m_2}^{b+m} \Delta_i$$

- Δ_i is any outcome of the MC like $-2\ell(x|\theta^{(i)})$, or $\log f(\theta^{(i)}|x)$

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Single-chain diagnostics: Geweke test

- As m goes to infinity

$$\frac{\bar{Y}_1 - \bar{Y}_2}{\sqrt{\hat{\mathbf{V}}(\bar{Y}_1) + \hat{\mathbf{V}}(\bar{Y}_1)}} \rightarrow N(0,1) \text{ in distribution}$$

- But since the chain data points are dependent then it requires different methods to estimate the variances. Then we can do a two population means test to see if the two parts are the same.

- It can be used to find the burn-in period but doesn't mean **well-mixing** (the chain has covered the support of the target distribution – it might be stuck in sub-region)

- It is available through a library called `CODA` using `geweke.diag`

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Single-chain diagnostics: Geweke test

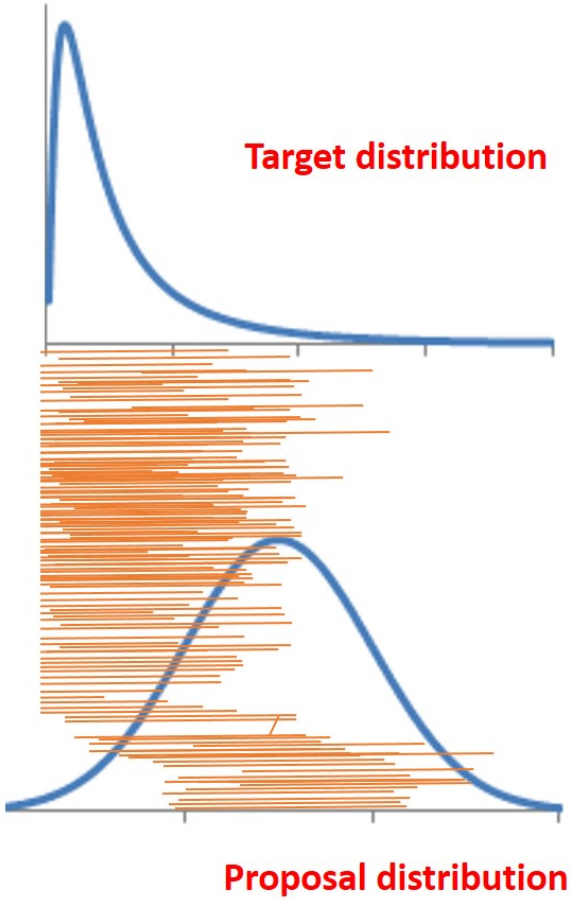
Example: In the previous example after thinning and discarding the burn-in period

```
library(coda)
pnorm(abs(geweke.diag(mcmc(X))$z),lower.tail=FALSE)*2
0.952771
```

We cannot reject that it converged; but cannot guarantee well-mixing.

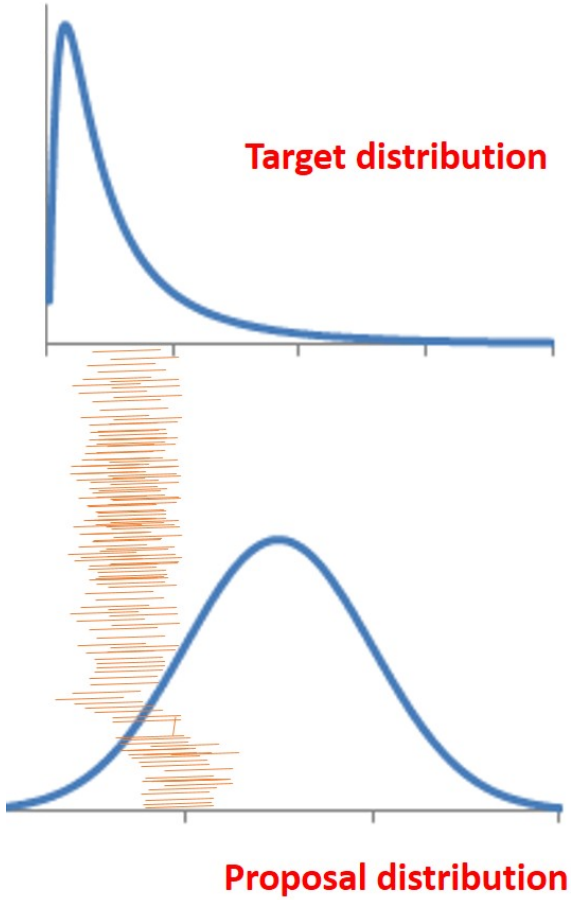
Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Single-chain diagnostics: Geweke test



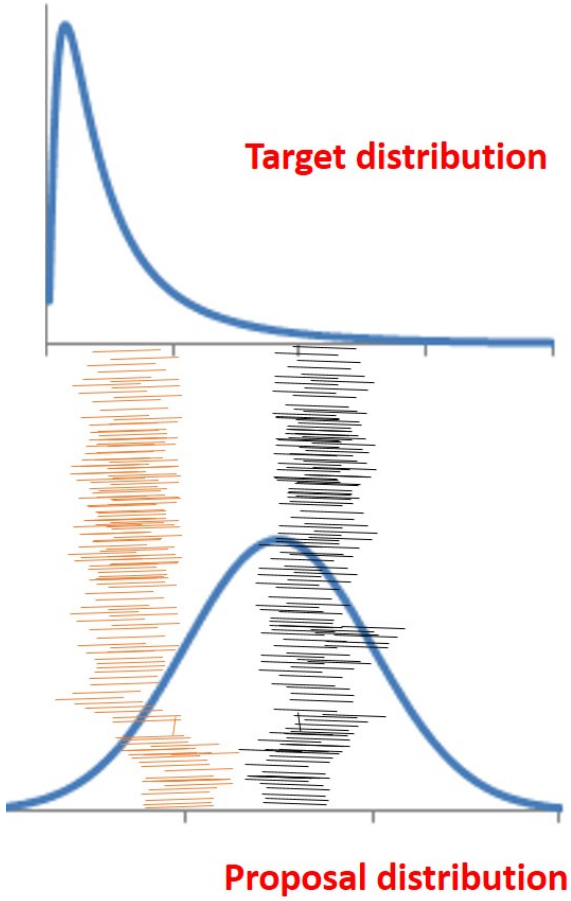
Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Single-chain diagnostics: Geweke test



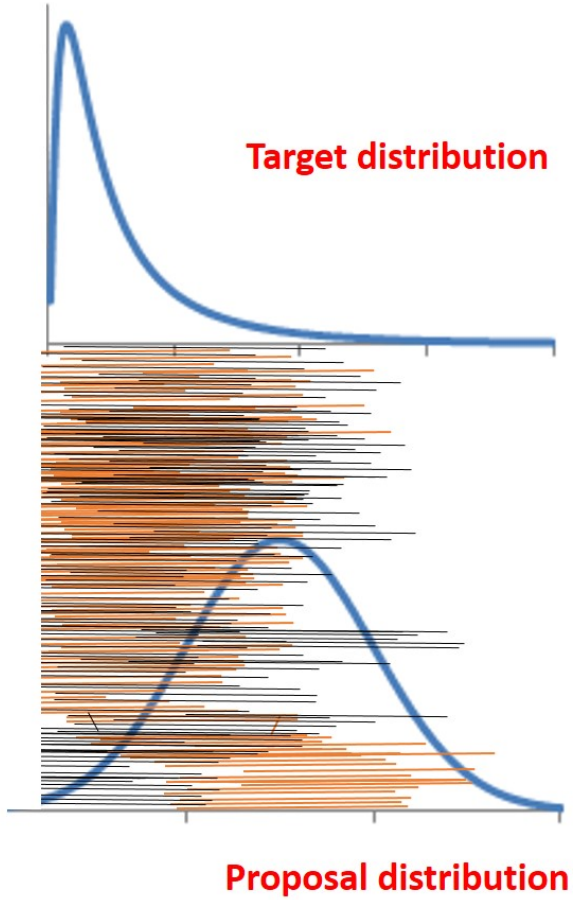
Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics



Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics



Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

- BGR resembles ANOVA in finding the the ratio of between-sample and within-sample mean squared errors
- If we have k chains starting from different/dispresed initial states and are of length m of $\theta^{(i,j)}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, m$
- Make new chains of

$$\Delta_{ij} = \Delta(\theta^{(i,1)}, \ldots, \theta^{(i,j)})$$

for $i = 1, \ldots, k$ and $j = 1, \ldots, m$

- The per-chain mean is $\bar{\Delta}_{i\cdot} = \frac{1}{m} \sum_{j=1}^{m} \Delta_{ij}$ and the overall (grand) mean is $\bar{\Delta}_{\cdot\cdot} = \frac{1}{k} \sum_{i=1}^{k} \Delta_{i\cdot}$

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

- The estimate of between-chain variance is

$$\hat{\sigma}_B^2 = \frac{m}{k-1} \sum_{i=1}^{k} (\bar{\Delta}_{i\cdot} - \bar{\Delta}_{\cdot\cdot})^2$$

and the estimate of within-chain variance (pooled-estimate of variance)

$$\hat{\sigma}_W^2 = \frac{1}{k(m-1)} \sum_{i=1}^{k} \sum_{j=1}^{m} (\Delta_{ij} - \bar{\Delta}_{i\cdot})^2$$

which gives a lower bound for $\mathbf{V}(\Delta)$ (the posterior variance of Δ) if the chain didn't mix well

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

- The marginal posterior variance of Δ is estimated to be

$$\hat{\mathbf{V}}(\Delta) = \frac{m-1}{m}\hat{\sigma}_W^2 + \frac{1}{m}\hat{\sigma}_B^2$$

which gives an upper bound for $\mathbf{V}(\Delta)$

- The BGR potential scale reduction

$$\hat{R} = \sqrt{\frac{\hat{\mathbf{V}}(\Delta)}{\hat{\sigma}_W^2}} = \sqrt{\frac{m-1}{m} + \frac{1}{m}\frac{\hat{\sigma}_B^2}{\hat{\sigma}_W^2}}$$

which must be close to one upon convergence (practically 1.1 or 1.2)

- It is available through a library called `CODA` using `gelman.diag`

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

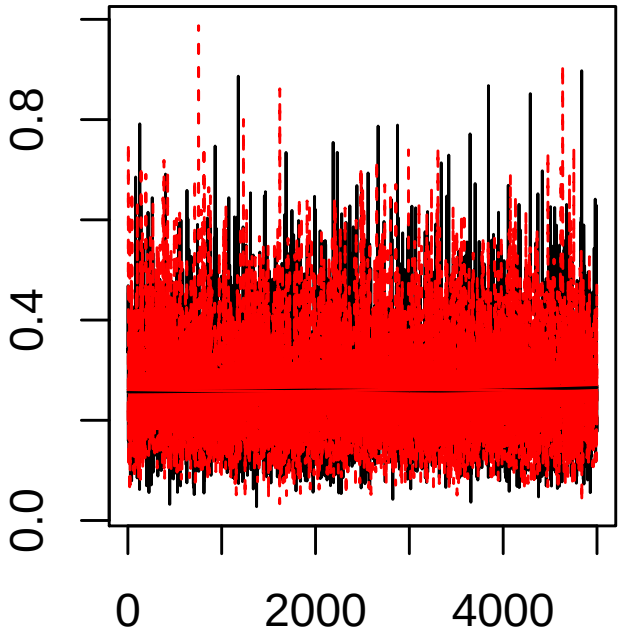
Example: In the previous example after thinning and discarding the burn-in period

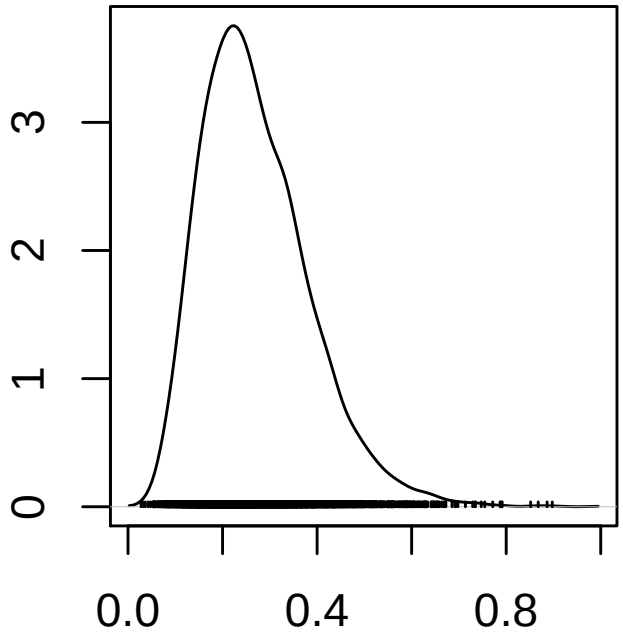
```
library(coda)
X1 = chain(initial=.1)
X2 = chain(initial=2.5)
X1X2 = mcmc.list(mcmc(X1), mcmc(X2))
```

Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

```
plot(X1X2)
```

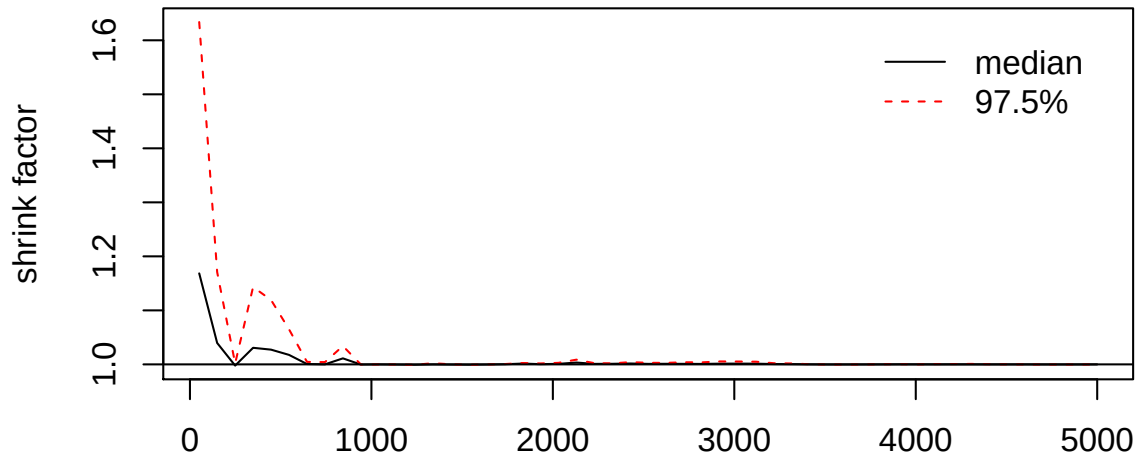




Monitoring Convergence: Convergence Diagnostic and Output Analysis (CODA)

Multi-chain diagnostics: Brooks–Gelman–Rubin (BGR) test

```
gelman.diag(X1X2)
Potential scale reduction factors:
     Point est.  Upper C.I.
[1,]          1          1
gelman.plot(X1X2)
```



Multiple–Block M–H Algorithm (M–B M–H)

Multiple–Block M–H Algorithm (M–B M–H)

- Split θ into p blocks of vectors of parameters $\theta_1, \theta_2, \ldots, \theta_p$.
- M–B M–H requires full conditional distributions

$$f(\theta_k|\theta_{(-k)})$$

for $k = 1, 2, \ldots, p$ where

$$\theta_{(-k)} = (\theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_p)$$

- M–B M–H requires proposal for each block, $g_k(\theta_k|\theta)$ for $k = 1, 2, \ldots, p$

Multiple–Block M–H Algorithm (M–B M–H)

Multiple–Block M–H Algorithm:

1. Choose $\theta^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \ldots, \theta_p^{(0)})$
2. Repeat for $j = 0, \ldots, m-1$

(>) Repeat for $k = 1, 2, \ldots, p$

(i) Generate a random number θ_k' from $g_k(\cdot|\theta_k^{(j)}, \theta_{(-k)})$

(ii) Generate a uniform random number U from $unif(0,1)$

(iii) Calculate $\alpha_k(\theta_k^{(j)}, \theta_k'|\theta_{(-k)}) = \min\{1, \dfrac{f(\theta_k'|\theta_{(-k)})g_k(\theta_k^{(j)}|\theta_k', \theta_{(-k)})}{f(\theta_k^{(j)}|\theta_{(-k)})g_k(\theta_k'|\theta_k^{(j)}, \theta_{(-k)})}\}$

(iv) $\theta_k^{(j+1)} = \theta_k'$ if $U \leq \alpha_k$ and $\theta_k^{(j+1)} = \theta_k^{(j)}$ if not.

:) Return $\theta^{(j+1)} = (\theta_1^{(j+1)}, \theta_2^{(j+1)}, \ldots, \theta_p^{(j+1)})$

Gibbs Sampler

Gibbs Sampler

- GS is a type of M–H algorithm; good for missing values and latent variables
- All proposals are accepted in contrast to M–H algorithm
- It requires full conditionals

$$f(\theta_k|\theta_{(-k)})$$

for $k = 1, 2, \ldots, p$ where

$$\theta_{(-k)} = (\theta_1, \ldots, \theta_{k-1}, \theta_{k+1}, \ldots, \theta_p)$$

Gibbs Sampler

Gibbs sampler algorithm: To have a sample from the target distribution f given full conditionals

1. Choose $\theta^{(0)} = (\theta_1^{(0)}, \theta_2^{(0)}, \theta_3^{(0)}, \ldots, \theta_p^{(0)})$
2. Repeat for $j = 0, \ldots, m-1$

(1) Generate $\theta_1^{(j+1)}$ from $f(\theta_1 | \theta_2^{(j)}, \theta_3^{(j)}, \theta_4^{(j)}, \ldots, \theta_p^{(j)})$

(2) Generate $\theta_2^{(j+1)}$ from $f(\theta_2 | \theta_1^{(j+1)}, \theta_3^{(j)}, \theta_4^{(j)}, \ldots, \theta_p^{(j)})$

(3) Generate $\theta_3^{(j+1)}$ from $f(\theta_3 | \theta_1^{(j+1)}, \theta_2^{(j+1)}, \theta_4^{(j)}, \ldots, \theta_p^{(j)})$

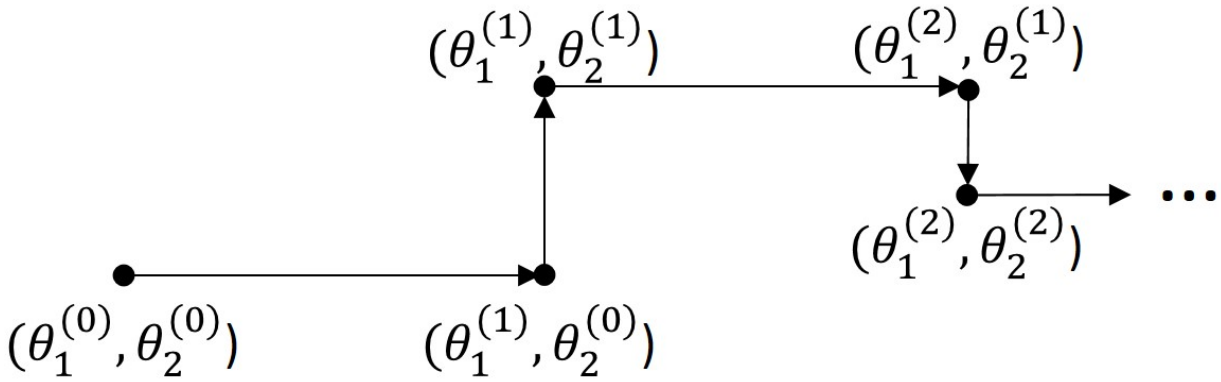
$\vdots$

(k) Generate $\theta_p^{(j+1)}$ from $f(\theta_p | \theta_1^{(j+1)}, \theta_2^{(j+1)}, \theta_3^{(j+1)}, \ldots, \theta_{p-1}^{(j+1)})$

:) Return $\theta^{(j+1)} = (\theta_1^{(j+1)}, \theta_2^{(j+1)}, \theta_3^{(j+1)}, \ldots, \theta_p^{(j+1)})$

Gibbs Sampler

In case of two parameters (θ_1, θ_2)



Gibbs Sampler

The transition kernel is

$$P(\theta, \theta') = \prod_{j=1}^{p} f(\theta_j | \theta'_1, \theta'_2, \theta'_3, \ldots, \theta'_{j-1}, \theta_{j+1}, \ldots, \theta_p)$$

which is a collection of movements parallel to each axis giving one step for the whole MC

WTS that $f(\theta_1, \ldots, \theta_p)$ is the stationary distribution (invariant law):

Gibbs Sampler

When $k = 2$

$$\begin{aligned}
\int\int P(\theta,\theta')f(\theta_1,\theta_2)d\theta_1 d\theta_2 &= \int\int f(\theta_1'|\theta_2)f(\theta_2'|\theta_1')f(\theta_1,\theta_2)d\theta_1 d\theta_2 \\
&= f(\theta_2'|\theta_1')\int\int f(\theta_1'|\theta_2)f(\theta_1,\theta_2)d\theta_1 d\theta_2 \\
&= f(\theta_2'|\theta_1')\int f(\theta_1'|\theta_2)(\int f(\theta_1,\theta_2)d\theta_1)d\theta_2 \\
&= f(\theta_2'|\theta_1')\int f(\theta_1'|\theta_2)f(\theta_2)d\theta_2 \\
&= f(\theta_2'|\theta_1')f(\theta_1') \\
&= f(\theta_1',\theta_2')
\end{aligned}$$

Gibbs Sampler

Example 1: Let $x_1, x_2, \ldots, x_n$ be an observed data of completion time at checking out at a grocery store with two cashiers and no waiting lines. They are modeled by a mixture of two exponential distributions with rates λ_1 and λ_2 and pdfs $f(x|\lambda_1)$ and $f(x|\lambda_2)$.

It is latent to the experimenter, by which cashier, the customers have been serviced.

That corresponds to latent variable $z_1, \ldots, z_n$ for which cashier was selected, encoded as $z_i = 1$ if cashier 1 is selected and $z_i = 0$ if cashier 2 is selected.

Gibbs Sampler

The complete likelihood function can be written as (compare to set 6)

$$\begin{aligned} L(p,\lambda_1,\lambda_2|x,z) &= \prod_{i=1}^{n} (p\cdot f(x_i|\lambda_1))^{z_i} ((1-p)\cdot f(x_i|\lambda_2))^{1-z_i} \\ &= p^{n_1}(1-p)^{n_2}\lambda_1^{n_1} e^{-\lambda_1 n_1 \bar{x}_1} \lambda_2^{n_2} e^{-\lambda_2 n_2 \bar{x}_2} \end{aligned}$$

where $n_1 = \sum_{i=1}^{n} z_i$, $n_2 = n - n_1$, $\bar{x}_1 = \frac{\sum_{i=1}^{n} z_i x_i}{n_1}$, $\bar{x}_2 = \frac{\sum_{i=1}^{n}(1-z_i)x_i}{n_2}$.

Gibbs Sampler

Using priors ...

- $p \sim beta(a, b)$.

  Use the least informative hyper-parameters $a = b = 1$. We can also use hyper-priors like $gamma(.001, .001)$.

- $\lambda_i \sim gamma(\alpha_i, \beta_i)$, for $i = 1, 2$.

  Use $\alpha_i = \beta_i = .001$.

... the posterior is

$$f(p, \lambda_1, \lambda_2 | x, z) \propto p^{n_1+a-1}(1-p)^{n_2+b-1}.$$
$$\lambda_1^{n_1+\alpha_1-1} e^{-\lambda_1(n_1\bar{x}_1+\beta_1)} \cdot \lambda_2^{n_2+\alpha_2-1} e^{-\lambda_2(n_2\bar{x}_2+\beta_2)}$$

Gibbs Sampler

Using the posterior

$$f(p, \lambda_1, \lambda_2 | x, z) \propto p^{n_1+a-1}(1-p)^{n_2+b-1}.$$

$$\lambda_1^{n_1+\alpha_1-1} e^{-\lambda_1(n_1\bar{x}_1+\beta_1)} \cdot \lambda_2^{n_2+\alpha_2-1} e^{-\lambda_2(n_2\bar{x}_2+\beta_2)}$$

The full-conditional distributions are

- $f(p|\lambda_1, \lambda_2, x, z) \propto p^{n_1+a-1}(1-p)^{n_2+b-1}$
- $f(\lambda_1|p, \lambda_2, x, z) \propto \lambda_1^{n_1+\alpha_1-1} e^{-\lambda_1(n_1\bar{x}_1+\beta_1)}$
- $f(\lambda_2|p, \lambda_1, x, z) \propto \lambda_2^{n_2+\alpha_2-1} e^{-\lambda_2(n_2\bar{x}_2+\beta_2)}$

where $n_1 = \sum_{i=1}^{n} z_i$, $n_2 = n - n_1$, $\bar{x}_1 = \frac{\sum_{i=1}^{n} z_i x_i}{n_1}$, $\bar{x}_2 = \frac{\sum_{i=1}^{n}(1-z_i)x_i}{n_2}$.

Gibbs Sampler

To find n_1, n_2, $\bar{x}_1$, and $\bar{x}_2$, we can use

$$\rho_i = P(Z_i = 1 | X = x_i, p, \lambda_1, \lambda_2) = \frac{p \cdot f(x_i|\lambda_1)}{p \cdot f(x_i|\lambda_1) + (1-p) \cdot f(x_i|\lambda_2)}$$

due to Bayes' theorem.

Gibbs Sampler

Algorithm:

1. Initialize $p^{(0)}$ and $\lambda_1^{(0)}$ and $\lambda_2^{(0)}$

> for $j = 0 \ldots, m-1$

(i) For each $i = 1, \ldots, n$: Generate z_i using the full-conditional

$$[z_i^{(j+1)} | X = x_i, p^{(j)}, \lambda_1^{(j)}, \lambda_2^{(j)}] \sim Bernoulli(\rho_i^{(j)})$$

where

$$\rho_i^{(j)} = \frac{p^{(j)} \cdot f(x_i|\lambda_1^{(j)})}{p^{(j)} \cdot f(x_i|\lambda_1^{(j)}) + (1-p^{(j)}) \cdot f(x_i|\lambda_2^{(j)})}$$

(ii) Update $n_1^{(j+1)} = \sum_{i=1}^n z_i^{(j+1)}$, $n_2^{(j+1)} = n - n_1^{(j+1)}$,
$\bar{x}_1^{(j+1)} = \frac{\sum_{i=1}^n z_i^{(j+1)} x_i}{n_1^{(j+1)}}$, $\bar{x}_2^{(j+1)} = \frac{\sum_{i=1}^n (1-z_i^{(j+1)}) x_i}{n_2^{(j+1)}}$

Gibbs Sampler

Algorithm: (cont'd)

(iii) Generate p using the full-conditional

$$[p^{(j+1)}|X, z^{(j+1)}, \lambda_1^{(j)}, \lambda_2^{(j)}] \sim beta(n_1^{(j+1)} + a, n_2^{(j+1)} + b)$$

(iv) For each $k = 1, 2$: Generate λ_k using the full-conditional

$$[\lambda_k^{(j+1)}|X, z^{(j+1)}, p^{(j+1)}] \sim gamma(n_k^{(j+1)} + \alpha_k, n_k^{(j+1)}\bar{x}_k^{(j+1)} + \beta_k)$$

:) Return $(p^{(j+1)}, \lambda_1^{(j+1)}, \lambda_2^{(j+1)})$.

Gibbs Sampler

```
n<-1000;p<-.2;lambda1<-.3;lambda2<-.5
lambda<-c(lambda1,lambda2)
K<-sample(1:2,n,prob=c(p,1-p),rep=T)
x<-rexp(n,rate=lambda[K])

rho<-function(p,lambda1,lambda2,x)
{p*dexp(x,lambda1)
/(p*dexp(x,lambda1)+(1-p)*dexp(x,lambda2))}
a<-b<-1;alpha1<-beta1<-alpha2<-beta2<-.0001;
```

Gibbs Sampler

```
chain<-function(initial,thining=100,m=10000,
burnin=1000)
{ m<-m*thining
 X<-t(c(initial))
 for (i in 2:m){
 z<-rbinom(n,1,rho(X[i-1,1],X[i-1,2],X[i-1,3],x))
 n1<-sum(z);n2<-n-n1;
 x1<-sum(z*x);x2<-sum((1-z)*x)
 p<-rbeta(1,a+n1,b+n2)
 lam1<-rgamma(1,n1+alpha1,x1+beta1)
 lam2<-rgamma(1,n2+alpha2,x2+beta2)
 X<-rbind(X,c(p,lam1,lam2))}
 burnin<-burnin*thining
 return(X[seq((burnin+1),m,thining),])}
```

Gibbs Sampler

```
X<-chain(initial=c(.1,.1,2))
Y<-chain(initial=c(.5,.1,10))
Z<-chain(initial=c(.9,2,.1))
```

Gibbs Sampler

Example 2: Let $x_1, x_2, \ldots, x_n$ be count of Zika carrying mosquitoes found in n regions among N similar traps set up in each region. Estimate the prevalence in each region using a logistic model and accounting for the region-specific variability.

That means, we use the model

$$X_i \sim Binomial(N, p_i)$$

such that

$$q_i := logit(p_i) = \log(\frac{p_i}{1-p_i}) = U_i$$

and

$$U_i \sim N(\mu_i, \frac{1}{\tau})$$

Gibbs Sampler

The likelihood function can be written as

$$L(\{p_i\}, \{\mu_i\}, \tau | x) \propto \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{N - x_i} \cdot \tau^{\frac{1}{2}} e^{-\frac{\tau}{2}(q_i - \mu_i)^2}$$

where $q_i = logit(p_i)$.

Gibbs Sampler

Using priors ...

- $\tau \sim gamma(a, b)$.
  Use the least informative hyper-parameters $a = b = .001$.

- $[\mu_i|\tau] \sim N(\alpha, \frac{\beta}{\tau})$.
  Use $\alpha = 0$ and $\beta = 1$.

... the posterior is

$$f(\{p_i\}, \{\mu_i\}, \tau|x) \propto \left(\prod_{i=1}^{n} p_i^{x_i} (1-p_i)^{N-x_i} \cdot \tau^{\frac{1}{2}} e^{-\frac{\tau}{2}(q_i-\mu_i)^2} \right) \cdot$$

$$\prod_{i=1}^{n} \left(\tau^{\frac{1}{2}} e^{-\frac{\tau}{2\beta}(\mu_i-\alpha)^2} \right) \cdot \tau^{a-1} e^{-b\tau}$$

Gibbs Sampler

Using the posterior

$$f(\{p_i\},\{\mu_i\},\tau|x) \propto \left(\prod_{i=1}^{n} p_i^{x_i}(1-p_i)^{N-x_i}\cdot \tau^{\frac{1}{2}} e^{-\frac{\tau}{2}(q_i-\mu)^2}\right)\cdot$$

$$\prod_{i=1}^{n}\left(\tau^{\frac{1}{2}} e^{-\frac{\tau}{2\beta}(\mu_i-\alpha)^2}\right)\cdot \tau^{a-1}e^{-b\tau}$$

The full-conditional distributions are

- $f(\tau|\{p_i\},\{\mu_i\},x) \propto \tau^{n+a-1} e^{-(b+\frac{1}{2}\sum_{i=1}^{n}[(q_i-\mu_i)^2+\frac{1}{\beta}(\mu_i-\alpha)^2])\tau}$
- $f(\mu_i|\{p_i\},\tau,x) \propto e^{-\frac{1}{2}((q_i-\mu_i)^2+\frac{1}{\beta}(\mu_i-\alpha)^2)\tau} \propto e^{-\frac{1}{2\sigma^2}(\mu_i-\mu_0)^2}$, for each $i=1,\ldots,n$
- $f(p_i|\mu_i,\tau,x) \propto p_i^{x_i}(1-p_i)^{N-x_i} e^{-\frac{\tau}{2}(q_i-\mu_i)^2}$, for each $i=1,\ldots,n$

Then use gamma and normal random numbers to generate the first two. Use acceptance-rejection method to generate the last one with beta proposal distribution.

Gibbs Sampler

Remark in Acceptance-Rejection method: If $f(x) = af_0(x)f_1(x)$ is a target pdf with **unknown** normalization constant a and is given by the product of two functions. If $g(y) = bf_0(y)$ is a well-defined proposal pdf with normalization constant b and $f_1(x) \leq d$ then

$$1 = \int_{-\infty}^{\infty} f(x)dx \leq ad \int_{-\infty}^{\infty} f_0(x)dx = \frac{ad}{b}$$

Thus

$$\frac{f(x)}{g(x)} \leq \frac{ad}{b} := c$$

and so

$$\frac{1}{d} f_1(x) = \frac{f(x)}{cg(x)} \leq 1$$

and the acceptance condition is $U < \frac{1}{d} f_1(Y)$ with Y generated by the proposal $g(y)$. But what if we don't know d?!

Slice Sampler

Slice Sampler

- Proposed by Edwards and Sokal (1988) to improve mixing
- The target distribution $f(\theta)$ in SS is a marginal to a new target
- The new target is made by introducing an arbitrary auxiliary variable ξ with a conditional distribution $f(\xi|\theta)$, but that is easy to simulate
- The new target is then the joint pdf $f(\theta, \xi) = f(\theta) \cdot f(\xi|\theta)$
- Generate a chain ξ and θ using Gibbs sampling (via the full-conditionals $f(\xi|\theta)$ and $f(\theta|\xi)$) but only retain the states of θ

Slice Sampler

Application: Generate random numbers from $f(x) \propto f_0(x)f_1(x)$ where $f_0(x)$ is a well-defined pdf and $f_1(x)$ is a complicated function that we cannot deal with.

Introduce the auxiliary variable U for which

$$[U|X = x] \sim unif(0, f_1(x))$$

Therefore, the new target distribution is

$$\begin{aligned} f_{X,U}(x,u) &= f(x)f(u|x) \\ &\propto f_0(x)f_1(x)\frac{I(0 < u < f_1(x))}{f_1(x)} \\ &\propto f_0(x)I(f_1(x) > u) \end{aligned}$$

Slice Sampler

Then the algorithm goes along the lines:

1. Generate a $u^{(j+1)}$ using $unif(0, f_1(x^{(j)}))$
2. Generate an x using $f_0(x)$ and set $x^{(j+1)} = x$ if $f_1(x) > u^{(j+1)}$ or regenerate another x

Reversible Jump MCMC Algorithm

Reversible Jump MCMC Algorithm

- Reversible Jump MCMC Algorithm is used in model selection among $\{\mathcal{M}_i : i \in I\}$, where I is possibly uncountable index set
- Each $\mathcal{M}_i$ has a parameter space Θ_i with different parameter dimensions $dim(\Theta_i)$
- Jumps between models can be done over models' index via Birth and Death process
- Reversibility is in the jump from index i to index j
- It resembles M–H algorithm in using acceptance probability

Reversible Jump MCMC Algorithm

- If the model $\mathcal{M}_i$ has a vector parameter $\theta_{(i)}$ with length equals to $dim(\Theta_i)$ then augment a new parameter $\imath$ to the model so that $(\imath, \theta_{(\imath)})$ is the parameter vector and the posterior is

$$f(\imath, \theta_{(\imath)}|x)$$

- A jump from $\imath = i$ to $\imath = i'$ through a BDP causes a change in dimension from $dim(\Theta_i)$ to $dim(\Theta_{i'})$

Reversible Jump MCMC Algorithm

- If $dim(\Theta_i) > dim(\Theta_{i'})$, so you do a completion of $\Theta_{i'}$ by augmenting it with a random vector $(\alpha_1, \ldots, \alpha_r) \sim g$ from a known probability distribution g so as that

$$dim(\Theta_{i'}) + r = dim(\Theta_i)$$

- In general, we can add another completion of Θ_i by augmenting it with another random vector $(\beta_1, \ldots, \beta_s) \sim g$ such that

$$dim(\Theta_{i'}) + r = dim(\Theta_i) + s$$

with the possibility that r and/or s be equal to zero

To be continued ...

OpenBUGS

CODA of files produced in OpenBUGS can be done as follows

- Save CODA files as CODAindex.txt, CODAchain1.txt, CODAchain2.txt and store them in one folder, say analysis
- Run

```
library(coda)
data<-read.openbugs(stem="C:/analysis/")
```

- Use the same functions discussed above in monitoring convergence

End of Set 8