

# Introducing Monte Carlo Methods with R

**Christian P. Robert**

Université Paris Dauphine

[xian@ceremade.dauphine.fr](mailto:xian@ceremade.dauphine.fr)

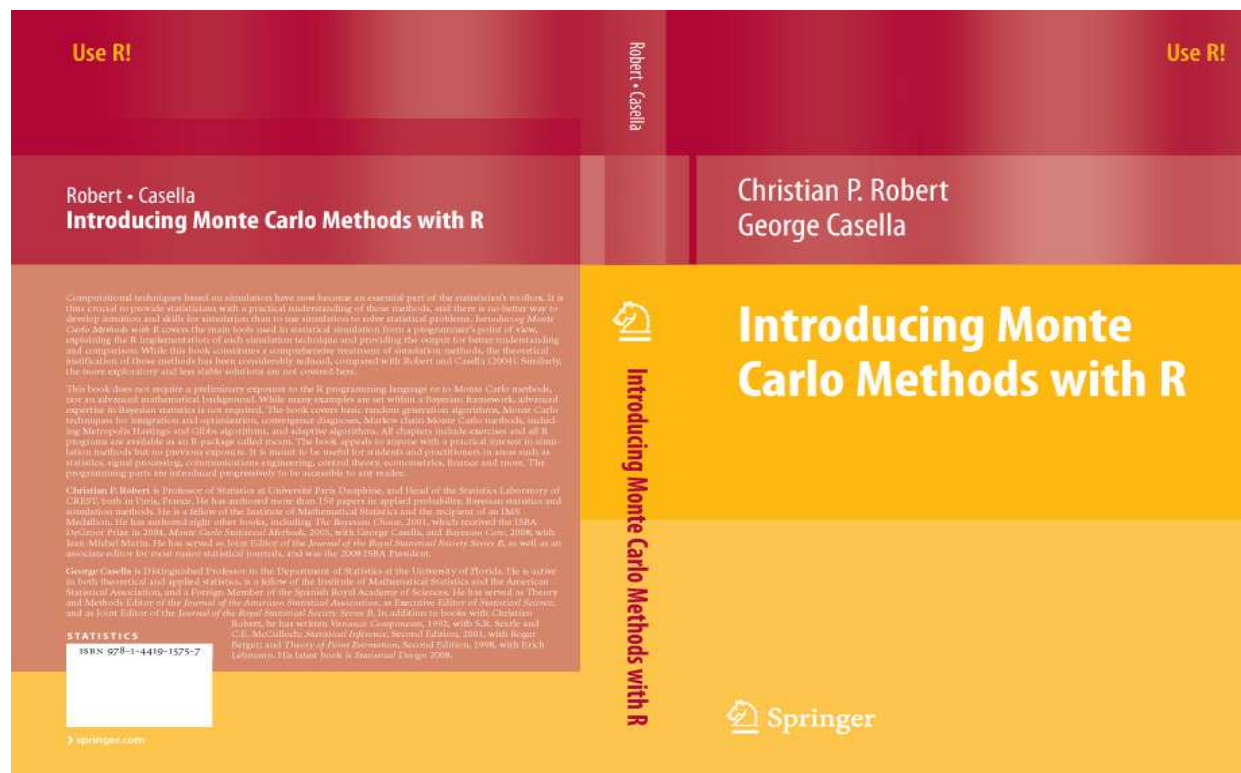
**George Casella**

University of Florida

[casella@ufl.edu](mailto:casella@ufl.edu)

## Based on

- **Introducing Monte Carlo Methods with R**, 2009, Springer-Verlag
- Data and R programs for the course available at <http://www.stat.ufl.edu/casella/IntroMonte/>



## Chapter 1: Basic R Programming

*“You’re missing the big picture,” he told her. “A good album should be more than the sum of its parts.”*

**Ian Rankin**  
*Exit Music*

### **This Chapter**

- ▶ We introduce the programming language R
- ▶ Input and output, data structures, and basic programming commands
- ▶ The material is both crucial and unavoidably sketchy

## Basic R Programming

### Introduction

- ▶ This is a quick introduction to **R**
- ▶ There are entire books devoted to **R**
  - ▷ **R Reference Card**
  - ▷ available at <http://cran.r-project.org/doc/contrib/Short-refcard.pdf>
- ▶ Take Heart!
  - ▷ The syntax of **R** is simple and logical
  - ▷ The best, and in a sense the only, way to learn **R** is through trial-and-error
- ▶ Embedded help commands **help()** and **help.search()**
  - ▷ **help.start()** opens a Web browser linked to the local manual pages

## Basic R Programming

### Why R ?

- ▶ There exist other languages, most (all?) of them faster than **R**, like **Matlab**, and even free, like **C** or **Python**.
- ▶ The language combines a sufficiently high power (for an interpreted language) with a very clear syntax both for statistical computation and graphics.
- ▶ **R** is a flexible language that is *object-oriented* and thus allows the manipulation of complex data structures in a condensed and efficient manner.
- ▶ Its graphical abilities are also remarkable
  - ▷ Possible interfacing with **L<sup>A</sup>T<sub>E</sub>X** using the package **Sweave**.

## Basic R Programming

### Why R ?

- ▶ R offers the additional advantages of being a free and open-source system
  - ▷ There is even an R newsletter, *R-News*
  - ▷ Numerous (free) Web-based tutorials and user's manuals
- ▶ It runs on all platforms: **Mac**, **Windows**, **Linux** and **Unix**
- ▶ R provides a powerful *interface*
  - ▷ Can integrate programs written in other languages
  - ▷ Such as **C**, **C++**, **Fortran**, **Perl**, **Python**, and **Java**.
- ▶ It is increasingly common to see people who develop new methodology simultaneously producing an R package
- ▶ Can interface with **WinBugs**

## Basic R Programming

### Getting started

- ▶ Type 'demo()' for some demos; `demo(image)` and `demo(graphics)`
- ▶ 'help()' for on-line help, or 'help.start()' for an HTML browser interface to help.
- ▶ Type 'q()' to quit R.
- ▶ Additional packages can be loaded via the `library` command, as in
  - > `library(combinat)` # combinatorics utilities
  - > `library(datasets)` # The R Datasets Package
  - ▷ There exist hundreds of packages available on the Web.
  - > `install.package("mcsm")`
- ▶ A `library` call is required each time R is launched

## Basic R Programming

### R objects

- ▶ R distinguishes between several types of *objects*
  - ▷ scalar, vector, matrix, time series, data frames, functions, or graphics.
  - ▷ An R object is mostly characterized by a *mode*
  - ▷ The different modes are
    - **null** (empty object),
    - **logical** (**TRUE** or **FALSE**),
    - **numeric** (such as **3**, **0.14159**, or **2+sqrt(3)**),
    - **complex**, (such as **3-2i** or **complex(1,4,-2)**), and
    - **character** (such as **"Blue"**, **"binomial"**, **"male"**, or **"y=a+bx"**),
- ▶ The R function **str** applied to any R object will show its structure.



## Basic R Programming Interpreted

- ▶ R operates on those types as a regular function would operate on a scalar
- ▶ R is interpreted  $\Rightarrow$  Slow
- ▶ Avoid loops in favor of matrix manipulations

## Basic R Programming – The **vector** class

```
> a=c(5,5.6,1,4,-5)  build the object a containing a numeric vector  
                      of dimension 5 with elements 5, 5.6, 1, 4, -5  
  
> a[1]              display the first element of a  
  
> b=a[2:4]          build the numeric vector b of dimension 3  
                      with elements 5.6, 1, 4  
  
> d=a[c(1,3,5)]      build the numeric vector d of dimension 3  
                      with elements 5, 1, -5  
  
> 2*a               multiply each element of a by 2  
                      and display the result  
  
> b%%3              provides each element of b modulo 3
```

## Basic R Programming

### More **vector** class

- > `e=3/d`      build the numeric vector **e** of dimension 3  
                 and elements  $3/5$ ,  $3$ ,  $-3/5$
- > `log(d*e)`    multiply the vectors **d** and **e** term by term  
                 and transform each term into its natural logarithm
- > `sum(d)`      calculate the sum of **d**
- > `length(d)`   display the length of **d**

## Basic R Programming

### Even more **vector** class

<code>&gt; t(d)</code>	transpose <b>d</b> , the result is a row vector
<code>&gt; t(d)*e</code>	elementwise product between two vectors with identical lengths
<code>&gt; t(d)%*%e</code>	matrix product between two vectors with identical lengths
<code>&gt; g=c(sqrt(2),log(10))</code>	build the numeric vector <b>g</b> of dimension 2 and elements $\sqrt{2}$ , $\log(10)$
<code>&gt; e[d==5]</code>	build the subvector of <b>e</b> that contains the components <b>e[i]</b> such that <b>d[i]=5</b>
<code>&gt; a[-3]</code>	create the subvector of <b>a</b> that contains all components of <b>a</b> but the third.
<code>&gt; is.vector(d)</code>	display the logical expression <b>TRUE</b> if a vector and <b>FALSE</b> else

Basic R Programming  
Comments on the **vector** class

- The ability to apply scalar functions to vectors: [Major Advantage of R](#).

- ▷ `> lgamma(c(3,5,7))`

- ▷ returns the vector with components  $(\log \Gamma(3), \log \Gamma(5), \log \Gamma(7))$ .

- Functions that are specially designed for vectors include

**sample**, **permn**, **order**, **sort**, and **rank**

- ▷ All manipulate the order in which the components of the vector occur.

- ▷ **permn** is part of the **combinat** library

- The components of a vector can also be identified by names.

- ▷ For a vector **x**, **names(x)** is a vector of characters of the same length as **x**

## Basic R Programming

### The `matrix`, `array`, and `factor` classes

- ▶ The `matrix` class provides the R representation of matrices.
- ▶ A typical entry is
  - > `x=matrix(vec,nrow=n,ncol=p)`
    - ▷ Creates an  $n \times p$  matrix whose elements are of the dimension  $np$  vector `vec`
- ▶ Some manipulations on matrices
  - ▷ The standard matrix product is denoted by `%*%`,
  - ▷ while `*` represents the term-by-term product.
  - ▷ `diag` gives the vector of the diagonal elements of a matrix
  - ▷ `crossprod` replaces the product `t(x)%*%y` on either vectors or matrices
    - ▷ `crossprod(x,y)` more efficient
  - ▷ `apply` is easy to use for functions operating on matrices by row or column

## Basic R Programming

### Some **matrix** commands

<code>&gt; x1=matrix(1:20,nrow=5)</code>	build the numeric matrix <b>x1</b> of dimension $5 \times 4$ with first row 1, 6, 11, 16
<code>&gt; x2=matrix(1:20,nrow=5,byrow=T)</code>	build the numeric matrix <b>x2</b> of dimension $5 \times 4$ with first row 1, 2, 3, 4
<code>&gt; a=x1%*%t(x2)</code>	matrix product
<code>&gt; c=x1*x2</code>	term-by-term product between <b>x1</b> and <b>x2</b>
<code>&gt; dim(x1)</code>	display the dimensions of <b>x1</b>
<code>&gt; b[,2]</code>	select the second column of <b>b</b>
<code>&gt; b[c(3,4),]</code>	select the third and fourth rows of <b>b</b>
<code>&gt; b[-2,]</code>	delete the second row of <b>b</b>
<code>&gt; rbind(x1,x2)</code>	vertical merging of <b>x1</b> and <b>x2</b> <code>rbind(*)rbind</code>
<code>&gt; cbind(x1,x2)</code>	horizontal merging of <b>x1</b> and <b>x2</b> <code>cbind(*)cbind</code>
<code>&gt; apply(x1,1,sum)</code>	calculate the sum of each row of <b>x1</b>
<code>&gt; as.matrix(1:10)</code>	turn the vector <b>1:10</b> into a $10 \times 1$ matrix

► Lots of other commands that we will see throughout the course

Basic R Programming  
The `list` and `data.frame` classes  
The Last One

- ▶ A **list** is a collection of arbitrary objects known as its *components*
    - > `li=list(num=1:5,y="color",a=T)` create a list with three arguments
  - ▶ The last class we briefly mention is the **data frame**
    - ▷ A list whose elements are possibly made of differing modes and attributes
    - ▷ But have the same length
- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>&gt; <code>v1=sample(1:12,30,rep=T)</code></li><li>&gt; <code>v2=sample(LETTERS[1:10],30,rep=T)</code></li><li>&gt; <code>v3=runif(30)</code></li><li>&gt; <code>v4=rnorm(30)</code></li><li>&gt; <code>xx=data.frame(v1,v2,v3,v4)</code></li></ul> | <ul style="list-style-type: none"><li>simulate 30 independent uniform <math>\{1, 2, \dots, 12\}</math></li><li>simulate 30 independent uniform <math>\{a, b, \dots, j\}</math></li><li>simulate 30 independent uniform <math>[0, 1]</math></li><li>simulate 30 independent standard normals</li><li>create a data frame</li></ul> |
|---|---|
- ▶ R **code**



## Probability distributions in R

- R , or the web, has about all probability distributions
- Prefixes: **p**, **d**, **q**, **r**

Distribution	Core	Parameters	Default Values
Beta	beta	shape1, shape2	
Binomial	binom	size, prob	
Cauchy	cauchy	location, scale	0, 1
Chi-square	chisq	df	
Exponential	exp	1/mean	1
F	f	df1, df2	
Gamma	gamma	shape, 1/scale	NA, 1
Geometric	geom	prob	
Hypergeometric	hyper	m, n, k	
Log-normal	lnorm	mean, sd	0, 1
Logistic	logis	location, scale	0, 1
Normal	norm	mean, sd	0, 1
Poisson	pois	lambda	
Student	t	df	
Uniform	unif	min, max	0, 1
Weibull	weibull	shape	

## Basic and not-so-basic statistics

### $t$ -test

#### ► Testing equality of two means

```
> x=rnorm(25) #produces a N(0,1) sample of size 25  
> t.test(x)
```

#### One Sample $t$ -test

```
data:  x  
t = -0.8168, df = 24, p-value = 0.4220  
alternative hypothesis: true mean is not equal to 0  
95 percent confidence interval:  
 -0.4915103  0.2127705  
sample estimates:  
 mean of x  
-0.1393699
```

## Basic and not-so-basic statistics

### Correlation

#### ► Correlation

```
> attach(faithful) #resident dataset  
> cor.test(faithful[,1],faithful[,2])
```

Pearson's product-moment correlation

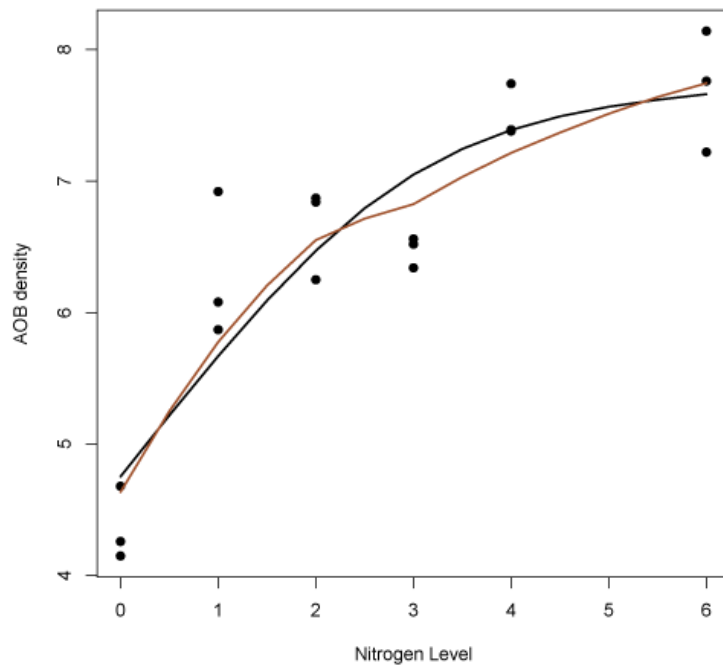
```
data: faithful[, 1] and faithful[, 2]  
t = 34.089, df = 270, p-value < 2.2e-16  
alternative hypothesis: true correlation is not equal to 0  
95 percent confidence interval:  
 0.8756964 0.9210652  
sample estimates:  
      cor  
0.9008112
```

#### ► R [code](#)

## Basic and not-so-basic statistics

### Splines

- ▶ Nonparametric regression with **loess** function or using *natural splines*
- ▶ Relationship between nitrogen level in soil and abundance of a bacteria **AOB**



- ▶ Natural spline fit (*dark*)
  - ▷ With **ns=2** (linear model)
- ▶ Loess fit (*brown*) with **span=1.25**
- ▶ R **code**

## Basic and not-so-basic statistics Generalized Linear Models

- Fitting a binomial (logistic) glm to the probability of suffering from diabetes for a woman within the Pima Indian population

```
> glm(formula = type ~ bmi + age, family = "binomial", data = Pima.tr)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7935	-0.8368	-0.5033	1.0211	2.2531

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )
(Intercept)	-6.49870	1.17459	-5.533	3.15e-08 ***
bmi	0.10519	0.02956	3.558	0.000373 ***
age	0.07104	0.01538	4.620	3.84e-06 ***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 256.41 on 199 degrees of freedom

Residual deviance: 215.93 on 197 degrees of freedom

AIC: 221.93

Number of Fisher Scoring iterations: 4

Basic and not-so-basic statistics  
Generalized Linear Models – Comments

- ▶ Concluding with the significance both of the body mass index **bmi** and the age
- ▶ Other generalized linear models can be defined by using a different **family** value
  - > `glm(y ~x, family=quasi(var="mu^2", link="log"))`
  - ▷ Quasi-Likelihood also
- ▶ Many many other procedures
  - ▷ Time series, anova,...
- ▶ One last one

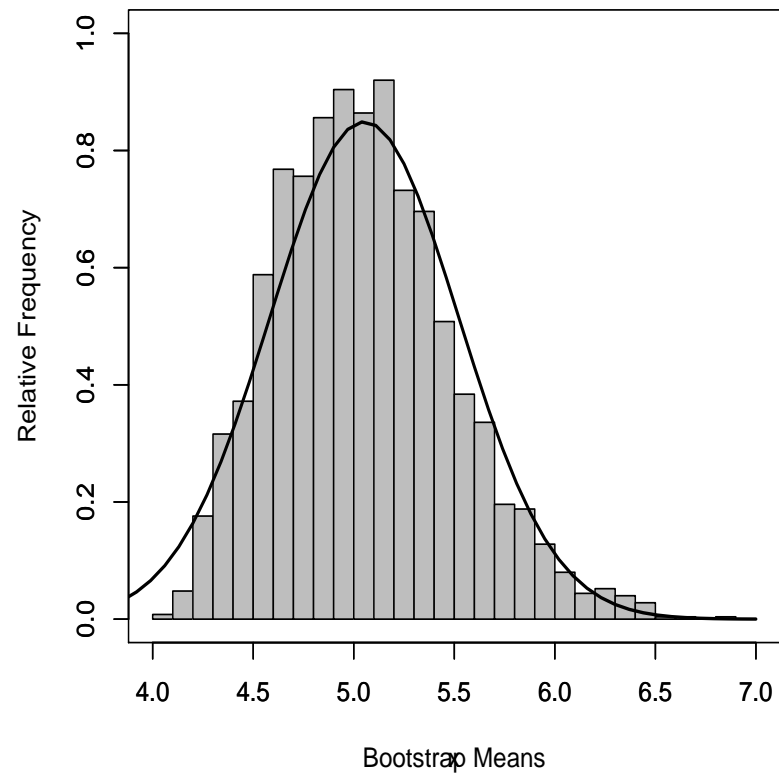
## Basic and not-so-basic statistics

### Bootstrap

- ▶ The bootstrap procedure uses the empirical distribution as a substitute for the true distribution to construct variance estimates and confidence intervals.
  - ▷ A sample  $X_1, \dots, X_n$  is **resampled** with replacement
  - ▷ The empirical distribution has a finite but large support made of  $n^n$  points
  
- ▶ For example, with data  $y$ , we can create a bootstrap sample  $y^*$  using the code
  - > `ystar=sample(y,replace=T)`
  - ▷ For each resample, we can calculate a mean, variance, etc

## Basic and not-so-basic statistics

### Simple illustration of bootstrap



- ▶ A histogram of 2500 bootstrap means
- ▶ Along with the normal approximation
- ▶ Bootstrap shows some skewness
- ▶ R `code`



Basic and not-so-basic statistics  
Bootstrapping Regression

- ▶ The bootstrap is not a panacea
  - ▷ Not always clear which quantity should be bootstrapped
  - ▷ In regression, bootstrapping the residuals is preferred

- ▶ Linear regression

$$Y_{ij} = \alpha + \beta x_i + \varepsilon_{ij},$$

$\alpha$  and  $\beta$  are the unknown intercept and slope,  $\varepsilon_{ij}$  are the iid normal errors

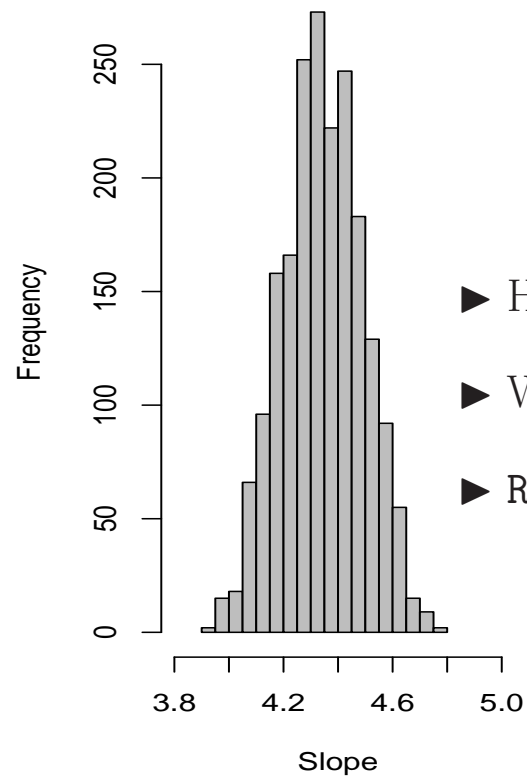
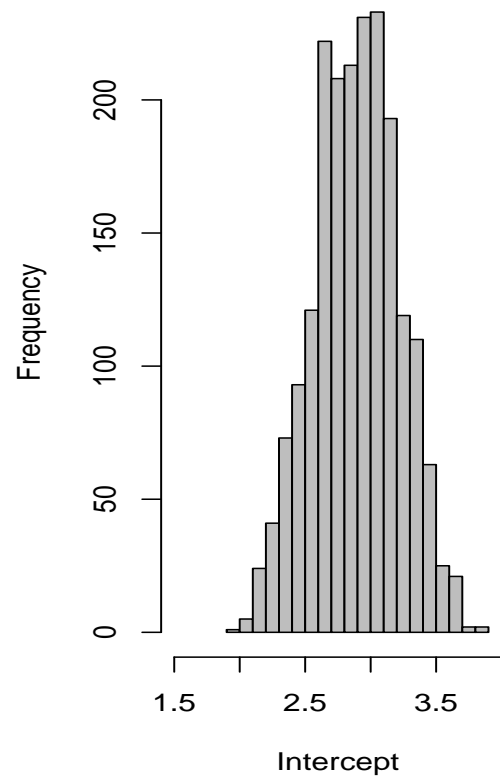
- ▶ The residuals from the least squares fit are given by

$$\hat{\varepsilon}_{ij} = y_{ij} - \hat{\alpha} - \hat{\beta}x_i,$$

- ▷ We bootstrap the residuals
- ▷ Produce a new sample  $(\hat{\varepsilon}_{ij}^*)_{ij}$  by resampling from the  $\hat{\varepsilon}_{ij}$ 's
- ▷ The bootstrap samples are then  $y_{ij}^* = y_{ij} + \hat{\varepsilon}_{ij}^*$

## Basic and not-so-basic statistics

### Bootstrapping Regression – 2



- Histogram of 2000 bootstrap samples
- We can also get confidence intervals
- R `code`

## Basic R Programming

### Some Other Stuff

- ▶ Graphical facilities
  - ▷ Can do a lot; see `plot` and `par`
- ▶ Writing new R functions
  - ▷ `h=function(x)(sin(x)^2+cos(x)^3)^(3/2)`
  - ▷ We will do this a lot
- ▶ Input and output in R
  - ▷ `write.table`, `read.table`, `scan`
- ▶ Don't forget the `mcs` package

## Chapter 2: Random Variable Generation

*“It has long been an axiom of mine that the little things are infinitely the most important.”*

**Arthur Conan Doyle**  
*A Case of Identity*

### **This Chapter**

- ▶ We present practical techniques that can produce random variables
- ▶ From both standard and nonstandard distributions
- ▶ First: Transformation methods
- ▶ Next: Indirect Methods - Accept–Reject

## Introduction

- ▶ Monte Carlo methods rely on
  - ▷ The possibility of producing a supposedly endless flow of random variables
  - ▷ For well-known or new distributions.
- ▶ Such a simulation is, in turn,
  - ▷ Based on the production of uniform random variables on the interval  $(0, 1)$ .
- ▶ We are not concerned with the [details](#) of producing uniform random variables
- ▶ We assume the existence of such a sequence

## Introduction

### Using the R Generators

R has a large number of functions that will generate the standard random variables

```
> rgamma(3,2.5,4.5)
```

produces three independent generations from a  $\mathcal{G}(5/2, 9/2)$  distribution

- ▶ It is therefore,
  - ▷ Counter-productive
  - ▷ Inefficient
  - ▷ And even dangerous,
- ▶ To generate from those standard distributions
- ▶ If it is built into R , use it
- ▶ But....we will practice on these.
- ▶ The principles are essential to deal with distributions that are not built into R.

## Uniform Simulation

- ▶ The uniform generator in R is the function `runif`
- ▶ The only required entry is the number of values to be generated.
- ▶ The other optional parameters are `min` and `max`, with R **code**

```
> runif(100, min=2, max=5)
```

will produce 100 random variables  $\mathcal{U}(2, 5)$ .

## Uniform Simulation Checking the Generator

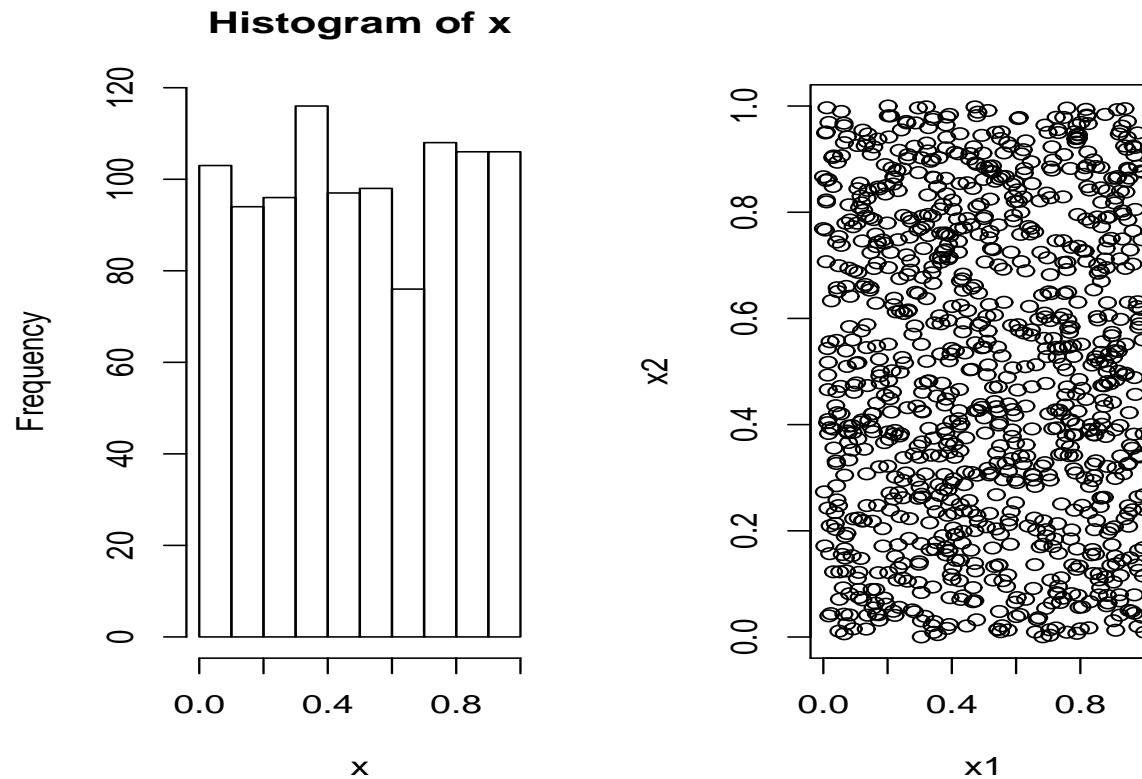
- ▶ A quick check on the properties of this uniform generator is to
  - ▷ Look at a histogram of the  $X_i$ 's,
  - ▷ Plot the pairs  $(X_i, X_{i+1})$
  - ▷ Look at the estimate autocorrelation function
- ▶ Look at the **R code**

```

> Nsim=10^4                #number of random numbers
> x=runif(Nsim)
> x1=x[-Nsim]              #vectors to plot
> x2=x[-1]                 #adjacent pairs
> par(mfrow=c(1,3))
> hist(x)
> plot(x1,x2)
> acf(x)
    
```



Uniform Simulation  
Plots from the Generator



- Histogram (*left*), pairwise plot (*center*), and estimated autocorrelation function (*right*) of a sequence of  $10^4$  uniform random numbers generated by `runif`.

## Uniform Simulation

### Some Comments

- ▶ Remember: `runif` does not involve randomness per se.
- ▶ It is a deterministic sequence based on a random starting point.
- ▶ The R function `set.seed` can produce the same sequence.

```
> set.seed(1)
> runif(5)
[1] 0.2655087 0.3721239 0.5728534 0.9082078 0.2016819
> set.seed(1)
> runif(5)
[1] 0.2655087 0.3721239 0.5728534 0.9082078 0.2016819
> set.seed(2)
> runif(5)
[1] 0.0693609 0.8177752 0.9426217 0.2693818 0.1693481
```

- ▶ Setting the seed determines all the subsequent values

## The Inverse Transform

### ► The *Probability Integral Transform*

▷ Allows us to transform a uniform into any random variable

► For example, if  $X$  has density  $f$  and cdf  $F$ , then we have the relation

$$F(x) = \int_{-\infty}^x f(t) dt,$$

and we set  $U = F(X)$  and solve for  $X$

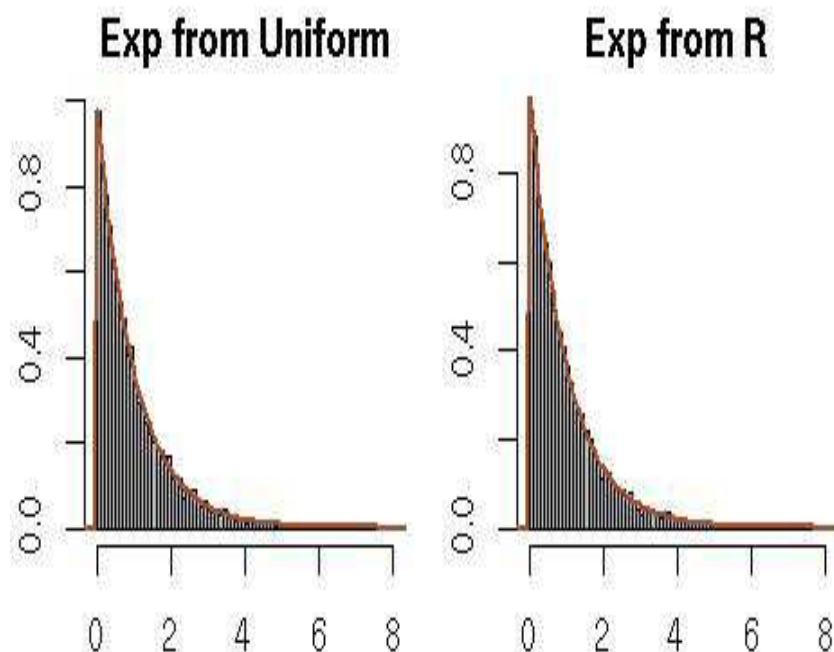
### ► **Example 2.1**

▷ If  $X \sim \mathcal{Exp}(1)$ , then  $F(x) = 1 - e^{-x}$

▷ Solving for  $x$  in  $u = 1 - e^{-x}$  gives  $x = -\log(1 - u)$

## Generating Exponentials

```
> Nsim=10^4           #number of random variables
> U=runif(Nsim)
> X=-log(U)           #transforms of uniforms
> Y=rexp(Nsim)         #exponentials from R
> par(mfrow=c(1,2))   #plots
> hist(X,freq=F,main="Exp from Uniform")
> hist(Y,freq=F,main="Exp from R")
```



- Histograms of exponential random variables
  - ▷ Inverse transform (*right*)
  - ▷ R command **rexp** (*left*)
  - ▷  $\mathcal{Exp}(1)$  density on top

## Generating Other Random Variables From Uniforms

- ▶ This method is useful for other probability distributions
  - ▷ Ones obtained as a transformation of uniform random variables
- ▶ Logistic pdf:  $f(x) = \frac{1}{\beta} \frac{e^{-(x-\mu)/\beta}}{[1+e^{-(x-\mu)/\beta}]^2}$ , cdf:  $F(x) = \frac{1}{1+e^{-(x-\mu)/\beta}}$ .
- ▶ Cauchy pdf:  $f(x) = \frac{1}{\pi\sigma} \frac{1}{1+\left(\frac{x-\mu}{\sigma}\right)^2}$ , cdf:  $F(x) = \frac{1}{2} + \frac{1}{\pi} \arctan((x - \mu)/\sigma)$ .

## General Transformation Methods

- When a density  $f$  is linked in a relatively simple way
  - ▷ To another distribution easy to simulate
  - ▷ This relationship can be use to construct an algorithm to simulate from  $f$
- If the  $X_i$ 's are iid  $\mathcal{Exp}(1)$  random variables,
  - ▷ Three standard distributions can be derived as

$$Y = 2 \sum_{j=1}^{\nu} X_j \sim \chi_{2\nu}^2, \quad \nu \in \mathbb{N}^*,$$

$$Y = \beta \sum_{j=1}^a X_j \sim \mathcal{G}(a, \beta), \quad a \in \mathbb{N}^*,$$

$$Y = \frac{\sum_{j=1}^a X_j}{\sum_{j=1}^{a+b} X_j} \sim \mathcal{Be}(a, b), \quad a, b \in \mathbb{N}^*,$$

where  $\mathbb{N}^* = \{1, 2, \dots\}$ .

## General Transformation Methods

### $\chi^2_6$ Random Variables

- For example, to generate  $\chi^2_6$  random variables, we could use the R [code](#)

```
> U=runif(3*10^4)
> U=matrix(data=U,nrow=3) #matrix for sums
> X=-log(U)                #uniform to exponential
> X=2* apply(X,2,sum)       #sum up to get chi squares
```

- [Not nearly as efficient as calling rchisq](#), as can be checked by the R [code](#)

```
> system.time(test1());system.time(test2())
  user  system elapsed
0.104   0.000   0.107
  user  system elapsed
0.004   0.000   0.004
```

- `test1` corresponds to the R [code](#) above
- `test2` corresponds to `X=rchisq(10^4,df=6)`

## General Transformation Methods

### Comments

- ▶ These transformations are quite simple and will be used in our illustrations.
- ▶ However, there are limits to their usefulness,
  - ▷ No odd degrees of freedom
  - ▷ No normals
- ▶ For any specific distribution, efficient algorithms have been developed.
- ▶ Thus, if **R** has a distribution built in, it is almost always worth using



## General Transformation Methods A Normal Generator

- ▶ Box–Muller algorithm - two normals from two uniforms

- ▶ If  $U_1$  and  $U_2$  are iid  $\mathcal{U}_{[0,1]}$

- ▶ The variables  $X_1$  and  $X_2$

$$X_1 = \sqrt{-2 \log(U_1)} \cos(2\pi U_2) , \quad X_2 = \sqrt{-2 \log(U_1)} \sin(2\pi U_2) ,$$

- ▶ Are iid  $\mathcal{N}(0, 1)$  by virtue of a change of variable argument.

- ▶ The Box–Muller algorithm is exact, not a crude CLT-based approximation

- ▶ Note that this is *not* the generator implemented in **R**

- ▷ It uses the probability inverse transform

- ▷ With a very accurate representation of the normal cdf

## General Transformation Methods Multivariate Normals

- ▶ Can simulate a multivariate normal variable using univariate normals
    - ▷ Cholesky decomposition of  $\Sigma = AA'$
    - ▷  $Y \sim \mathcal{N}_p(0, I) \Rightarrow AY \sim \mathcal{N}_p(0, \Sigma)$
  - ▶ There is an **R** package that replicates those steps, called **rmnorm**
    - ▷ In the **mnormt** library
    - ▷ Can also calculate the probability of hypercubes with the function **sadmvn**
- ```
> sadmvn(low=c(1,2,3),upp=c(10,11,12),mean=rep(0,3),var=B)
[1] 9.012408e-05
attr(,"error")
[1] 1.729111e-08
```
- ▶ B is a positive-definite matrix
  - ▶ This is quite useful since the analytic derivation of this probability is almost always impossible.

## Discrete Distributions

- ▶ To generate discrete random variables we have an “all-purpose” algorithm.
- ▶ Based on the inverse transform principle
- ▶ To generate  $X \sim P_\theta$ , where  $P_\theta$  is supported by the integers,
  - ▷ We can calculate—the probabilities
  - ▷ Once for all, assuming we can store them

$$p_0 = P_\theta(X \leq 0), \quad p_1 = P_\theta(X \leq 1), \quad p_2 = P_\theta(X \leq 2), \quad \dots,$$

- ▷ And then generate  $U \sim \mathcal{U}_{[0,1]}$  and take

$$X = k \text{ if } p_{k-1} < U < p_k.$$

## Discrete Distributions

### Binomial

► **Example** To generate  $X \sim \mathcal{Bin}(10, .3)$

▷ The probability values are obtained by `pbinom(k,10,.3)`

$$p_0 = 0.028, \quad p_1 = 0.149, \quad p_2 = 0.382, \dots, p_{10} = 1,$$

▷ And to generate  $X \sim \mathcal{P}(7)$ , take

$$p_0 = 0.0009, \quad p_1 = 0.0073, \quad p_2 = 0.0296, \dots,$$

▷ Stopping the sequence when it reaches 1 with a given number of decimals.

▷ For instance,  $p_{20} = 0.999985$ .

► Check the **R code**

## Discrete Distributions

### Comments

- ▶ Specific algorithms are usually more efficient
- ▶ Improvement can come from a judicious choice of the probabilities first computed.
- ▶ For example, if we want to generate from a Poisson with  $\lambda = 100$ 
  - ▷ The algorithm above is woefully inefficient
  - ▷ We expect most of our observations to be in the interval  $\lambda \pm 3\sqrt{\lambda}$
  - ▷ For  $\lambda = 100$  this interval is  $(70, 130)$
  - ▷ Thus, starting at 0 is quite wasteful
- ▶ A first remedy is to “ignore” what is outside of a highly likely interval
  - ▷ In the current example  $P(X < 70) + P(X > 130) = 0.00268$ .

## Discrete Distributions

### Poisson R Code

► R code that can be used to generate Poisson random variables for large values of lambda.

► The sequence `t` contains the integer values in the range around the mean.

```
> Nsim=10^4; lambda=100
> spread=3*sqrt(lambda)
> t=round(seq(max(0,lambda-spread),lambda+spread,1))
> prob=ppois(t, lambda)
> X=rep(0,Nsim)
> for (i in 1:Nsim){
+   u=runif(1)
+   X[i]=t[1]+sum(prob<u)-1 }
```

► The last line of the program checks to see what interval the uniform random variable fell in and assigns the correct Poisson value to  $X$ .

## Discrete Distributions

### Comments

- ▶ Another remedy is to start the cumulative probabilities at the mode of the discrete distribution
  - ▶ Then explore neighboring values until the cumulative probability is almost 1.
- 
- ▶ Specific algorithms exist for almost any distribution and are often quite fast.
  - ▶ So, if **R** has it, use it.
  - ▶ But **R** does not handle every distribution that we will need,

## Mixture Representations

- ▶ It is sometimes the case that a probability distribution can be naturally represented as a *mixture distribution*
- ▶ That is, we can write it in the form

$$f(x) = \int_{\mathcal{Y}} g(x|y)p(y) \, dy \quad \text{or} \quad f(x) = \sum_{i \in \mathcal{Y}} p_i f_i(x) ,$$

- ▷ The mixing distribution can be continuous or discrete.
- ▶ To generate a random variable  $X$  using such a representation,
  - ▷ we can first generate a variable  $Y$  from the mixing distribution
  - ▷ Then generate  $X$  from the selected conditional distribution



## Mixture Representations

### Generating the Mixture

#### ► Continuous

$$f(x) = \int_{\mathcal{Y}} g(x|y)p(y) \, dy \Rightarrow y \sim p(y) \text{ and } X \sim f(x|y), \text{ then } X \sim f(x)$$

#### ► Discrete

$$f(x) = \sum_{i \in \mathcal{Y}} p_i f_i(x) \Rightarrow i \sim p_i \text{ and } X \sim f_i(x), \text{ then } X \sim f(x)$$

#### ► Discrete Normal Mixture R `code`

$$\triangleright p_1 * N(\mu_1, \sigma_1) + p_2 * N(\mu_2, \sigma_2) + p_3 * N(\mu_3, \sigma_3)$$

## Mixture Representations

### Continuous Mixtures

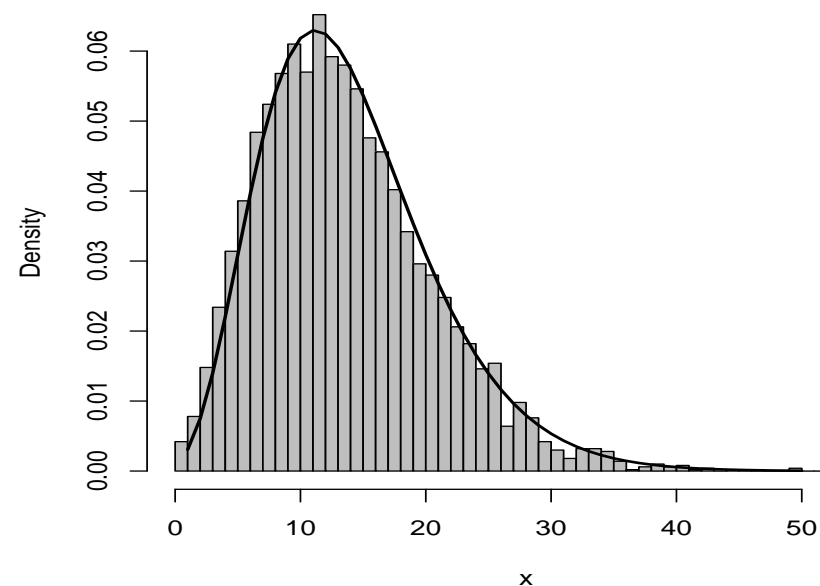
► Student's  $t$  density with  $\nu$  degrees of freedom

$$X|y \sim \mathcal{N}(0, \nu/y) \quad \text{and} \quad Y \sim \chi_\nu^2.$$

- ▷ Generate from a  $\chi_\nu^2$  then from the corresponding normal distribution
- ▷ Obviously, using **rt** is slightly more efficient

► If  $X$  is negative binomial  $X \sim \mathcal{Neg}(n, p)$

- ▷  $X|y \sim \mathcal{P}(y)$  and  $Y \sim \mathcal{G}(n, \beta)$ ,
- ▷ R **code** generates from this mixture



## Accept–Reject Methods

### Introduction

- ▶ There are many distributions where transform methods fail
- ▶ For these cases, we must turn to *indirect* methods
  - ▷ We generate a candidate random variable
  - ▷ Only accept it subject to passing a test
- ▶ This class of methods is extremely powerful.
  - ▷ It will allow us to simulate from virtually any distribution.
- ▶ Accept–Reject Methods
  - ▷ Only require the functional form of the density  $f$  of interest
  - ▷  $f$  = target,  $g$ =candidate
- ▶ Where it is simpler to simulate random variables from  $g$

## Accept–Reject Methods Accept–Reject Algorithm

- ▶ The only constraints we impose on this candidate density  $g$ 
  - ▷  $f$  and  $g$  have compatible supports (i.e.,  $g(x) > 0$  when  $f(x) > 0$ ).
  - ▷ There is a constant  $M$  with  $f(x)/g(x) \leq M$  for all  $x$ .
- ▶  $X \sim f$  can be simulated as follows.
  - ▷ Generate  $Y \sim g$  and, independently, generate  $U \sim \mathcal{U}_{[0,1]}$ .
  - ▷ If  $U \leq \frac{1}{M} \frac{f(Y)}{g(Y)}$ , set  $X = Y$ .
  - ▷ If the inequality is not satisfied, we then discard  $Y$  and  $U$  and start again.
- ▶ Note that  $M = \sup_x \frac{f(x)}{g(x)}$
- ▶  $P(\text{Accept}) = \frac{1}{M}$ , Expected Waiting Time =  $M$

## Accept–Reject Algorithm R Implementation

Succinctly, the Accept–Reject Algorithm is

### Accept–Reject Method

1. Generate  $Y \sim g$ ,  $U \sim \mathcal{U}_{[0,1]}$ ;
2. Accept  $X = Y$  if  $U \leq f(Y)/Mg(Y)$ ;
3. Return to 1 otherwise.

► R implementation: If **randg** generates from  $g$

```
> u=runif(1)*M
> y=randg(1)
> while (u>f(y)/g(y))
{
  u=runif(1)*M
  y=randg(1)
}
```

► Produces a single generation  $y$  from  $f$

## Accept–Reject Algorithm Normals from Double Exponentials

► **Candidate**  $Y \sim \frac{1}{2} \exp(-|y|)$

► **Target**  $X \sim \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$

$$\frac{\frac{1}{\sqrt{2\pi}} \exp(-y^2/2)}{\frac{1}{2} \exp(-|y|)} \leq \frac{2}{\sqrt{2\pi}} \exp(1/2)$$

▷ Maximum at  $y = 1$

► Accept  $Y$  if  $U \leq \exp(-.5Y^2 + |Y| - .5)$

► Look at R **code**

## Accept–Reject Algorithm Theory

- ▶ Why does this method work?
- ▶ A straightforward probability calculation shows

$$P(Y \leq x | \text{Accept}) = P\left(Y \leq x | U \leq \frac{f(Y)}{Mg(Y)}\right) = P(X \leq x)$$

▷ Simulating from  $g$ , the output of this algorithm is exactly distributed from  $f$ .

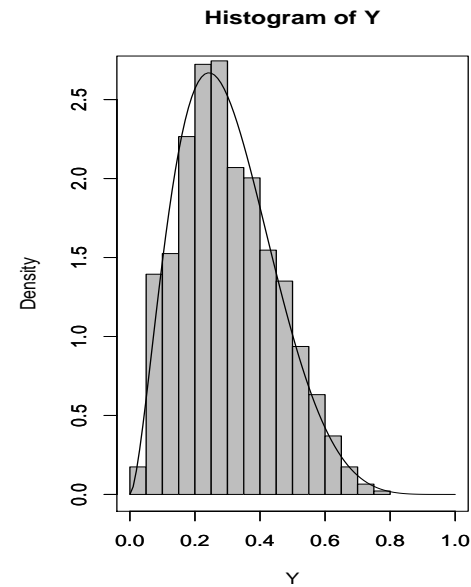
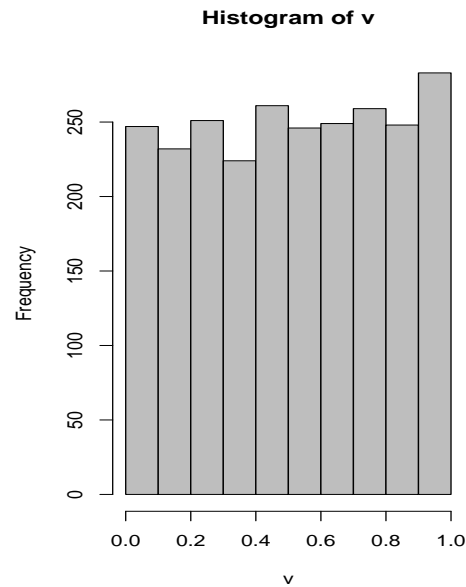


- ▶ The Accept–Reject method is applicable in any dimension
- ▶ As long as  $g$  is a density over the same space as  $f$ .

- ▶ Only need to know  $f/g$  up to a constant
- ▶ Only need an upper bound on  $M$

## Accept–Reject Algorithm Betas from Uniforms

- Generate  $X \sim \text{beta}(a, b)$ .
- No direct method if  $a$  and  $b$  are not integers.
- Use a uniform candidate
- For  $a = 2.7$  and  $b = 6.3$

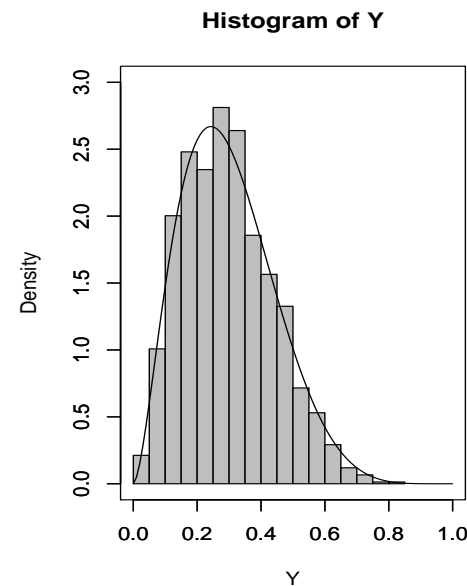
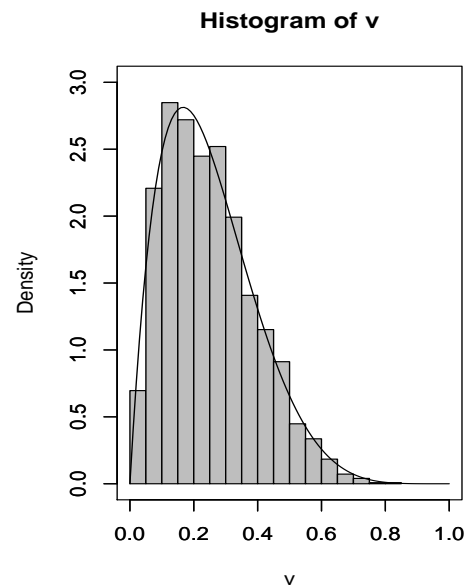


► Acceptance Rate = 37%



## Accept–Reject Algorithm Betas from Betas

- Generate  $X \sim \text{beta}(a, b)$ .
- No direct method if  $a$  and  $b$  are not integers.
- Use a beta candidate
- For  $a = 2.7$  and  $b = 6.3$ ,  $Y \sim \text{beta}(2, 6)$



► Acceptance Rate = 60%

Accept–Reject Algorithm  
Betas from Betas-Details

- ▶ Beta density  $\propto x^a(1-x)^b$
- ▶ Can generate if  $a$  and  $b$  integers
- ▶ If not, use candidate with  $a_1$  and  $b_1$  integers

$$\frac{y^a(1-y)^b}{y^{a_1}(1-y)^{b_1}} \text{ maximized at } y = \frac{a - a_1}{a - a_1 + b - b_1}$$

- ▷ Need  $a_1 < a$  and  $b_1 < b$
- ▶ Efficiency  $\uparrow$  as the candidate gets closer to the target
- ▶ Look at R code

## Accept–Reject Algorithm

### Comments

⚡ Some key properties of the Accept–Reject algorithm::

1. Only the ratio  $f/M$  is needed

▷ So the algorithm does not depend on the normalizing constant.

2. The bound  $f \leq Mg$  need not be tight

▷ Accept–Reject is valid, but less efficient, if  $M$  is replaced with a larger constant.

3. The probability of acceptance is  $1/M$

▷ So  $M$  should be as small as possible for a given computational effort.

## Chapter 3: Monte Carlo Integration

*“Every time I think I know what’s going on, suddenly there’s another layer of complications. I just want this damn thing solved.”*

**John Scalzi**  
*The Last Colony*

### **This Chapter**

- ▶ This chapter introduces the major concepts of Monte Carlo methods
- ▶ The validity of Monte Carlo approximations relies on the Law of Large Numbers
- ▶ The versatility of the representation of an integral as an expectation

## Monte Carlo Integration

### Introduction

- We will be concerned with evaluating integrals of the form

$$\int_{\mathcal{X}} h(x) f(x) \, dx,$$

- ▷  $f$  is a density
  - ▷ We can produce an almost infinite number of random variables from  $f$
- We apply probabilistic results
  - ▷ Law of Large Numbers
  - ▷ Central Limit Theorem
- The Alternative - Deterministic Numerical Integration
  - ▷ R functions `area` and `integrate`
  - ▷ OK in low (one) dimensions
  - ▷ Usually needs some knowledge of the function

## Classical Monte Carlo Integration

### The Monte Carlo Method

- The generic problem: Evaluate

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x) f(x) \, dx,$$

- ▷  $X$  takes its values in  $\mathcal{X}$

- The Monte Carlo Method

- ▷ Generate a sample  $(X_1, \dots, X_n)$  from the density  $f$
  - ▷ Approximate the integral with

$$\bar{h}_n = \frac{1}{n} \sum_{j=1}^n h(x_j) ,$$

## Classical Monte Carlo Integration

### Validating the Monte Carlo Method

#### ► The Convergence

$$\bar{h}_n = \frac{1}{n} \sum_{j=1}^n h(x_j) \rightarrow \int_{\mathcal{X}} h(x) f(x) \, dx = \mathbb{E}_f[h(X)]$$

▷ Is valid by the Strong Law of Large Numbers

#### ► When $h^2(X)$ has a finite expectation under $f$ ,

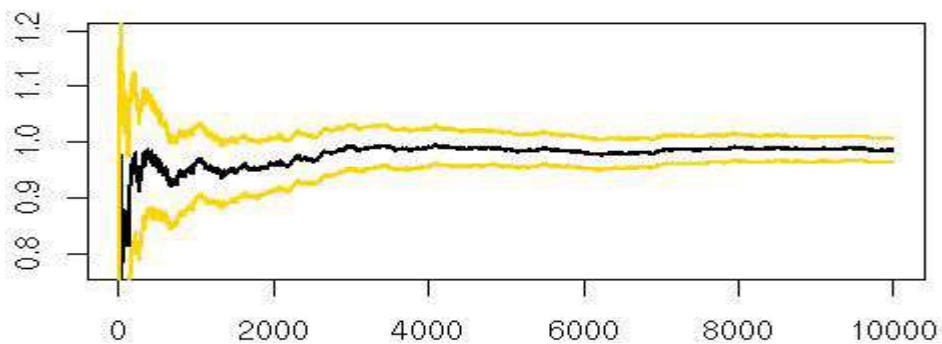
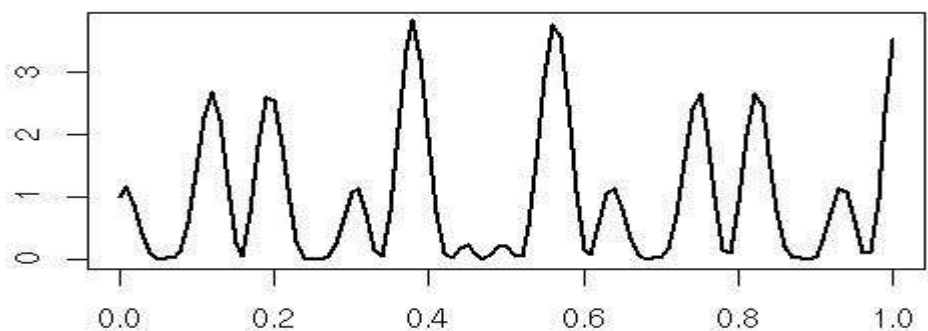
$$\frac{\bar{h}_n - \mathbb{E}_f[h(X)]}{\sqrt{v_n}} \rightarrow \mathcal{N}(0, 1)$$

▷ Follows from the Central Limit Theorem

▷  $v_n = \frac{1}{n^2} \sum_{j=1}^n [h(x_j) - \bar{h}_n]^2.$

## Classical Monte Carlo Integration A First Example

► Look at the function



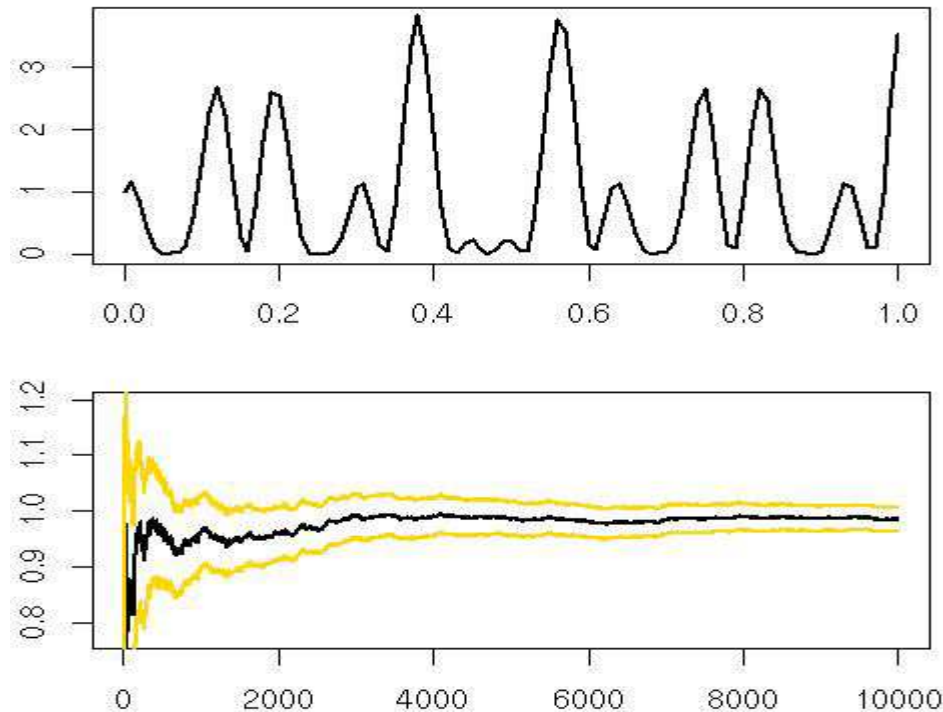
►  $h(x) = [\cos(50x) + \sin(20x)]^2$

► Monitoring Convergence

► R `code`



## Classical Monte Carlo Integration A Caution



- The confidence band produced in this figure is not a 95% confidence band in the classical sense

- They are **Confidence Intervals** were you to stop at a chosen number of iterations

## Classical Monte Carlo Integration Comments

⚡

- ▶ The evaluation of the Monte Carlo error is a bonus
- ▶ It assumes that  $v_n$  is a **proper** estimate of the variance of  $\bar{h}_n$
- ▶ If  $v_n$  does not converge, converges too slowly, a CLT may not apply

## Classical Monte Carlo Integration

### Another Example

#### ► Normal Probability

$$\hat{\Phi}(t) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}_{x_i \leq t} \rightarrow \Phi(t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy$$

▷ The exact variance  $\Phi(t)[1 - \Phi(t)]/n$

▷ Conservative:  $\text{Var} \approx 1/4n$

▷ For a precision of four decimals

▷ Want  $2 \times \sqrt{1/4n} \leq 10^{-4}$  simulations

▷ Take  $n = (10^4)^2 = 10^8$

► This method breaks down for tail probabilities

## Importance Sampling Introduction

► **Importance sampling** is based on an alternative formulation of the SLLN

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x) \frac{f(x)}{g(x)} g(x) \, dx = \mathbb{E}_g \left[ \frac{h(X)f(X)}{g(X)} \right] ;$$

- ▷  $f$  is the target density
- ▷  $g$  is the candidate density
  
- ▷ Sound Familiar?

## Importance Sampling Introduction

- Importance sampling is based on an alternative formulation of the SLLN

$$\mathbb{E}_f[h(X)] = \int_{\mathcal{X}} h(x) \frac{f(x)}{g(x)} g(x) \, dx = \mathbb{E}_g \left[ \frac{h(X)f(X)}{g(X)} \right] ;$$

- ▷  $f$  is the target density
- ▷  $g$  is the candidate density
- ▷ Sound Familiar? – Just like Accept–Reject

- So

$$\frac{1}{n} \sum_{j=1}^n \frac{f(X_j)}{g(X_j)} h(X_j) \rightarrow \mathbb{E}_f[h(X)]$$

- As long as

- ▷  $\text{Var}(h(X)f(X)/g(X)) < \infty$
- ▷  $\text{supp}(g) \supset \text{supp}(h \times f)$

## Importance Sampling Revisiting Normal Tail Probabilities

- ▶  $Z \sim \mathcal{N}(0, 1)$  and we are interested in the probability  $P(Z > 4.5)$
- ▶ 

```
> pnorm(-4.5, log=T)
```

```
[1] -12.59242
```
- ▶ Simulating  $Z^{(i)} \sim \mathcal{N}(0, 1)$  only produces a hit once in about 3 million iterations!
  - ▷ Very rare event for the normal
  - ▷ Not-so-rare for a distribution sitting out there!
- ▶ Take  $g = \mathcal{E}xp(1)$  truncated at 4.5:

$$g(y) = \frac{e^{-y}}{\int_{4.5}^{\infty} e^{-x} dx} = e^{-(y-4.5)},$$

- ▶ The IS estimator is

$$\frac{1}{n} \sum_{i=1}^n \frac{f(Y^{(i)})}{g(Y^{(i)})} = \frac{1}{n} \sum_{i=1}^n \frac{e^{-Y_i^2/2 + Y_i - 4.5}}{\sqrt{2\pi}}$$

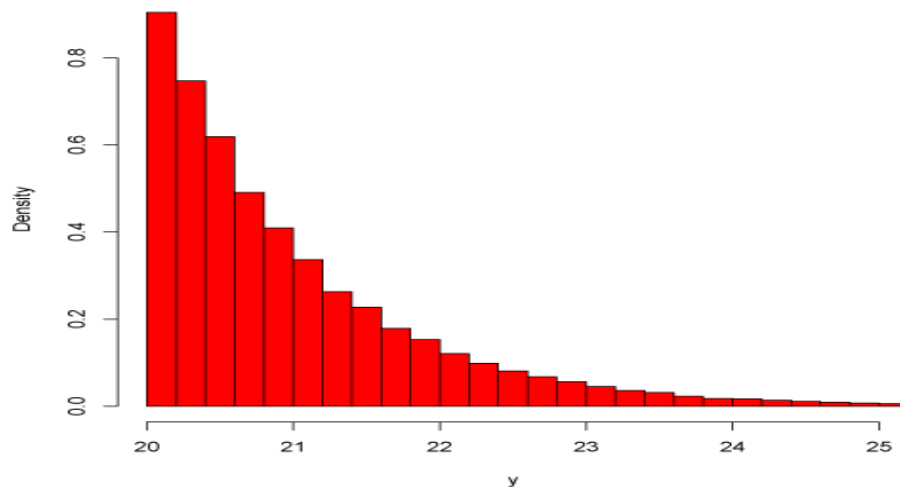
R code

## Importance Sampling Normal Tail Variables

- ▶ The Importance sampler does not give us a sample  $\Rightarrow$  Can use Accept–Reject
- ▶ Sample  $Z \sim \mathcal{N}(0, 1)$ ,  $Z > a \Rightarrow$  Use Exponential Candidate

$$\frac{\frac{1}{\sqrt{2\pi}} \exp(-.5x^2)}{\exp(-(x-a))} = \frac{1}{\sqrt{2\pi}} \exp(-.5x^2 + x + a) \leq \frac{1}{\sqrt{2\pi}} \exp(-.5a^{*2} + a^* + a)$$

▷ Where  $a^* = \max\{a, 1\}$



- ▶ Normals  $> 20$
- ▶ The Twilight Zone
- ▶ R code

## Importance Sampling Comments

- ⚡ Importance sampling has little restriction on the choice of the candidate
  - ▶  $g$  can be chosen from distributions that are easy to simulate
    - ▷ Or efficient in the approximation of the integral.
  - ▶ Moreover, the same sample (generated from  $g$ ) can be used repeatedly
    - ▷ Not only for different functions  $h$  but also for different densities  $f$ .



## Importance Sampling Easy Model - Difficult Distribution

### Example: Beta posterior importance approximation

- Have an observation  $x$  from a beta  $\mathcal{B}(\alpha, \beta)$  distribution,

$$x \sim \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1 - x)^{\beta-1} \mathbb{I}_{[0,1]}(x)$$

- There exists a family of conjugate priors on  $(\alpha, \beta)$  of the form

$$\pi(\alpha, \beta) \propto \left\{ \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \right\}^{\lambda} x_0^{\alpha} y_0^{\beta},$$

where  $\lambda, x_0, y_0$  are hyperparameters,

- The posterior is then equal to

$$\pi(\alpha, \beta | x) \propto \left\{ \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \right\}^{\lambda+1} [xx_0]^{\alpha} [(1 - x)y_0]^{\beta}.$$

## Importance Sampling

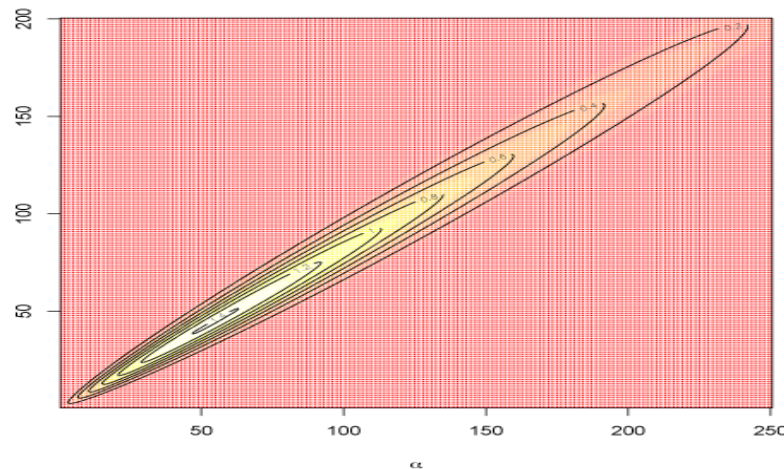
### Easy Model - Difficult Distribution -2

- The posterior distribution is intractable

$$\pi(\alpha, \beta | x) \propto \left\{ \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \right\}^{\lambda+1} [xx_0]^\alpha [(1-x)y_0]^\beta.$$

- ▷ Difficult to deal with the gamma functions
- ▷ Simulating directly from  $\pi(\alpha, \beta | x)$  is impossible.

- What candidate to use?



- Contour Plot
- Suggest a candidate?
- R code

## Importance Sampling

### Easy Model - Difficult Distribution – 3

- ▶ Try a Bivariate Student's  $T$  (or Normal)
- ▶ Trial and error
  - ▷ Student's  $\mathcal{T}(3, \mu, \Sigma)$  distribution with  $\mu = (50, 45)$  and

$$\Sigma = \begin{pmatrix} 220 & 190 \\ 190 & 180 \end{pmatrix}$$

- ▷ Produce a reasonable fit
  - ▷ R code
- ▶ Note that we are using the fact that

$$X \sim f(x) \Rightarrow \Sigma^{1/2}X + \mu \sim f\left((x - \mu)' \Sigma^{-1}(x - \mu)\right)$$

## Importance Sampling

### Easy Model - Difficult Distribution – Posterior Means

- The posterior mean of  $\alpha$  is

$$\int \int \alpha \pi(\alpha, \beta | x) d\alpha d\beta = \int \int \left[ \alpha \frac{\pi(\alpha, \beta | x)}{g(\alpha, \beta)} \right] g(\alpha, \beta) d\alpha d\beta \approx \frac{1}{M} \sum_{i=1}^M \alpha_i \frac{\pi(\alpha_i, \beta_i | x)}{g(\alpha_i, \beta_i)}$$

where

$$\triangleright \pi(\alpha, \beta | x) \propto \left\{ \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \right\}^{\lambda+1} [xx_0]^\alpha [(1-x)y_0]^\beta$$

$$\triangleright g(\alpha, \beta) = \mathcal{T}(3, \mu, \Sigma)$$

- Note that  $\pi(\alpha, \beta | x)$  is not normalized, so we have to calculate

$$\frac{\int \int \alpha \pi(\alpha, \beta | x) d\alpha d\beta}{\int \int \pi(\alpha, \beta | x) d\alpha d\beta} \approx \frac{\sum_{i=1}^M \alpha_i \frac{\pi(\alpha_i, \beta_i | x)}{g(\alpha_i, \beta_i)}}{\sum_{i=1}^M \frac{\pi(\alpha_i, \beta_i | x)}{g(\alpha_i, \beta_i)}}$$

- The same samples can be used for every posterior expectation

- R code

## Importance Sampling Probit Analysis

### Example: Probit posterior importance sampling approximation

- $y$  are binary variables, and we have covariates  $x \in \mathbb{R}^p$  such that

$$\Pr(y = 1|x) = 1 - \Pr(y = 0|x) = \Phi(x^T \beta), \quad \beta \in \mathbb{R}^p.$$

- We return to the dataset `Pima.tr`,  $x$ =BMI
- A GLM estimation of the model is (using centered  $x$ )

```
>glm(formula = y ~ x, family = binomial(link = "probit"))
```

Coefficients:

|             | Estimate | Std. Error | z value | Pr(> z ) |     |
|-------------|----------|------------|---------|----------|-----|
| (Intercept) | -0.44957 | 0.09497    | -4.734  | 2.20e-06 | *** |
| x           | 0.06479  | 0.01615    | 4.011   | 6.05e-05 | *** |
| ---         |          |            |         |          |     |

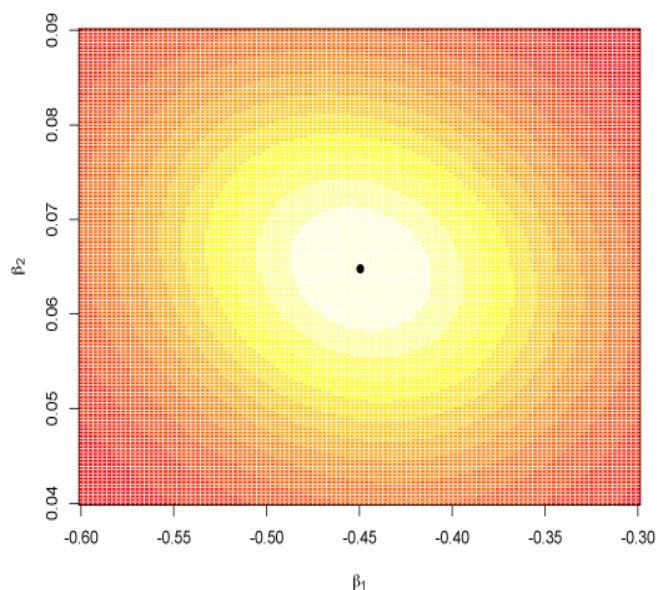
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

So BMI has a significant impact on the possible presence of diabetes.

## Importance Sampling Bayesian Probit Analysis

- ▶ From a Bayesian perspective, we use a vague prior
  - ▷  $\beta = (\beta_1, \beta_2)$  , each having a  $\mathcal{N}(0, 100)$  distribution
- ▶ With  $\Phi$  the normal cdf, the posterior is proportional to

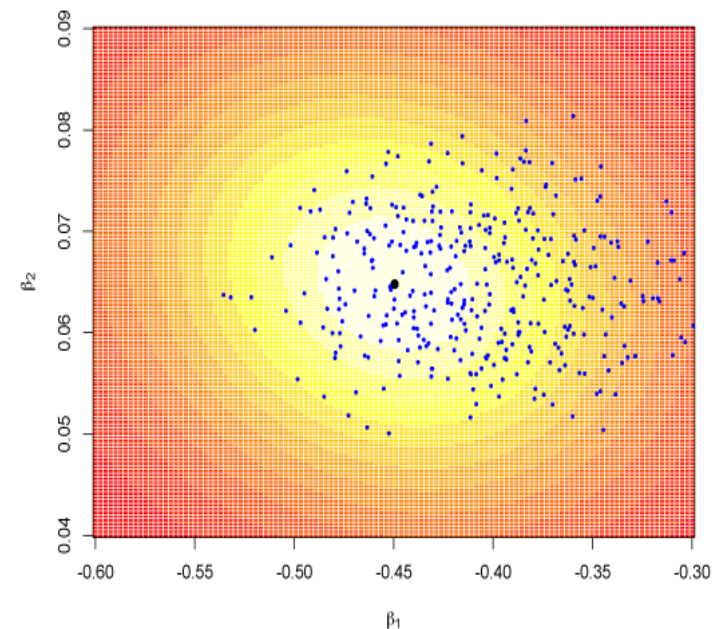
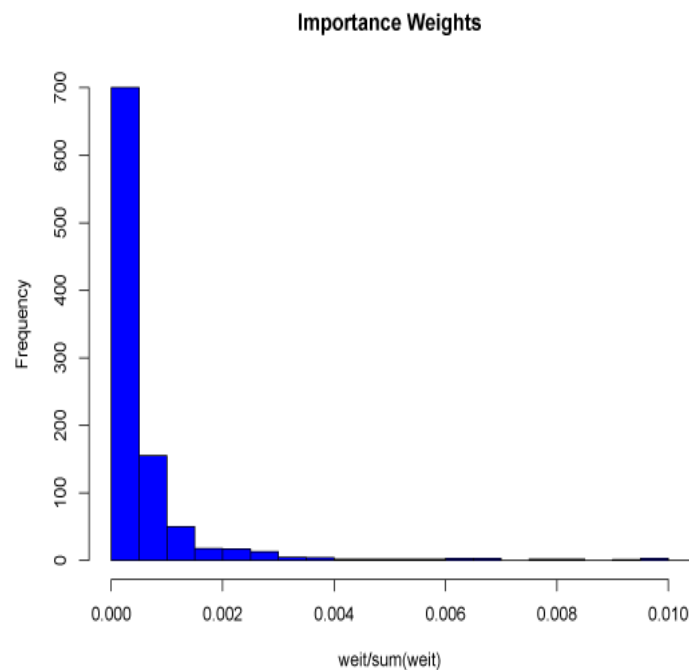
$$\prod_{i=1}^n [\Phi(\beta_1 + (x_i - \bar{x})\beta_2)]^{y_i} [\Phi(-\beta_1 - (x_i - \bar{x})\beta_2)]^{1-y_i} \times e^{-\frac{\beta_1^2 + \beta_2^2}{2 \times 100}}$$



- ▶ Level curves of posterior
- ▶ MLE in the center
- ▶ R [code](#)

## Importance Sampling Probit Analysis Importance Weights

- ▶ Normal candidate centered at the MLE - no finite variance guarantee
- ▶ The importance weights are rather uneven, if not degenerate



- ▶ Right side = reweighted candidate sample (**R** code)
- ▶ Somewhat of a failure

## Chapter 5: Monte Carlo Optimization

*“He invented a game that allowed players to predict the outcome?”*

**Susanna Gregory**

*To Kill or Cure*

### **This Chapter**

- ▶ Two uses of computer-generated random variables to solve optimization problems.
- ▶ **The first use** is to produce stochastic search techniques
  - ▷ To reach the maximum (or minimum) of a function
  - ▷ Avoid being trapped in local maxima (or minima)
  - ▷ Are sufficiently attracted by the global maximum (or minimum).
- ▶ **The second** use of simulation is to approximate the function to be optimized.



## Monte Carlo Optimization

### Introduction

- ▶ Optimization problems can mostly be seen as one of two kinds:
  - ▷ Find the extrema of a function  $h(\theta)$  over a domain  $\Theta$
  - ▷ Find the solution(s) to an implicit equation  $g(\theta) = 0$  over a domain  $\Theta$ .
- ▶ The problems are exchangeable
  - ▷ The second one is a minimization problem for a function like  $h(\theta) = g^2(\theta)$
  - ▷ while the first one is equivalent to solving  $\partial h(\theta)/\partial \theta = 0$
- ▶ We only focus on the maximization problem

## Monte Carlo Optimization Deterministic or Stochastic

- ▶ Similar to integration, optimization can be deterministic or stochastic
- ▶ **Deterministic**: performance dependent on properties of the function
  - ▷ such as convexity, boundedness, and smoothness
- ▶ **Stochastic** (simulation)
  - ▷ Properties of  $h$  play a lesser role in simulation-based approaches.
- ▶ Therefore, if  $h$  is complex or  $\Theta$  is irregular, chose the stochastic approach.

Monte Carlo Optimization  
Numerical Optimization

- ▶ R has several embedded functions to solve optimization problems
  - ▷ The simplest one is **optimize** (one dimensional)

Example: Maximizing a Cauchy likelihood  $\mathcal{C}(\theta, 1)$

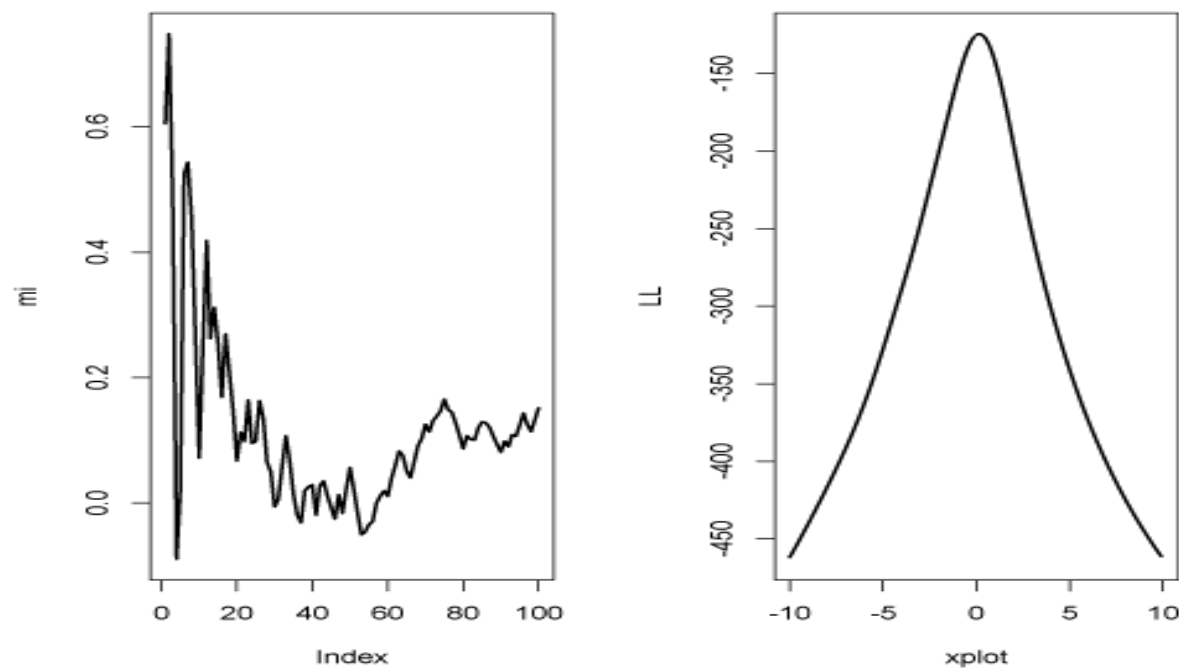
- ▶ When maximizing the likelihood of a Cauchy  $\mathcal{C}(\theta, 1)$  sample,

$$\ell(\theta|x_1, \dots, x_n) = \prod_{i=1}^n \frac{1}{1 + (x_i - \theta)^2},$$

- ▶ The sequence of maxima (MLEs)  $\rightarrow \theta^* = 0$  when  $n \rightarrow \infty$ .
- ▶ But the journey is not a smooth one...

## Monte Carlo Optimization

### Cauchy Likelihood

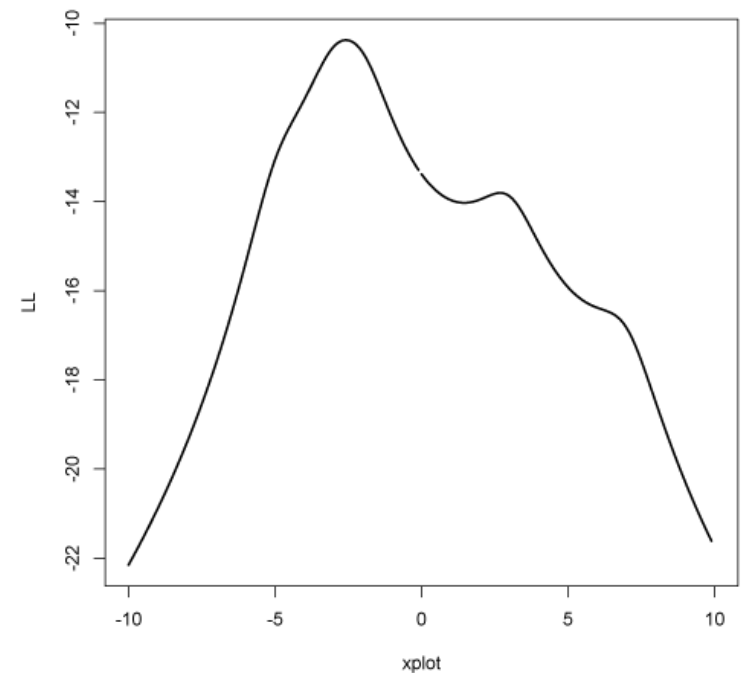


- MLEs (*left*) at each sample size,  $n = 1, 500$  , and plot of final likelihood (*right*).
  - ▷ Why are the MLEs so wiggly?
  - ▷ The likelihood is not as well-behaved as it seems

## Monte Carlo Optimization

### Cauchy Likelihood-2

- ▶ The likelihood  $\ell(\theta|x_1, \dots, x_n) = \prod_{i=1}^n \frac{1}{1+(x_i-\theta)^2}$
- ▶ Is like a polynomial of degree  $2n$
- ▶ The derivative has  $2n$  zeros
- ▶ Hard to see if  $n = 500$
- ▶ Here is  $n = 5$
- ▶ R `code`



## Monte Carlo Optimization Newton-Raphson

- ▶ Similarly, `nlm` is a generic **R** function using the Newton–Raphson method
- ▶ Based on the recurrence relation

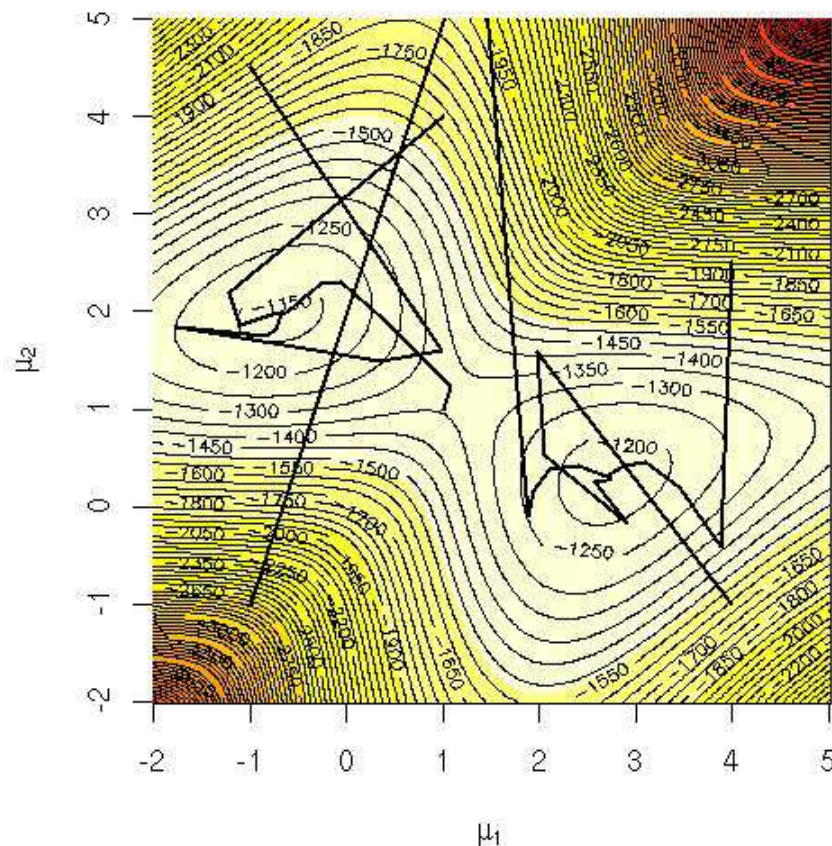
$$\theta_{i+1} = \theta_i - \left[ \frac{\partial^2 h}{\partial \theta \partial \theta^T}(\theta_i) \right]^{-1} \frac{\partial h}{\partial \theta}(\theta_i)$$

- ▶ Where the matrix of the second derivatives is called the *Hessian*
  - ▷ This method is perfect when  $h$  is quadratic
  - ▷ But may also deteriorate when  $h$  is highly nonlinear
  - ▷ It also obviously depends on the starting point  $\theta_0$  when  $h$  has several minima.

## Monte Carlo Optimization

### Newton-Raphson; Mixture Model Likelihood

► Bimodal Mixture Model Likelihood  $\frac{1}{4}\mathcal{N}(\mu_1, 1) + \frac{3}{4}\mathcal{N}(\mu_2, 1)$

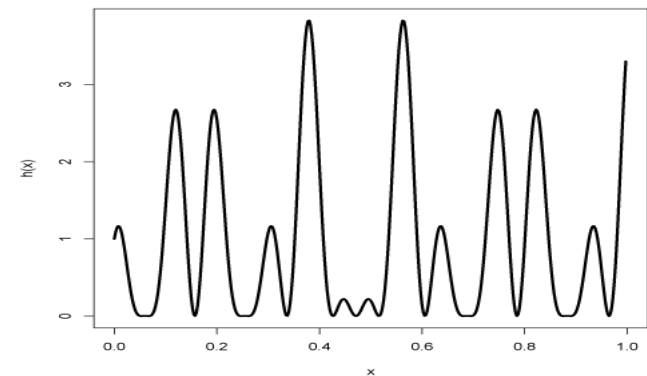
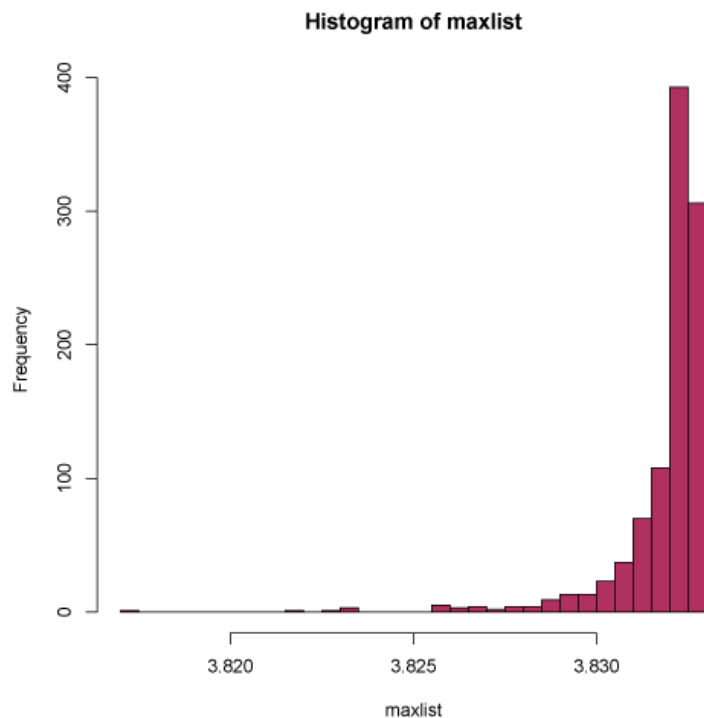


- Sequences go to the closest mode
- Starting point  $(-1, -1)$  has a steep gradient
  - ▷ Bypasses the main mode  $(-0.68, 1.98)$
  - ▷ Goes to other mode (lower likelihood)

## Stochastic search

### A Basic Solution

- A natural if rudimentary way of using simulation to find  $\max_{\theta} h(\theta)$ 
  - ▷ Simulate points over  $\Theta$  according to an arbitrary distribution  $f$  positive on  $\Theta$
  - ▷ Until a high value of  $h(\theta)$  is observed



- Recall  $h(x) = [\cos(50x) + \sin(20x)]^2$
- Max=3.8325
- Histogram of 1000 runs



Stochastic search  
Stochastic Gradient Methods

- ▶ Generating direct simulations from the target can be difficult.
- ▶ Different stochastic approach to maximization
  - ▷ Explore the surface in a local manner.
  - ▷ A Markov Chain
  - ▷ Can use  $\theta_{j+1} = \theta_j + \epsilon_j$
  - ▷ The random component  $\epsilon_j$  can be arbitrary
- ▶ Can also use features of the function: Newton-Raphson Variation

$$\theta_{j+1} = \theta_j + \alpha_j \nabla h(\theta_j) , \quad \alpha_j > 0 ,$$

- ▷ Where  $\nabla h(\theta_j)$  is the gradient
- ▷  $\alpha_j$  the step size

## Stochastic search Stochastic Gradient Methods-2

► In difficult problems

▷ The gradient sequence will most likely get stuck in a local extremum of  $h$ .

► Stochastic Variation

$$\nabla h(\theta_j) \approx \frac{h(\theta_j + \beta_j \zeta_j) - h(\theta_j - \beta_j \zeta_j)}{2\beta_j} \zeta_j = \frac{\Delta h(\theta_j, \beta_j \zeta_j)}{2\beta_j} \zeta_j,$$

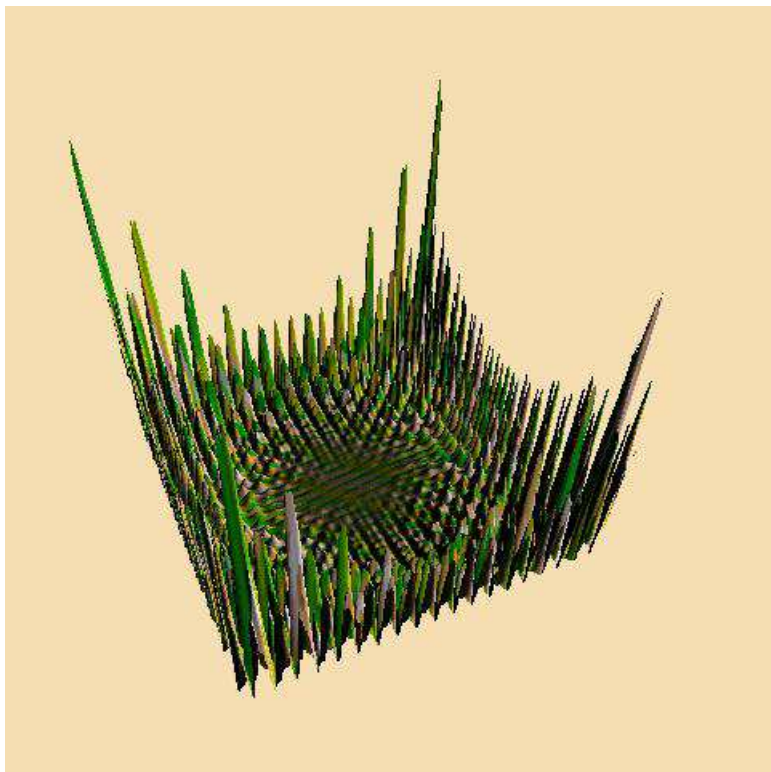
▷  $(\beta_j)$  is a second decreasing sequence

▷  $\zeta_j$  is uniform on the unit sphere  $\|\zeta_j\| = 1$ .

► We then use

$$\theta_{j+1} = \theta_j + \frac{\alpha_j}{2\beta_j} \Delta h(\theta_j, \beta_j \zeta_j) \zeta_j$$

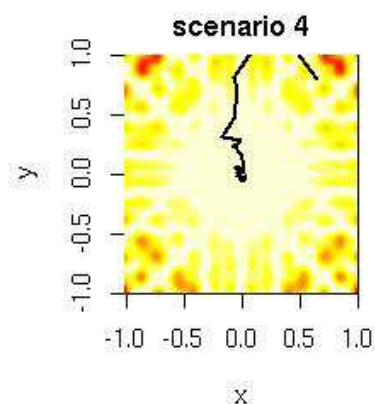
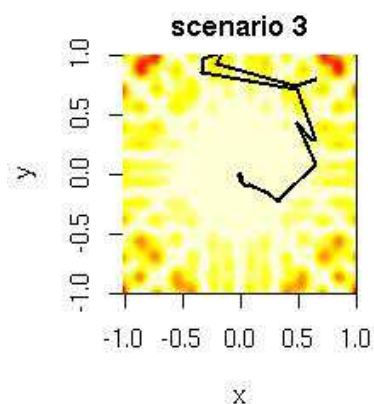
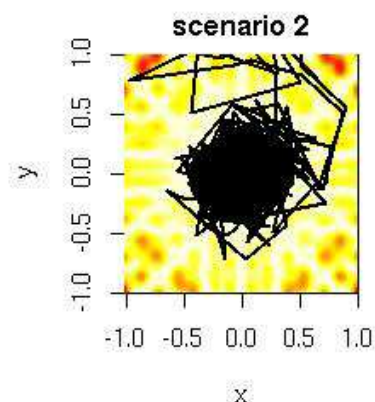
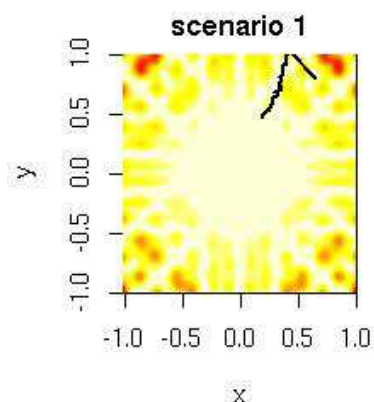
## Stochastic Search A Difficult Minimization



- ▶ Many Local Minima
- ▶ Global Min at  $(0, 0)$
- ▶ Code in the text

## Stochastic Search A Difficult Minimization – 2

| Scenario   | 1                  | 2                  | 3              | 4              |
|------------|--------------------|--------------------|----------------|----------------|
| $\alpha_j$ | $1/\log(j+1)$      | $1/100 \log(j+1)$  | $1/(j+1)$      | $1/(j+1)$      |
| $\beta_j$  | $1/\log(j+1)^{.1}$ | $1/\log(j+1)^{.1}$ | $1/(j+1)^{.5}$ | $1/(j+1)^{.1}$ |



- $\alpha \downarrow 0$  slowly,  $\sum_j \alpha_j = \infty$
- $\beta \downarrow 0$  more slowly,  $\sum_j (\alpha_j/\beta_j)^2 < \infty$
- Scenarios 1-2: Not enough energy
- Scenarios 3-4: Good

## Simulated Annealing Introduction

- ▶ This name is borrowed from Metallurgy:
- ▶ A metal manufactured by a slow decrease of temperature (*annealing*)
  - ▷ Is stronger than a metal manufactured by a fast decrease of temperature.
- ▶ The fundamental idea of simulated annealing methods
  - ▷ A change of scale, or temperature
  - ▷ Allows for faster moves on the surface of the function  $h$  to maximize.
  - ▷ Rescaling partially avoids the trapping attraction of local maxima.
- ▶ As  $T$  decreases toward 0, the values simulated from this distribution become concentrated in a narrower and narrower neighborhood of the local maxima of  $h$

Simulated Annealing  
Metropolis Algorithm/Simulated Annealing

- Simulation method proposed by Metropolis *et al.* (1953)
- Starting from  $\theta_0$ ,  $\zeta$  is generated from

$\zeta \sim \text{Uniform in a neighborhood of } \theta_0.$

- The new value of  $\theta$  is generated as

$$\theta_1 = \begin{cases} \zeta & \text{with probability } \rho = \exp(\Delta h/T) \wedge 1 \\ \theta_0 & \text{with probability } 1 - \rho, \end{cases}$$

- $\Delta h = h(\zeta) - h(\theta_0)$
- If  $h(\zeta) \geq h(\theta_0)$ ,  $\zeta$  is accepted
- If  $h(\zeta) < h(\theta_0)$ ,  $\zeta$  may still be accepted
- This allows escape from local maxima

## Simulated Annealing Metropolis Algorithm - Comments

- Simulated annealing typically modifies the temperature  $T$  at each iteration
- It has the form

1. Simulate  $\zeta$  from an instrumental distribution  
with density  $g(|\zeta - \theta_i|)$ ;

2. Accept  $\theta_{i+1} = \zeta$  with probability

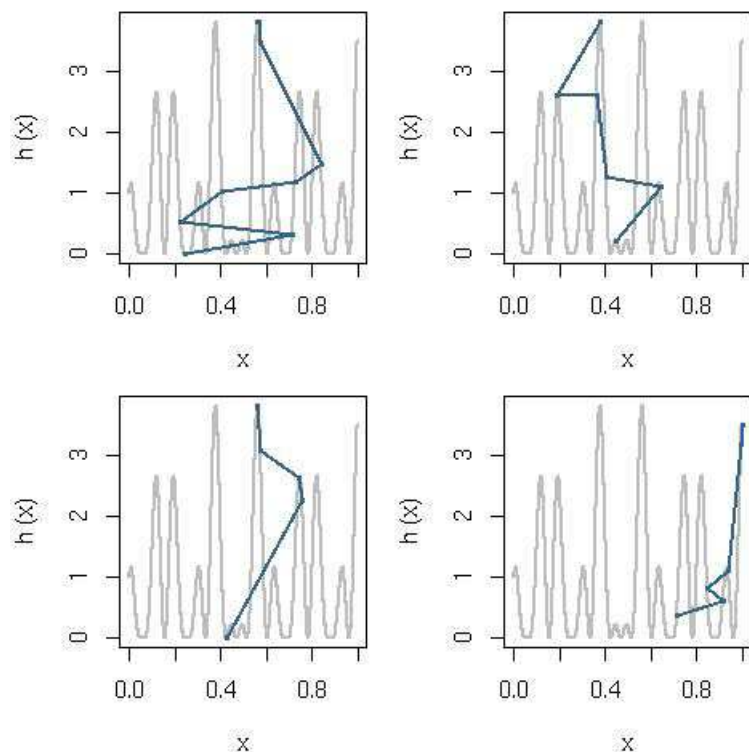
$$\rho_i = \exp\{\Delta h_i / T_i\} \wedge 1;$$

take  $\theta_{i+1} = \theta_i$  otherwise.

3. Update  $T_i$  to  $T_{i+1}$ .

- All positive moves accepted
- As  $T \downarrow 0$ 
  - Harder to accept downward moves
  - No big downward moves
- Not a Markov Chain - difficult to analyze

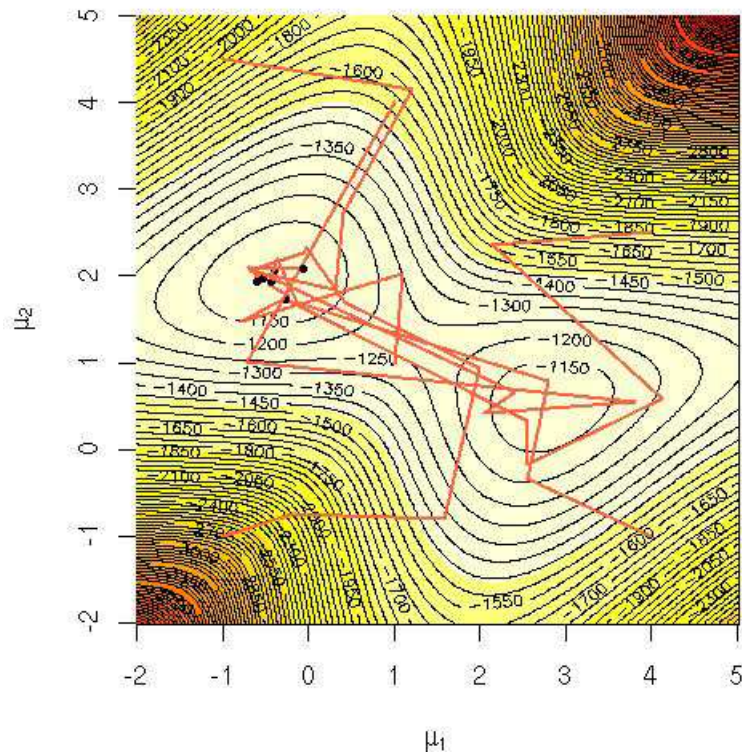
## Simulated Annealing Simple Example



- ▶ Trajectory:  $T_i = \frac{1}{(1+i)^2}$
- ▶ Log trajectory also works
- ▶ Can Guarantee Finding Global Max
- ▶ R code



## Simulated Annealing Normal Mixture



- ▶ Previous normal mixture
- ▶ Most sequences find max
- ▶ They visit both modes

## Stochastic Approximation Introduction

- ▶ We now consider methods that work with the objective function  $h$ 
  - ▷ Rather than being concerned with fast exploration of the domain  $\Theta$ .
- ▶ Unfortunately, the use of those methods results in an additional level of error
  - ▷ Due to this approximation of  $h$ .
- ▶ But, the objective function in many statistical problems can be expressed as
  - ▷  $h(x) = \mathbb{E}[H(x, Z)]$
  - ▷ This is the setting of so-called missing-data models

## Stochastic Approximation

### Optimizing Monte Carlo Approximations

- If  $h(x) = \mathbb{E}[H(x, Z)]$ , a Monte Carlo approximation is

$$\hat{h}(x) = \frac{1}{m} \sum_{i=1}^m H(x, z_i),$$

- ▷  $Z_i$ 's are generated from the conditional distribution  $f(z|x)$ .
- This approximation yields a convergent estimator of  $h(x)$  for every value of  $x$ 
  - ▷ This is a *pointwise convergent estimator*
  - ▷ Its use in optimization setups is not recommended
  - ▷ Changing sample of  $Z_i$ 's  $\Rightarrow$  unstable sequence of evaluations
  - ▷ And a rather noisy approximation to  $\arg \max h(x)$

## Stochastic Approximation Bayesian Probit

Example: Bayesian analysis of a simple probit model

- $Y \in \{0, 1\}$  has a distribution depending on a covariate  $X$ :

$$P_{\theta}(Y = 1|X = x) = 1 - P_{\theta}(Y = 0|X = x) = \Phi(\theta_0 + \theta_1 x),$$

▷ Illustrate with **Pima.tr** dataset,  $Y$ = diabetes indicator,  $X$ =BMI

- Typically infer from the **marginal posterior**

$$\arg \max_{\theta_0} \int \prod_{i=1} \Phi(\theta_0 + \theta_1 x_n)^{y_i} \Phi(-\theta_0 - \theta_1 x_n)^{1-y_i} d\theta_1 = \arg \max_{\theta_0} h(\theta_0)$$

▷ For a flat prior on  $\theta$  and a sample  $(x_1, \dots, x_n)$ .

## Stochastic Approximation Bayesian Probit – Importance Sampling

- ▶ No analytic expression for  $h$
- ▶ The conditional distribution of  $\theta_1$  given  $\theta_0$  is also nonstandard
  - ▷ Use importance sampling with a  $t$  distribution with 5 df
  - ▷ Take  $\mu = 0.1$  and  $\sigma = 0.03$  (MLEs)
- ▶ Importance Sampling Approximation

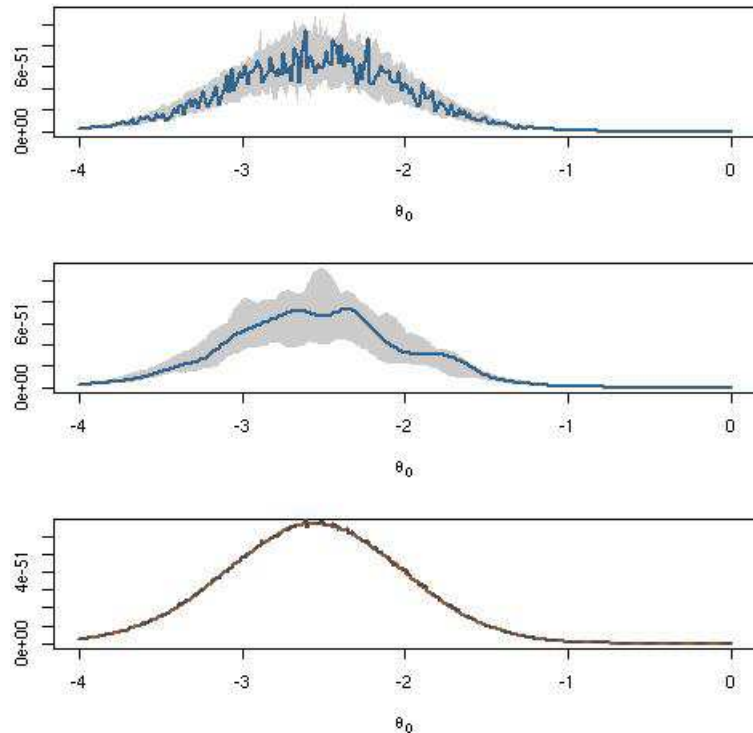
$$\hat{h}_0(\theta_0) = \frac{1}{M} \sum_{m=1}^M \prod_{i=1} \Phi(\theta_0 + \theta_1^m x_n)^{y_i} \Phi(-\theta_0 - \theta_1^m x_n)^{1-y_i} \mathbf{t}_5(\theta_1^m; \mu, \sigma)^{-1},$$

## Stochastic Approximation Importance Sampling Evaluation

- ▶ Plotting this approximation of  $h$  with  $t$  samples simulated for each value of  $\theta_0$ 
  - ▷ The maximization of the represented  $\hat{h}$  function is not to be trusted as an approximation to the maximization of  $h$ .
- ▶ But, if we use the same  $t$  sample for all values of  $\theta_0$ 
  - ▷ We obtain a much smoother function
- ▶ We use importance sampling based on a *single* sample of  $Z_i$ 's
  - ▷ Simulated from an importance function  $g(z)$  for *all* values of  $x$
  - ▷ Estimate  $h$  with

$$\hat{h}_m(x) = \frac{1}{m} \sum_{i=1}^m \frac{f(z_i|x)}{g(z_i)} H(x, z_i).$$

## Stochastic Approximation Importance Sampling Likelihood Representation



- ▶ Top: 100 runs, different samples
- ▶ Middle: 100 runs, same sample
- ▶ Bottom: averages over 100 runs

- ▶ The averages over 100 runs are the same - but we will not do 100 runs
- ▶ R **code**: Run `pimax(25)` from `mcs`

## Stochastic Approximation Comments

- ▶ This approach is not absolutely fool-proof
  - ▷ The precision of  $\hat{h}_m(x)$  has no reason to be independent of  $x$
  - ▷ The number  $m$  of simulations has to reflect the most varying case.
  
- ▶ As in every importance sampling experiment
  - ▷ The choice of the candidate  $g$  is influential
  - ▷ In obtaining a good (or a disastrous) approximation of  $h(x)$ .
  
- ▶ Checking for the finite variance of the ratio  $f(z_i|x)H(x, z_i)/g(z_i)$ 
  - ▷ Is a minimal requirement in the choice of  $g$



## Missing-Data Models and Demarginalization

### Introduction

- ▶ Missing data models are special cases of the representation  $h(x) = \mathbb{E}[H(x, Z)]$
- ▶ These are models where the density of the observations can be expressed as

$$g(x|\theta) = \int_{\mathcal{Z}} f(x, z|\theta) \, dz .$$

- ▶ This representation occurs in many statistical settings
  - ▷ Censoring models and mixtures
  - ▷ Latent variable models (tobit, probit, arch, stochastic volatility, etc.)
  - ▷ Genetics: Missing SNP calls

## Missing-Data Models and Demarginalization

### Mixture Model

Example: Normal mixture model as a missing-data model

- ▶ Start with a sample  $(x_1, \dots, x_n)$
- ▶ Introduce a vector  $(z_1, \dots, z_n) \in \{1, 2\}^n$  such that

$$P_\theta(Z_i = 1) = 1 - P_\theta(Z_i = 2) = 1/4, \quad X_i | Z_i = z \sim \mathcal{N}(\mu_z, 1),$$

- ▶ The (observed) likelihood is then obtained as  $\mathbb{E}[H(\mathbf{x}, \mathbf{Z})]$  for

$$H(\mathbf{x}, \mathbf{z}) \propto \prod_{i; z_i=1} \frac{1}{4} \exp \{-(x_i - \mu_1)^2/2\} \prod_{i; z_i=2} \frac{3}{4} \exp \{-(x_i - \mu_2)^2/2\},$$

- ▶ We recover the mixture model

$$\frac{1}{4} \mathcal{N}(\mu_1, 1) + \frac{3}{4} \mathcal{N}(\mu_2, 1)$$

- ▷ As the marginal distribution of  $X_i$ .

## Missing-Data Models and Demarginalization

### Censored-Data Likelihood

#### Example: Censored-data likelihood

- ▶ Censored data may come from experiments
  - ▷ Where some potential observations are replaced with a lower bound
  - ▷ Because they take too long to observe.
- ▶ Suppose that we observe  $Y_1, \dots, Y_m$ , iid, from  $f(y - \theta)$ 
  - ▷ And the  $(n - m)$  remaining  $(Y_{m+1}, \dots, Y_n)$  are censored at the threshold  $a$ .
- ▶ The corresponding likelihood function is

$$L(\theta|\mathbf{y}) = [1 - F(a - \theta)]^{n-m} \prod_{i=1}^m f(y_i - \theta),$$

- ▷  $F$  is the cdf associated with  $f$

## Missing-Data Models and Demarginalization

### Recovering the Observed Data Likelihood

- If we had observed the last  $n - m$  values
  - ▷ Say  $\mathbf{z} = (z_{m+1}, \dots, z_n)$ , with  $z_i \geq a$  ( $i = m + 1, \dots, n$ ),
  - ▷ We could have constructed the (complete data) likelihood

$$L^c(\theta|\mathbf{y}, \mathbf{z}) = \prod_{i=1}^m f(y_i - \theta) \prod_{i=m+1}^n f(z_i - \theta).$$

- Note that

$$L(\theta|\mathbf{y}) = \mathbb{E}[L^c(\theta|\mathbf{y}, \mathbf{Z})] = \int_{\mathcal{Z}} L^c(\theta|\mathbf{y}, \mathbf{z}) k(\mathbf{z}|\mathbf{y}, \theta) \, d\mathbf{z},$$

- ▷ Where  $k(\mathbf{z}|\mathbf{y}, \theta)$  is the density of the missing data
- ▷ Conditional on the observed data
- ▷ The product of the  $f(z_i - \theta)/[1 - F(a - \theta)]$ 's
- ▷  $f(z - \theta)$  restricted to  $(a, +\infty)$ .

## Missing-Data Models and Demarginalization Comments

- When we have the relationship

$$g(x|\theta) = \int_{\mathbf{Z}} f(x, z|\theta) \, dz .$$

- ▷  $\mathbf{Z}$  merely serves to simplify calculations
  - ▷ it does not necessarily have a specific meaning
- We have the **complete-data likelihood**  $L^c(\theta|\mathbf{x}, \mathbf{z})) = f(\mathbf{x}, \mathbf{z}|\theta)$ 
  - ▷ The likelihood we would obtain
  - ▷ Were we to observe  $(\mathbf{x}, \mathbf{z})$ , the **complete data**

- REMEMBER:

$$g(x|\theta) = \int_{\mathbf{Z}} f(x, z|\theta) \, dz .$$

## The EM Algorithm

### Introduction

- ▶ The EM algorithm is a deterministic optimization technique
  - ▷ Dempster, Laird and Rubin 1977
- ▶ Takes advantage of the missing data representation
  - ▷ Builds a sequence of easier maximization problems
  - ▷ Whose limit is the answer to the original problem
- ▶ We assume that we observe  $X_1, \dots, X_n \sim g(\mathbf{x}|\theta)$  that satisfies

$$g(\mathbf{x}|\theta) = \int_{\mathcal{Z}} f(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z},$$

- ▷ And we want to compute  $\hat{\theta} = \arg \max L(\theta|\mathbf{x}) = \arg \max g(\mathbf{x}|\theta)$ .

## The EM Algorithm First Details

- ▶ With the relationship  $g(\mathbf{x}|\theta) = \int_{\mathbf{Z}} f(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z}$ ,
  - ▷  $(\mathbf{X}, \mathbf{Z}) \sim f(\mathbf{x}, \mathbf{z}|\theta)$

- ▶ The conditional **distribution of the missing data  $\mathbf{Z}$** 
  - ▷ Given the observed data  $\mathbf{x}$  is

$$k(\mathbf{z}|\theta, \mathbf{x}) = f(\mathbf{x}, \mathbf{z}|\theta) / g(\mathbf{x}|\theta).$$

- ▶ Taking the logarithm of this expression leads to the following relationship

$$\underbrace{\log L(\theta|\mathbf{x})}_{\text{Obs. Data}} = \underbrace{\mathbb{E}_{\theta_0}[\log L^c(\theta|\mathbf{x}, \mathbf{Z})]}_{\text{Complete Data}} - \underbrace{\mathbb{E}_{\theta_0}[\log k(\mathbf{Z}|\theta, \mathbf{x})]}_{\text{Missing Data}},$$

- ▶ Where the expectation is with respect to  $k(\mathbf{z}|\theta_0, \mathbf{x})$ .
- ▶ In maximizing  $\log L(\theta|\mathbf{x})$ , we can ignore the last term

## The EM Algorithm Iterations

- Denoting

$$Q(\theta|\theta_0, \mathbf{x}) = \mathbb{E}_{\theta_0}[\log L^c(\theta|\mathbf{x}, \mathbf{Z})],$$

- EM algorithm indeed proceeds by maximizing  $Q(\theta|\theta_0, \mathbf{x})$  at each iteration

- ▷ If  $\hat{\theta}_{(1)} = \operatorname{argmax} Q(\theta|\theta_0, \mathbf{x})$ ,  $\hat{\theta}_{(0)} \rightarrow \hat{\theta}_{(1)}$

- Sequence of estimators  $\{\hat{\theta}_{(j)}\}$ , where

$$\hat{\theta}_{(j)} = \operatorname{argmax} Q(\theta|\hat{\theta}_{(j-1)})$$

- This iterative scheme

- ▷ Contains both an expectation step
  - ▷ And a maximization step
  - ▷ Giving the algorithm its name.



## The EM Algorithm

### The Algorithm

Pick a starting value  $\hat{\theta}_{(0)}$  and set  $m = 0$ .

Repeat

1. Compute (*the E-step*)

$$Q(\theta|\hat{\theta}_{(m)}, \mathbf{x}) = \mathbb{E}_{\hat{\theta}_{(m)}}[\log L^c(\theta|\mathbf{x}, \mathbf{Z})],$$

where the expectation is with respect to  $k(\mathbf{z}|\hat{\theta}_{(m)}, \mathbf{x})$ .

2. Maximize  $Q(\theta|\hat{\theta}_{(m)}, \mathbf{x})$  in  $\theta$  and take (*the M-step*)

$$\hat{\theta}_{(m+1)} = \arg \max_{\theta} Q(\theta|\hat{\theta}_{(m)}, \mathbf{x})$$

and set  $m = m + 1$

until a fixed point is reached; i.e.,  $\hat{\theta}_{(m+1)} = \hat{\theta}_{(m)}$ .fixed point

## The EM Algorithm Properties

- Jensen's inequality  $\Rightarrow$  The likelihood increases at each step of the EM algorithm

$$L(\hat{\theta}_{(j+1)}|\mathbf{x}) \geq L(\hat{\theta}_{(j)}|\mathbf{x}),$$

▷ Equality holding if and only if  $Q(\hat{\theta}_{(j+1)}|\hat{\theta}_{(j)}, \mathbf{x}) = Q(\hat{\theta}_{(j)}|\hat{\theta}_{(j)}, \mathbf{x})$ .

- Every limit point of an EM sequence  $\{\hat{\theta}_{(j)}\}$  is a stationary point of  $L(\theta|\mathbf{x})$ 
  - ▷ Not necessarily the maximum likelihood estimator
  - ▷ In practice, we run EM several times with different starting points.

- Implementing the EM algorithm thus means being able to

- (a) Compute the function  $Q(\theta'|\theta, \mathbf{x})$
- (b) Maximize this function.

## The EM Algorithm

### Censored Data Example

- The complete-data likelihood is

$$L^c(\theta|\mathbf{y}, \mathbf{z}) \propto \prod_{i=1}^m \exp\{-(y_i - \theta)^2/2\} \prod_{i=m+1}^n \exp\{-(z_i - \theta)^2/2\},$$

- With expected complete-data log-likelihood

$$Q(\theta|\theta_0, \mathbf{y}) = -\frac{1}{2} \sum_{i=1}^m (y_i - \theta)^2 - \frac{1}{2} \sum_{i=m+1}^n \mathbb{E}_{\theta_0}[(Z_i - \theta)^2],$$

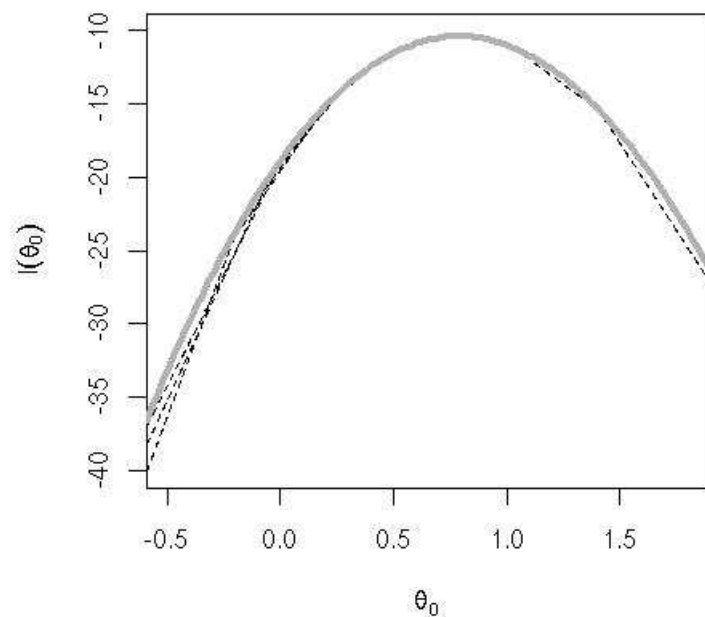
▷ the  $Z_i$  are distributed from a normal  $\mathcal{N}(\theta, 1)$  distribution truncated at  $a$ .

- M-step (differentiating  $Q(\theta|\theta_0, \mathbf{y})$  in  $\theta$  and setting it equal to 0 gives

$$\hat{\theta} = \frac{m\bar{y} + (n - m)\mathbb{E}_{\theta'}[Z_1]}{n}.$$

▷ With  $\mathbb{E}_{\theta}[Z_1] = \theta + \frac{\varphi(a-\theta)}{1-\Phi(a-\theta)},$

## The EM Algorithm Censored Data MLEs



► EM sequence

$$\hat{\theta}^{(j+1)} = \frac{m}{n}\bar{y} + \frac{n-m}{n} \left[ \hat{\theta}^{(j)} + \frac{\varphi(a - \hat{\theta}^{(j)})}{1 - \Phi(a - \hat{\theta}^{(j)})} \right]$$

► Climbing the Likelihood

► R code

## The EM Algorithm

### Normal Mixture

#### ► Normal Mixture Bimodal Likelihood

$$Q(\theta'|\theta, \mathbf{x}) = -\frac{1}{2} \sum_{i=1}^n \mathbb{E}_{\theta} [Z_i(x_i - \mu_1)^2 + (1 - Z_i)(x_i - \mu_2)^2 | \mathbf{x}] .$$

Solving the M-step then provides the closed-form expressions

$$\mu'_1 = \mathbb{E}_{\theta} \left[ \sum_{i=1}^n Z_i x_i | \mathbf{x} \right] / \mathbb{E}_{\theta} \left[ \sum_{i=1}^n Z_i | \mathbf{x} \right]$$

and

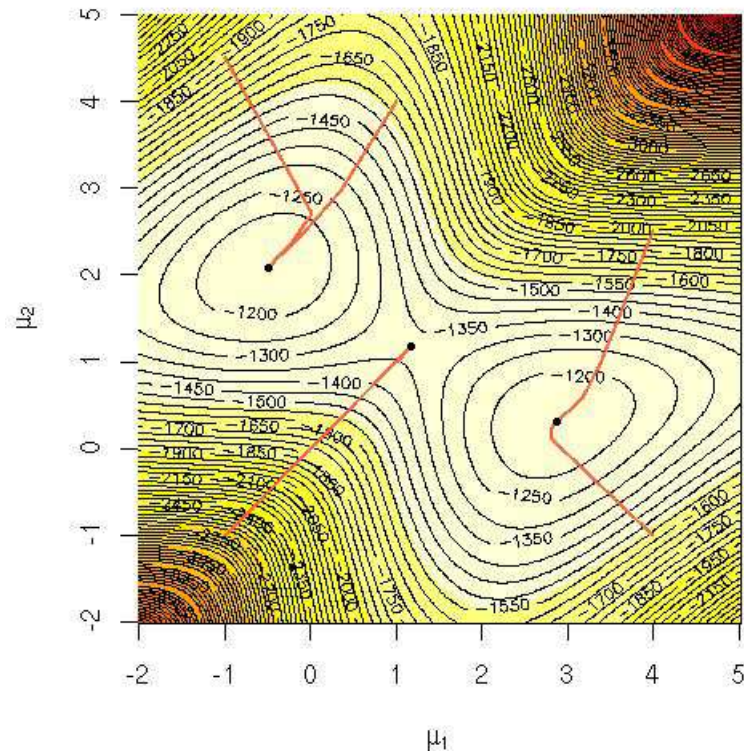
$$\mu'_2 = \mathbb{E}_{\theta} \left[ \sum_{i=1}^n (1 - Z_i) x_i | \mathbf{x} \right] / \mathbb{E}_{\theta} \left[ \sum_{i=1}^n (1 - Z_i) | \mathbf{x} \right] .$$

Since

$$\mathbb{E}_{\theta} [Z_i | \mathbf{x}] = \frac{\varphi(x_i - \mu_1)}{\varphi(x_i - \mu_1) + 3\varphi(x_i - \mu_2)} ,$$

## The EM Algorithm

### Normal Mixture MLEs



- ▶ EM five times with various starting points
- ▶ Two out of five sequences  $\rightarrow$  higher mode
- ▶ Others  $\rightarrow$  lower mode

## Monte Carlo EM Introduction

- If computation  $Q(\theta|\theta_0, \mathbf{x})$  is difficult, can use Monte Carlo
- For  $\mathbf{Z}_1, \dots, \mathbf{Z}_T \sim k(\mathbf{z}|\mathbf{x}, \hat{\theta}_{(m)})$ , maximize

$$\hat{Q}(\theta|\theta_0, \mathbf{x}) = \frac{1}{T} \sum_{i=1}^T \log L^c(\theta|\mathbf{x}, \mathbf{z}_i)$$

- Better: Use importance sampling

▷ Since

$$\arg \max_{\theta} L(\theta|\mathbf{x}) = \arg \max_{\theta} \log \frac{g(\mathbf{x}|\theta)}{g(\mathbf{x}|\theta_{(0)})} = \arg \max_{\theta} \log \mathbb{E}_{\theta_{(0)}} \left[ \frac{f(\mathbf{x}, \mathbf{z}|\theta)}{f(\mathbf{x}, \mathbf{z}|\theta_{(0)})} \middle| \mathbf{x} \right],$$

▷ Use the approximation to the log-likelihood

$$\log L(\theta|\mathbf{x}) \approx \frac{1}{T} \sum_{i=1}^T \frac{L^c(\theta|\mathbf{x}, \mathbf{z}_i)}{L^c(\theta_{(0)}|\mathbf{x}, \mathbf{z}_i)},$$

## Monte Carlo EM Genetics Data

Example: Genetic linkage.

- ▶ A classic example of the EM algorithm
- ▶ Observations  $(x_1, x_2, x_3, x_4)$  are gathered from the multinomial distribution

$$\mathcal{M} \left( n; \frac{1}{2} + \frac{\theta}{4}, \frac{1}{4}(1 - \theta), \frac{1}{4}(1 - \theta), \frac{\theta}{4} \right).$$

- ▶ Estimation is easier if the  $x_1$  cell is split into two cells
  - ▷ We create the augmented model

$$(z_1, z_2, x_2, x_3, x_4) \sim \mathcal{M} \left( n; \frac{1}{2}, \frac{\theta}{4}, \frac{1}{4}(1 - \theta), \frac{1}{4}(1 - \theta), \frac{\theta}{4} \right)$$

with  $x_1 = z_1 + z_2$ .

- ▷ Complete-data likelihood:  $\theta^{z_2+x_4}(1 - \theta)^{x_2+x_3}$
- ▷ Observed-data likelihood:  $(2 + \theta)^{x_1}\theta^{x_4}(1 - \theta)^{x_2+x_3}$



## Monte Carlo EM Genetics Linkage Calculations

- The expected complete log-likelihood function is

$$\mathbb{E}_{\theta_0}[(Z_2 + x_4) \log \theta + (x_2 + x_3) \log(1 - \theta)] = \left( \frac{\theta_0}{2 + \theta_0} x_1 + x_4 \right) \log \theta + (x_2 + x_3) \log(1 - \theta),$$

- ▷ which can easily be maximized in  $\theta$ , leading to the EM step

$$\hat{\theta}_1 = \left\{ \frac{\theta_0 x_1}{2 + \theta_0} \right\} / \left\{ \frac{\theta_0 x_1}{2 + \theta_0} + x_2 + x_3 + x_4 \right\}.$$

- Monte Carlo EM: Replace the expectation with

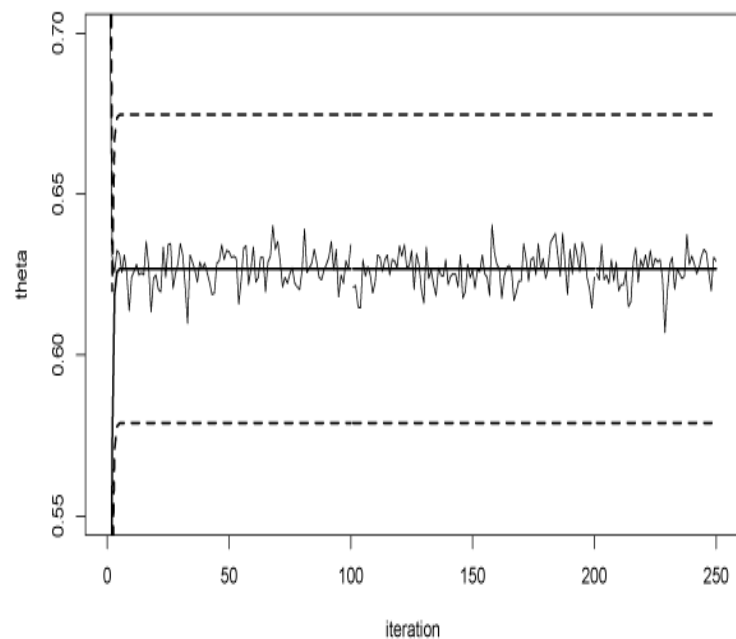
$$\triangleright \bar{z}_m = \frac{1}{m} \sum_{i=1}^m z_i, \quad z_i \sim \mathcal{B}(x_1, \theta_0 / (2 + \theta_0))$$

- The MCEM step would then be

$$\hat{\hat{\theta}}_1 = \frac{\bar{z}_m}{\bar{z}_m + x_2 + x_3 + x_4},$$

which converges to  $\hat{\theta}_1$  as  $m$  grows to infinity.

## Monte Carlo EM Genetics Linkage MLEs



- Note variation in MCEM sequence
- Can control with  $\uparrow$  simulations
- R `code`

## Monte Carlo EM Random effect logit model

### Example: Random effect logit model

► Random effect logit model,

▷  $y_{ij}$  is distributed conditionally on one covariate  $x_{ij}$  as a logit model

$$P(y_{ij} = 1 | x_{ij}, u_i, \beta) = \frac{\exp \{ \beta x_{ij} + u_i \}}{1 + \exp \{ \beta x_{ij} + u_i \}},$$

▷  $u_i \sim \mathcal{N}(0, \sigma^2)$  is an unobserved random effect.

▷  $(U_1, \dots, U_n)$  therefore corresponds to the missing data  $\mathbf{Z}$

## Monte Carlo EM

### Random effect logit model likelihood

- For the complete data likelihood with  $\theta = (\beta, \sigma)$ ,

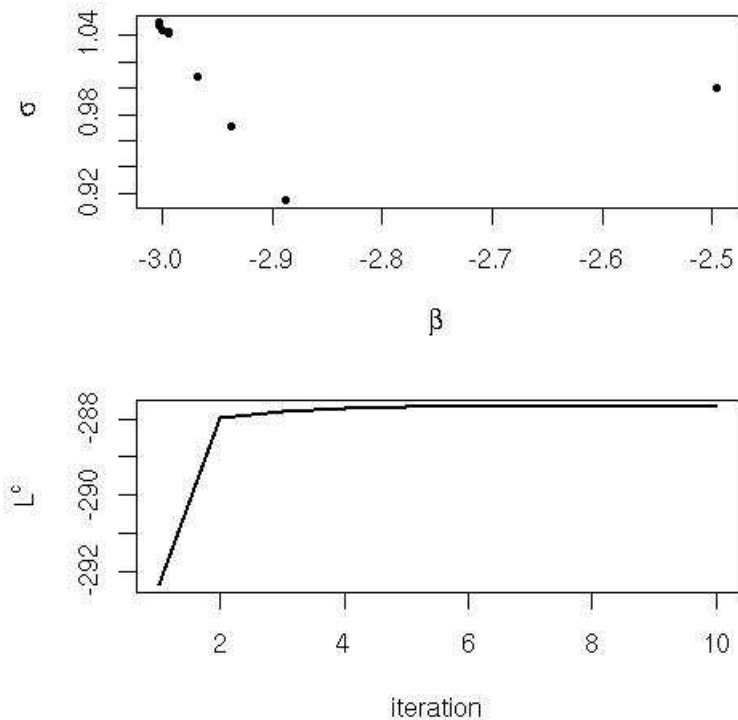
$$\begin{aligned} Q(\theta'|\theta, \mathbf{x}, \mathbf{y}) = & \sum_{i,j} y_{ij} \mathbb{E}[\beta' x_{ij} + U_i | \beta, \sigma, \mathbf{x}, \mathbf{y}] \\ & - \sum_{i,j} \mathbb{E}[\log 1 + \exp\{\beta' x_{ij} + U_i\} | \beta, \sigma, \mathbf{x}, \mathbf{y}] \\ & - \sum_i \mathbb{E}[U_i^2 | \beta, \sigma, \mathbf{x}, \mathbf{y}] / 2\sigma'^2 - n \log \sigma', \end{aligned}$$

▷ it is impossible to compute the expectations in  $U_i$ .

- Were those available, the M-step would be difficult but feasible
- MCEM: Simulate the  $U_i$ 's conditional on  $\beta, \sigma, \mathbf{x}, \mathbf{y}$  from

$$\pi(u_i | \beta, \sigma, \mathbf{x}, \mathbf{y}) \propto \frac{\exp \left\{ \sum_j y_{ij} u_i - u_i^2 / 2\sigma^2 \right\}}{\prod_j [1 + \exp \{ \beta x_{ij} + u_i \}]}$$

## Monte Carlo EM Random effect logit MLEs



- ▶ MCEM sequence
  - ▷ Increases the number of Monte Carlo steps at each iteration
- ▶ MCEM algorithm
  - ▷ Does not have EM monotonicity property

- ▶ Top: Sequence of  $\beta$ 's from the MCEM algorithm
- ▶ Bottom: Sequence of completed likelihoods

## Chapter 6: Metropolis–Hastings Algorithms

*“How absurdly simple!”, I cried.*

*“Quite so!”, said he, a little nettled. “Every problem becomes very childish when once it is explained to you.”*

**Arthur Conan Doyle**

*The Adventure of the Dancing Men*

### **This Chapter**

- ▶ The first of a of two on simulation methods based on *Markov chains*
- ▶ The Metropolis–Hastings algorithm is one of the most general MCMC algorithms
  - ▷ And one of the simplest.
- ▶ There is a quick refresher on Markov chains, just the basics.
- ▶ We focus on the most common versions of the Metropolis–Hastings algorithm.
- ▶ We also look at calibration of the algorithm via its acceptance rate

## Metropolis–Hastings Algorithms

### Introduction

- ▶ We now make a fundamental shift in the choice of our simulation strategy.
  - ▷ Up to now we have typically generated *iid* variables
  - ▷ The Metropolis–Hastings algorithm generates *correlated* variables
    - ▷ From a Markov chain
- ▶ The use of Markov chains broadens our scope of applications
  - ▷ The requirements on the target  $f$  are quite minimal
  - ▷ Efficient decompositions of high-dimensional problems
    - ▷ Into a sequence of smaller problems.
- ▶ This has been part of a Paradigm Shift in Statistics

## Metropolis–Hastings Algorithms

### A Peek at Markov Chain Theory

- ▶ A minimalist refresher on Markov chains
  - ▶ Basically to define terms
  - ▶ See Robert and Casella (2004, Chapter 6) for more of the story
- 
- ▶ A **Markov chain**  $\{X^{(t)}\}$  is a sequence of dependent random variables
$$X^{(0)}, X^{(1)}, X^{(2)}, \dots, X^{(t)}, \dots$$
where the probability distribution of  $X^{(t)}$  **depends only on**  $X^{(t-1)}$ .
  - ▶ The conditional distribution of  $X^{(t)} | X^{(t-1)}$  is a **transition kernel**  $K$ ,
$$X^{(t+1)} | X^{(0)}, X^{(1)}, X^{(2)}, \dots, X^{(t)} \sim K(X^{(t)}, X^{(t+1)}).$$



## Markov Chains Basics

- For example, a simple *random walk* Markov chain satisfies

$$X^{(t+1)} = X^{(t)} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, 1),$$

- ▷ The Markov kernel  $K(X^{(t)}, X^{(t+1)})$  corresponds to a  $\mathcal{N}(X^{(t)}, 1)$  density.
- Markov chain Monte Carlo (MCMC) Markov chains typically have a very strong stability property.
- They have a *stationary probability distribution*
  - ▷ A *probability distribution*  $f$  such that if  $X^{(t)} \sim f$ , then  $X^{(t+1)} \sim f$ , so we have the equation

$$\int_{\mathcal{X}} K(x, y) f(x) dx = f(y).$$

## Markov Chains Properties

- ▶ MCMC Markov chains are also *irreducible*, or else they are useless
  - ▷ The kernel  $K$  allows for free moves all over the state-space
  - ▷ For any  $X^{(0)}$ , the sequence  $\{X^{(t)}\}$  has a positive probability of eventually reaching any region of the state-space
- ▶ MCMC Markov chains are also *recurrent*, or else they are useless
  - ▷ They will return to any arbitrary nonnegligible set an infinite number of times

## Markov Chains AR(1) Process

- ▶ AR(1) models provide a simple illustration of continuous Markov chains

- ▶ Here

$$X_n = \theta X_{n-1} + \varepsilon_n, \quad \theta \in \mathbb{R},$$

with  $\varepsilon_n \sim N(0, \sigma^2)$

- ▶ If the  $\varepsilon_n$ 's are independent

▷  $X_n$  is independent from  $X_{n-2}, X_{n-3}, \dots$  conditionally on  $X_{n-1}$ .

- ▶ The stationary distribution  $\phi(x|\mu, \tau^2)$  is

$$\mathcal{N}\left(0, \frac{\sigma^2}{1 - \theta^2}\right),$$

▷ which requires  $|\theta| < 1$ .

## Markov Chains Statistical Language

- We associate the probabilistic language of Markov chains
  - ▷ With the statistical language of data analysis.

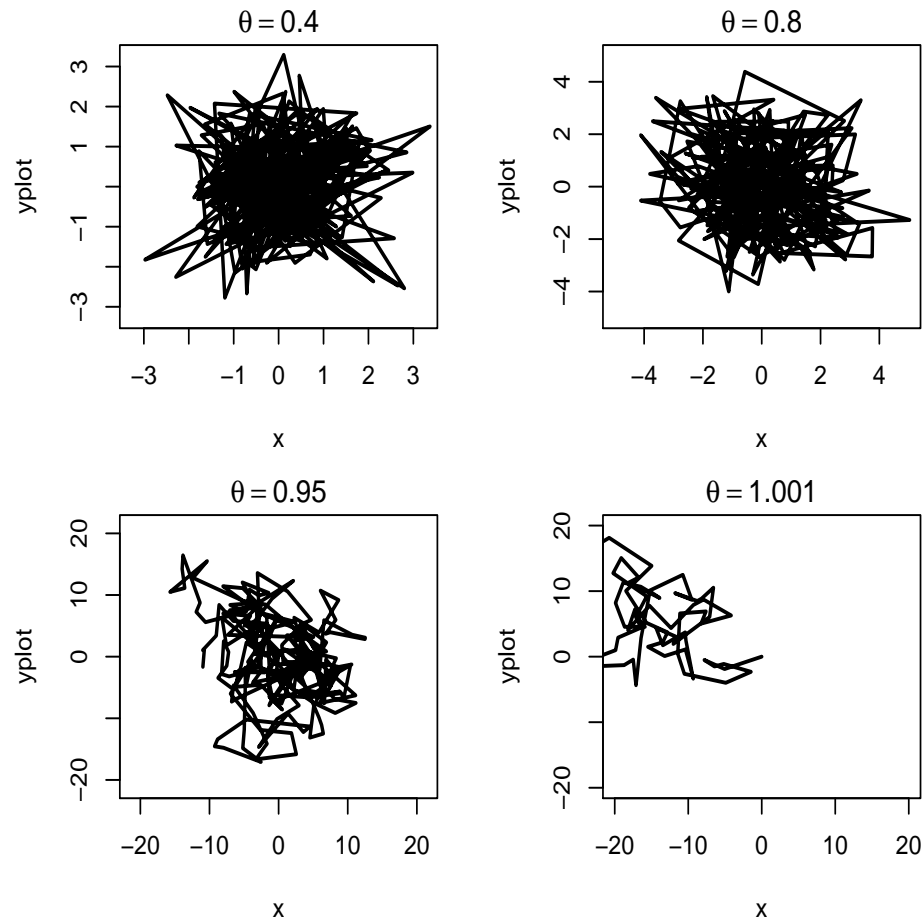
| Statistics            | Markov Chain                             |
|-----------------------|------------------------------------------|
| marginal distribution | $\Leftrightarrow$ invariant distribution |
| proper marginals      | $\Leftrightarrow$ positive recurrent     |

- If the marginals are not proper, or if they do not exist
  - ▷ Then the chain is not positive recurrent.
  - ▷ It is either null recurrent or transient, and both are bad.

## Markov Chains

### Pictures of the AR(1) Process

#### ► AR(1) Recurrent and Transient -Note the Scale



#### ► R code

## Markov Chains Ergodicity

- ▶ In recurrent chains, the stationary distribution is also a *limiting distribution*
- ▶ If  $f$  is the limiting distribution

$$X^{(t)} \rightarrow X \sim f, \text{ for any initial value } X^{(0)}$$

- ▷ This property is also called *ergodicity*
- ▶ For integrable functions  $h$ , the standard average

$$\frac{1}{T} \sum_{t=1}^T h(X^{(t)}) \longrightarrow \mathbb{E}_f[h(X)],$$

- ▷ The Law of Large Numbers
  - ▷ Sometimes called the *Ergodic Theorem*

## Markov Chains In Bayesian Analysis

- ▶ There is one case where convergence never occurs
- ▶ When, in a Bayesian analysis, the posterior distribution is not proper
- ▶ The use of improper priors  $f(x)$  is quite common in complex models,
  - ▷ Sometimes the posterior is proper, and MCMC works (recurrent)
  - ▷ Sometimes the posterior is improper, and MCMC fails (transient)
- ▶ These transient Markov chains may present all the outer signs of stability
  - ▷ More later

## Basic Metropolis–Hastings algorithms

### Introduction

- ▶ The working principle of Markov chain Monte Carlo methods is straightforward
- ▶ Given a target density  $f$ 
  - ▷ We build a Markov kernel  $K$  with stationary distribution  $f$
  - ▷ Then generate a Markov chain  $(X^{(t)}) \rightarrow X \sim f$
  - ▷ Integrals can be approximated by to the Ergodic Theorem
- ▶ The Metropolis–Hastings algorithm is an example of those methods.
  - ▷ Given the target density  $f$ , we simulate from a candidate  $q(y|x)$
  - ▷ Only need that the ratio  $f(y)/q(y|x)$  is known up to a constant



## Basic Metropolis–Hastings algorithms

### A First Metropolis–Hastings Algorithm

Metropolis–Hastings **Given**  $x^{(t)}$ ,

1. **Generate**  $Y_t \sim q(y|x^{(t)})$ .

2. **Take**

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \rho(x^{(t)}, Y_t), \\ x^{(t)} & \text{with probability } 1 - \rho(x^{(t)}, Y_t), \end{cases}$$

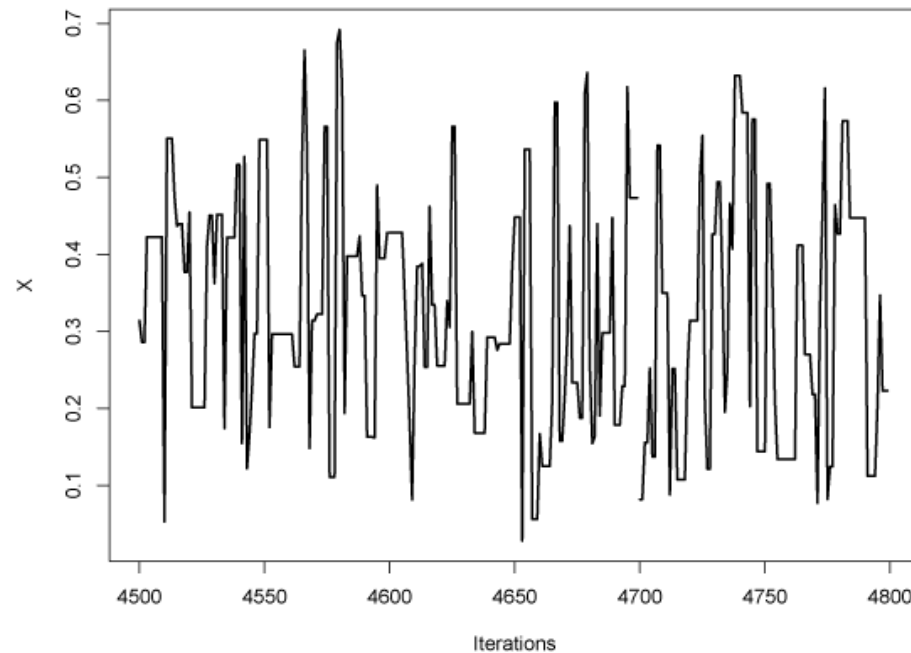
where

$$\rho(x, y) = \min \left\{ \frac{f(y)}{f(x)} \frac{q(x|y)}{q(y|x)}, 1 \right\}.$$

- ▶  $q$  is called the instrumental or proposal or candidate distribution
- ▶  $\rho(x, y)$  is the Metropolis–Hastings acceptance probability
- ▶ Looks like Simulated Annealing - but constant temperature
  - ▷ Metropolis–Hastings explores rather than maximizes

## Basic Metropolis–Hastings algorithms

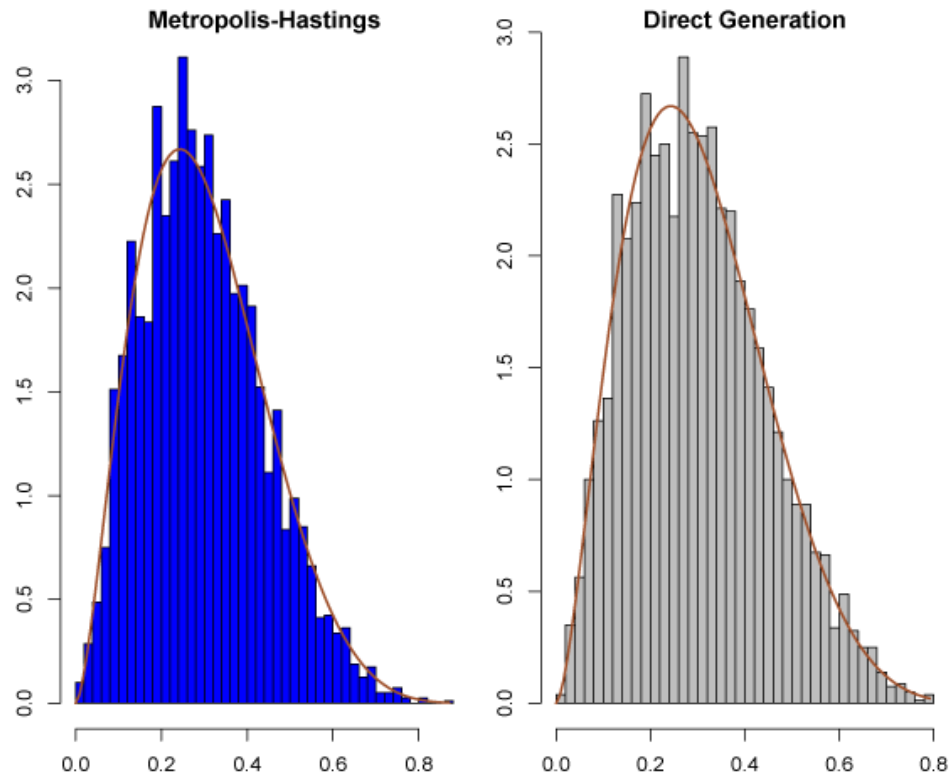
### Generating Beta Random Variables



- ▶ Target density  $f$  is the  $\mathcal{Be}(2.7, 6.3)$
- ▶ Candidate  $q$  is uniform
- ▶ Notice the repeats
- ▶ Repeats must be kept!

## Basic Metropolis–Hastings algorithms

### Comparing Beta densities



- ▶ Comparison with independent sampling
- ▶ Histograms indistinguishable
  - ▷ Moments match
  - ▷ K-S test accepts
- ▶ R [code](#)

## Basic Metropolis–Hastings algorithms

### A Caution

- ▶ The MCMC and exact sampling outcomes look identical, but
  - ▷ Markov chain Monte Carlo sample has correlation, the iid sample does not
  - ▷ This means that the quality of the sample is necessarily degraded
  - ▷ We need more simulations to achieve the same precision
- ▶ This is formalized by the *effective sample size* for Markov chains - later

## Basic Metropolis–Hastings algorithms

### Some Comments

- In the symmetric case  $q(x|y) = q(y|x)$ ,

$$\rho(x_t, y_t) = \min \left\{ \frac{f(y_t)}{f(x_t)}, 1 \right\} .$$

- ▷ The acceptance probability is independent of  $q$

- Metropolis–Hastings always accept values of  $y_t$  such that

$$f(y_t)/q(y_t|x^{(t)}) > f(x^{(t)})/q(x^{(t)}|y_t)$$

- Values  $y_t$  that decrease the ratio may also be accepted

- Metropolis–Hastings only depends on the ratios

$$f(y_t)/f(x^{(t)}) \quad \text{and} \quad q(x^{(t)}|y_t)/q(y_t|x^{(t)}) .$$

- ▷ Independent of normalizing constants

Basic Metropolis–Hastings algorithms  
The Independent Metropolis–Hastings algorithm

- The Metropolis–Hastings algorithm allows  $q(y|x)$ 
  - ▷ We can use  $q(y|x) = g(y)$ , a special case

### Independent Metropolis–Hastings

Given  $x^{(t)}$

1. Generate  $Y_t \sim g(y)$ .

2. Take

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \min \left\{ \frac{f(Y_t) g(x^{(t)})}{f(x^{(t)}) g(Y_t)}, 1 \right\} \\ x^{(t)} & \text{otherwise.} \end{cases}$$

## Basic Metropolis–Hastings algorithms

### Properties of the Independent Metropolis–Hastings algorithm

- ▶ Straightforward generalization of the Accept–Reject method
- ▶ Candidates are independent, but still a Markov chain
  - ▷ The Accept–Reject sample is iid, but the Metropolis–Hastings sample is not
  - ▷ The Accept–Reject acceptance step requires calculating  $M$
  - ▷ Metropolis–Hastings is Accept–Reject “for the lazy person”

## Basic Metropolis–Hastings algorithms

### Application of the Independent Metropolis–Hastings algorithm

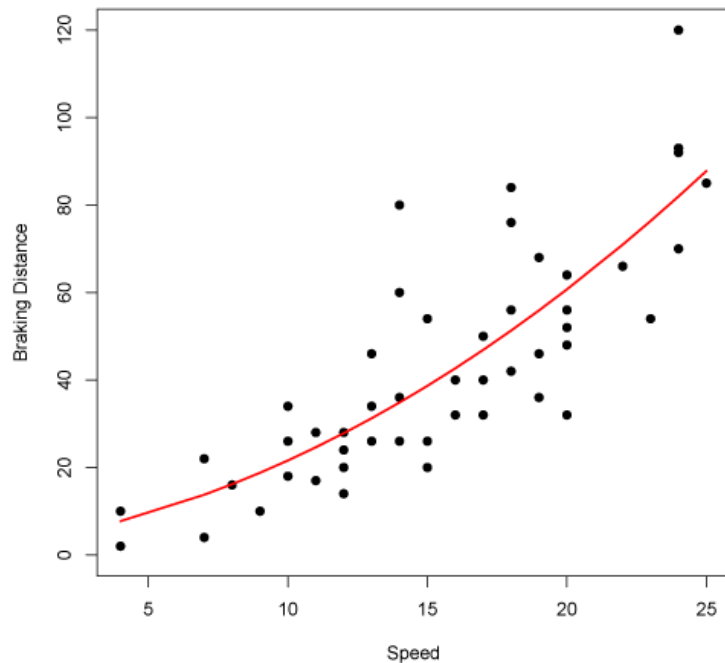
- ▶ We now look at a somewhat more realistic statistical example
  - ▷ Get preliminary parameter estimates from a model
  - ▷ Use an independent proposal with those parameter estimates.
  
- ▶ For example, to simulate from a posterior distribution  $\pi(\theta|x) \propto \pi(\theta)f(x|\theta)$ 
  - ▷ Take a normal or a  $t$  distribution centered at the MLE  $\hat{\theta}$
  - ▷ Covariance matrix equal to the inverse of Fisher's information matrix.



## Independent Metropolis–Hastings algorithm

### Braking Data

- The **cars** dataset relates braking distance ( $y$ ) to speed ( $x$ ) in a sample of cars.



- Model

$$y_{ij} = a + bx_i + cx_i^2 + \varepsilon_{ij}$$

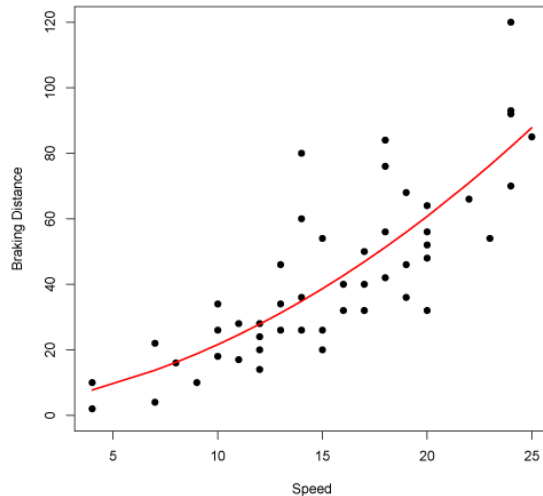
- The likelihood function is

$$\left(\frac{1}{\sigma^2}\right)^{N/2} \exp \left\{ \frac{-1}{2\sigma^2} \sum_{ij} (y_{ij} - a - bx_i - cx_i^2)^2 \right\},$$

$$\text{where } N = \sum_i n_i$$

## Independent Metropolis–Hastings algorithm

### Braking Data Least Squares Fit



► Candidate from Least Squares

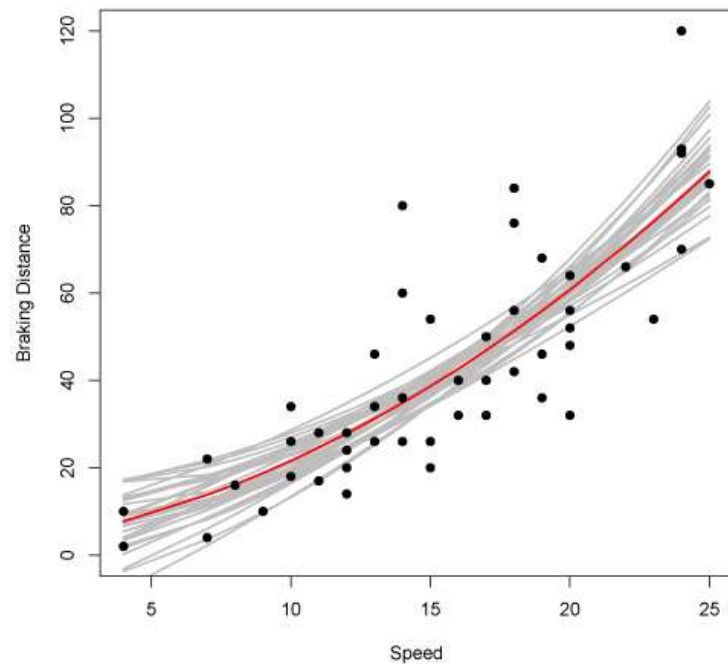
R command: `x2=x^2; summary(lm(y~x+x2))`

Coefficients:

|             | Estimate | Std. Error | t value | Pr(>  t ) |
|-------------|----------|------------|---------|-----------|
| (Intercept) | 2.63328  | 14.80693   | 0.178   | 0.860     |
| x           | 0.88770  | 2.03282    | 0.437   | 0.664     |
| x2          | 0.10068  | 0.06592    | 1.527   | 0.133     |

Residual standard error: 15.17 on 47 degrees of freedom

Independent Metropolis–Hastings algorithm  
Braking Data Metropolis Algorithm



- Candidate: normal centered at the MLEs,

$$a \sim \mathcal{N}(2.63, (14.8)^2),$$

$$b \sim \mathcal{N}(.887, (2.03)^2),$$

$$c \sim \mathcal{N}(.100, (0.065)^2),$$

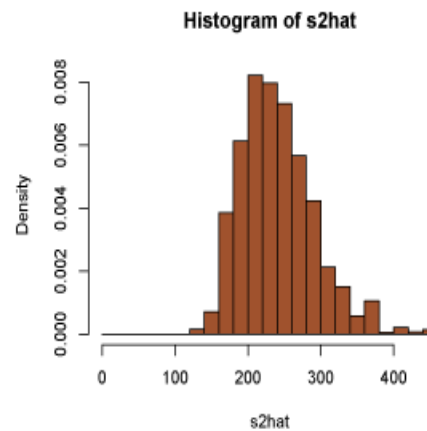
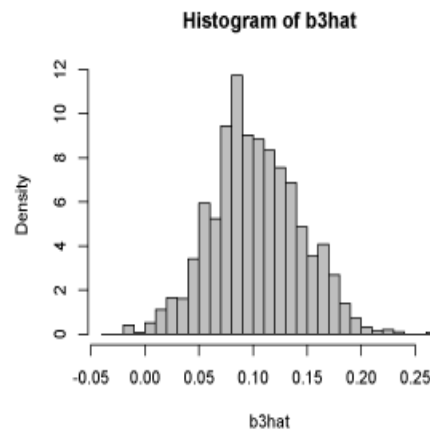
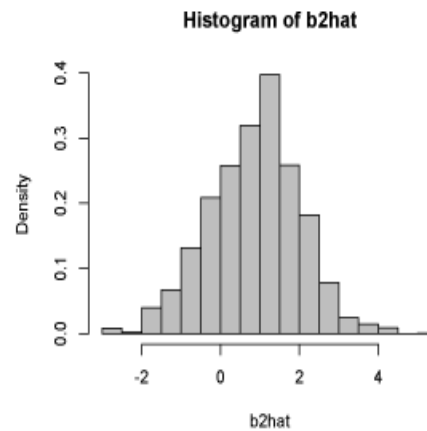
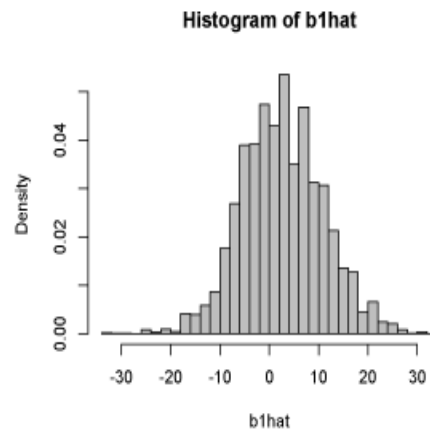
- Inverted gamma

$$\sigma^{-2} \sim \mathcal{G}(n/2, (n - 3)(15.17)^2)$$

- See the variability of the curves associated with the simulation.

## Independent Metropolis–Hastings algorithm

### Braking Data Coefficients

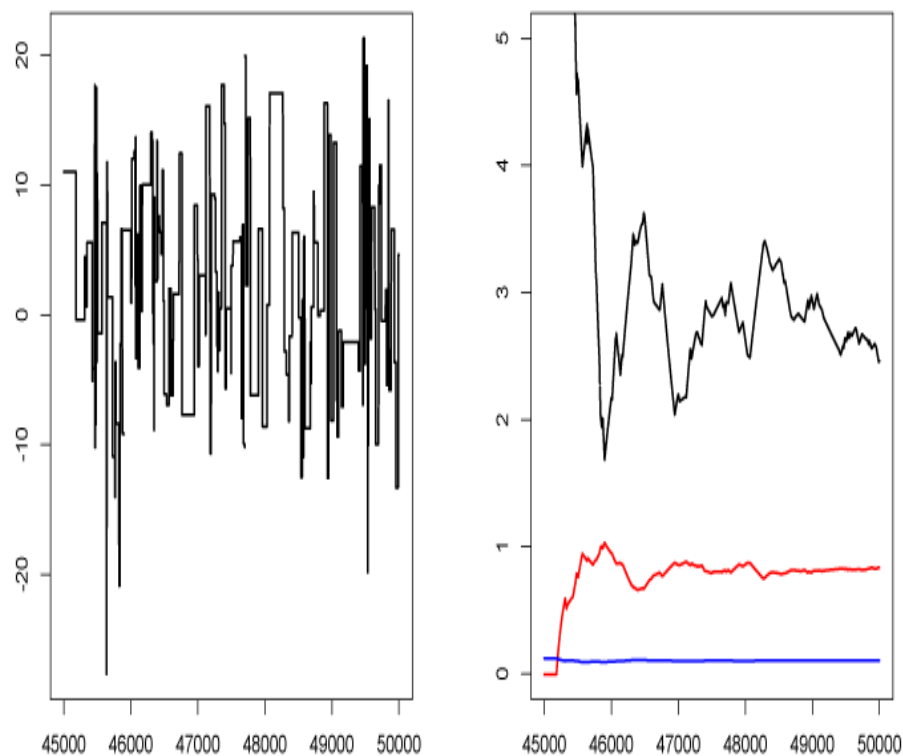


- Distributions of estimates
- Credible intervals
- See the skewness

► Note that these are marginal distributions

## Independent Metropolis–Hastings algorithm

### Braking Data Assessment



- ▶ 50,000 iterations
- ▶ See the repeats
- ▶ Intercept may not have converged

▶ R [code](#)

## Random Walk Metropolis–Hastings Introduction

- ▶ Implementation of independent Metropolis–Hastings can sometimes be difficult
  - ▷ Construction of the proposal may be complicated
  - ▷ They ignore local information
- ▶ An alternative is to gather information stepwise
  - ▷ Exploring the neighborhood of the current value of the chain
- ▶ Can take into account the value previously simulated to generate the next value
  - ▷ Gives a more local exploration of the neighborhood of the current value

## Random Walk Metropolis–Hastings Some Details

- The implementation of this idea is to simulate  $Y_t$  according to

$$Y_t = X^{(t)} + \varepsilon_t,$$

- ▷  $\varepsilon_t$  is a random perturbation
  - ▷ with distribution  $g$ , independent of  $X^{(t)}$
  - ▷ Uniform, normal, etc...
- The proposal density  $q(y|x)$  is now of the form  $g(y - x)$ 
  - ▷ Typically,  $g$  is symmetric around zero, satisfying  $g(-t) = g(t)$
  - ▷ The Markov chain associated with  $q$  is a *random walk*

## Random Walk Metropolis–Hastings The Algorithm

Given  $x^{(t)}$ ,

1. Generate  $Y_t \sim g(y - x^{(t)})$ .

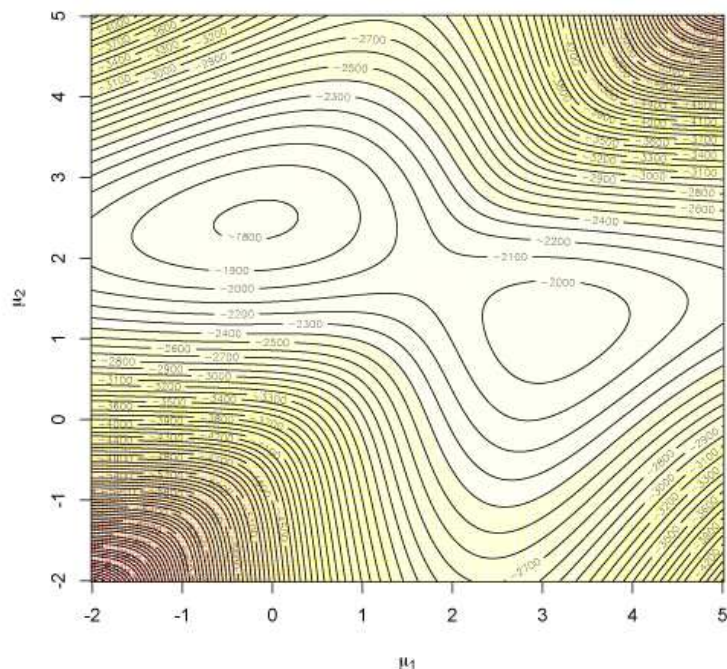
2. Take

$$X^{(t+1)} = \begin{cases} Y_t & \text{with probability } \min \left\{ 1, \frac{f(Y_t)}{f(x^{(t)})} \right\}, \\ x^{(t)} & \text{otherwise.} \end{cases}$$

- ▶ The  $g$  chain is a random walk
  - ▷ Due to the Metropolis–Hastings acceptance step, the  $\{X^{(t)}\}$  chain is not
- ▶ The acceptance probability does not depend on  $g$ 
  - ▷ But different  $g$ s result in different ranges and different acceptance rates
- ▶ Calibrating the scale of the random walk is for good exploration

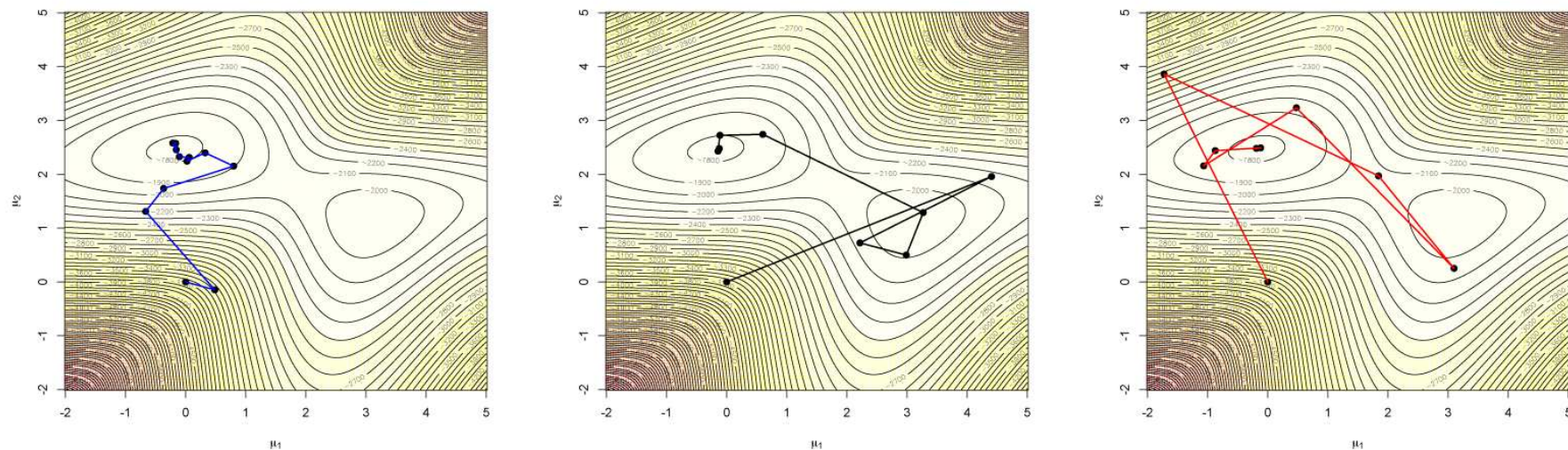


## Random Walk Metropolis–Hastings Normal Mixtures



- ▶ Explore likelihood with random walk
- ▶ Similar to Simulated Annealing
  - ▷ But constant temperature (scale)
- ▶ Multimodal  $\Rightarrow$  Scale is important
  - ▷ Too small  $\Rightarrow$  get stuck
  - ▷ Too big  $\Rightarrow$  miss modes

## Random Walk Metropolis–Hastings Normal Mixtures - Different Scales



► Left → Right: **Scale=1**, Scale=2, **Scale=3**

▷ **Scale=1**: Too small, gets stuck

▷ Scale=2: Just right, finds both modes

▷ **Scale=3**: Too big, misses mode

► R **code**

## Random Walk Metropolis–Hastings Model Selection or Model Choice

- ▶ Random walk Metropolis–Hastings algorithms also apply to discrete targets.
  - ▶ As an illustration, we consider a regression
    - ▷ The **swiss** dataset in R
    - ▷  $y$  = logarithm of the fertility in 47 districts of Switzerland  $\approx 1888$
    - ▷ The covariate matrix  $X$  involves five explanatory variables
- ```
> names(swiss)
[1] "Fertility"  "Agriculture"  "Examination"  "Education"
[5] "Catholic"   "Infant.Mortality"
```
- ▶ Compare the  $2^5 = 32$  models corresponding to all possible subsets of covariates.
    - ▷ If we include squares and twoway interactions
    - ▷  $2^{20} = 1048576$  models, same R code

## Random Walk Metropolis–Hastings Model Selection using Marginals

- Given an ordinary linear regression with  $n$  observations,

$$\mathbf{y}|\beta, \sigma^2, X \sim \mathcal{N}_n(X\beta, \sigma^2 I_n), X \text{ is an } (n, p) \text{ matrix}$$

- The likelihood is

$$\ell(\beta, \sigma^2|\mathbf{y}, X) = (2\pi\sigma^2)^{-n/2} \exp \left[ -\frac{1}{2\sigma^2}(\mathbf{y} - X\beta)^T(\mathbf{y} - X\beta) \right]$$

- Using Zellner's  $g$ -prior, with the constant  $g = n$

$$\beta|\sigma^2, X \sim \mathcal{N}_{k+1}(\tilde{\beta}, n\sigma^2(X^T X)^{-1}) \quad \text{and} \quad \pi(\sigma^2|X) \propto \sigma^{-2}$$

- ▷ The marginal distribution of  $\mathbf{y}$  is a multivariate  $t$  distribution,

$$m(\mathbf{y}|X) \propto \left[ \mathbf{y}' \left( I - \frac{n}{n+1} X(X'X)^{-1} X' \right) \mathbf{y} - \frac{1}{n+1} \tilde{\beta}' X' X \tilde{\beta} \right]^{-n/2}.$$

- Find the model with maximum marginal probability

## Random Walk Metropolis–Hastings Random Walk on Model Space

- To go from  $\gamma^{(t)} \rightarrow \gamma^{(t+1)}$ 
  - ▷ **First** get a candidate  $\gamma^*$

$$\gamma^{(t)} = \begin{pmatrix} 1 \\ 0 \\ \textcolor{blue}{1} \\ 1 \\ 0 \end{pmatrix} \rightarrow \gamma^* = \begin{pmatrix} 1 \\ 0 \\ \textcolor{blue}{0} \\ 1 \\ 0 \end{pmatrix}$$

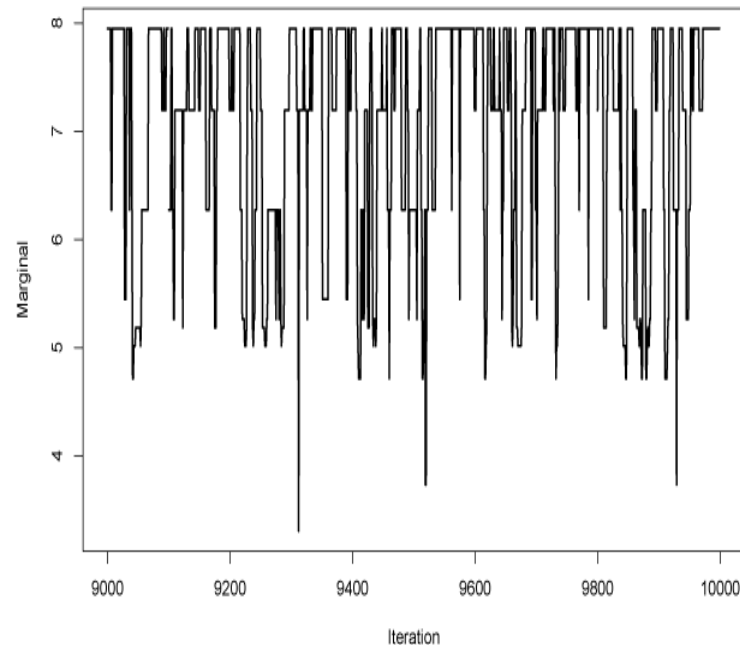
- ▷ **Choose** a component of  $\gamma^{(t)}$  at random, and flip  $1 \rightarrow 0$  or  $0 \rightarrow 1$
- ▷ **Accept** the proposed model  $\gamma^*$  with probability

$$\min \left\{ \frac{m(\mathbf{y}|X, \gamma^*)}{m(\mathbf{y}|X, \gamma^{(t)})}, 1 \right\}$$

- The candidate is symmetric
- Note: This is not the Metropolis–Hastings algorithm in the book - it is simpler

## Random Walk Metropolis–Hastings

### Results from the Random Walk on Model Space



- Last iterations of the MH search
- The chain goes down often

#### ► Top Five Models

Marg.	$\gamma$				
7.95	1	0	1	1	1
7.19	0	0	1	1	1
6.27	1	1	1	1	1
5.44	1	0	1	1	0
5.45	1	0	1	1	0

- Best model excludes  
the variable Examination

$$\triangleright \gamma = (1, 0, 1, 1, 1)$$

- Inclusion rates:

Agri	Exam	Educ	Cath	Inf.Mort
0.661	0.194	1.000	0.904	0.949

## Metropolis–Hastings Algorithms

### Acceptance Rates

- ▶ Infinite number of choices for the candidate  $q$  in a Metropolis–Hastings algorithm
- ▶ Is there and “optimal” choice?
  - ▷ The choice of  $q = f$ , the target distribution? Not practical.
- ▶ A criterion for comparison is the acceptance rate
  - ▷ It can be easily computed with the empirical frequency of acceptance
- ▶ In contrast to the Accept–Reject algorithm
  - ▷ Maximizing the acceptance rate will is not necessarily best
  - ▷ Especially for random walks
- ▶ Also look at autocovariance

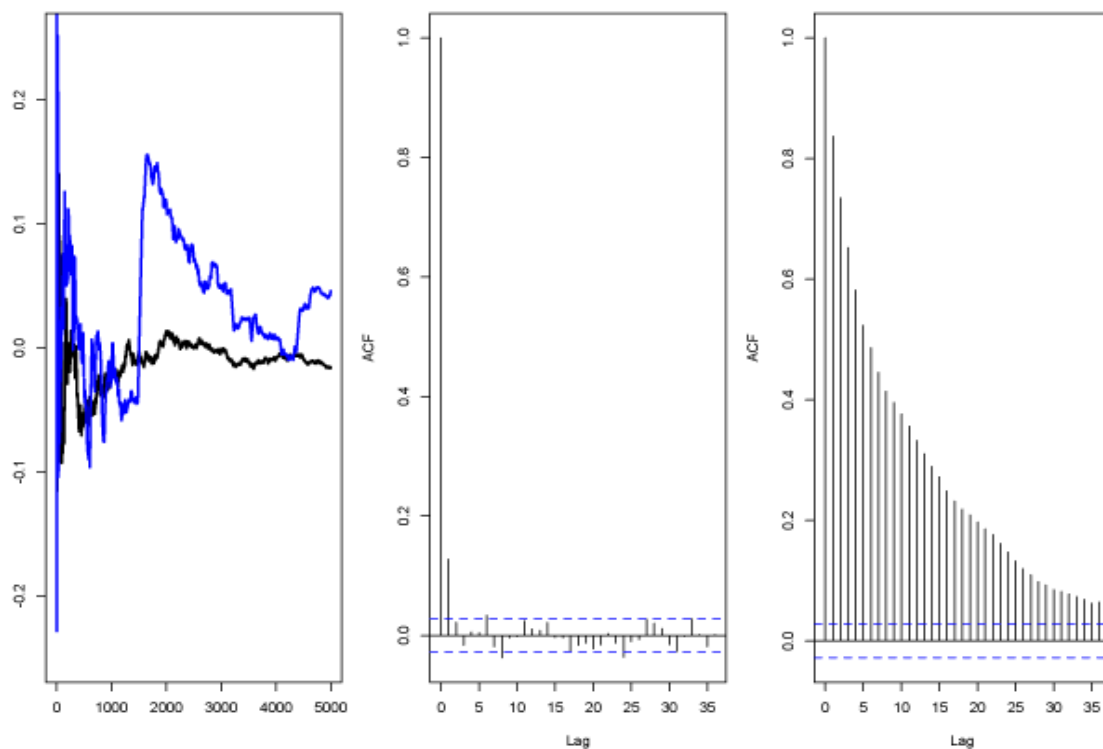
Acceptance Rates  
Normals from Double Exponentials

- ▶ In the Accept–Reject algorithm
  - ▷ To generate a  $\mathcal{N}(0, 1)$  from a double-exponential  $\mathcal{L}(\alpha)$
  - ▷ The choice  $\alpha = 1$  optimizes the acceptance rate
  
- ▶ In an independent Metropolis–Hastings algorithm
  - ▷ We can use the double-exponential as an independent candidate  $q$
  
- ▶ Compare the behavior of Metropolis–Hastings algorithm
  - ▷ When using the  $\mathcal{L}(1)$  candidate or the  $\mathcal{L}(3)$  candidate



## Acceptance Rates

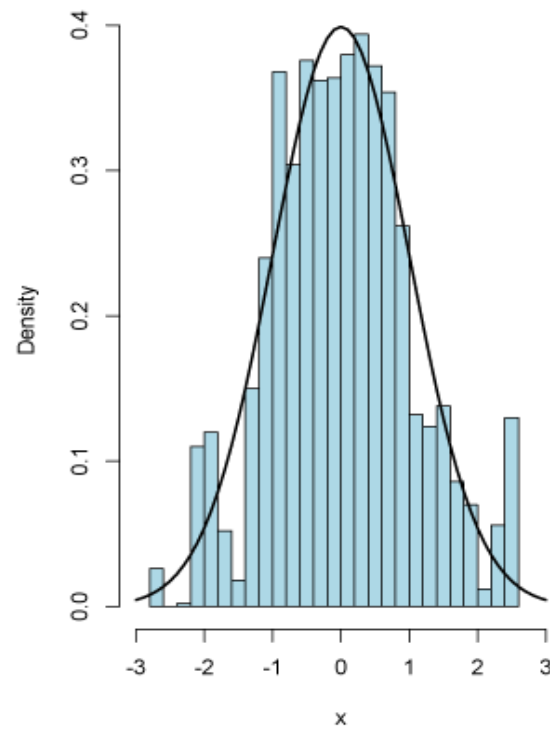
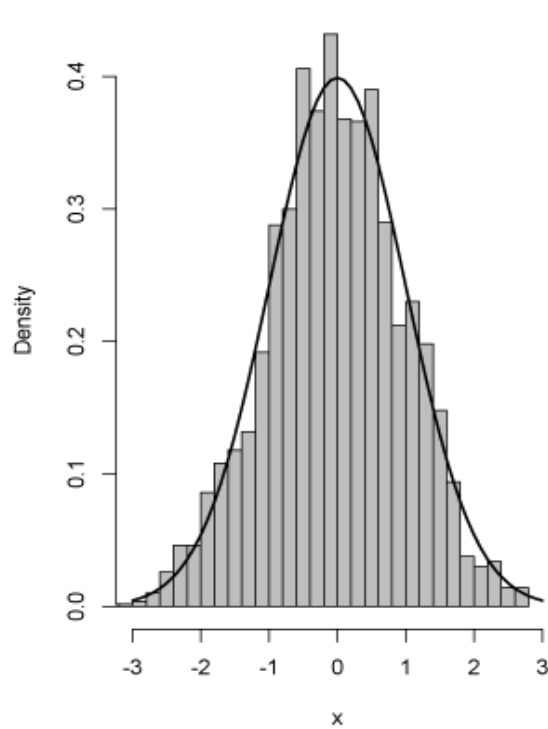
### Normals from Double Exponentials Comparison



- $\mathcal{L}(1)$  (black)
  - ▷ Acceptance Rate = 0.83
- $\mathcal{L}(3)$  (blue)
  - ▷ Acceptance Rate = 0.47
- $\mathcal{L}(3)$  has terrible acf (right)
- $\mathcal{L}(3)$  has not converged

## Acceptance Rates

### Normals from Double Exponentials Histograms



- $\mathcal{L}(1)$  has converged (gray)
- $\mathcal{L}(3)$  not yet there (blue)
- R [code](#)

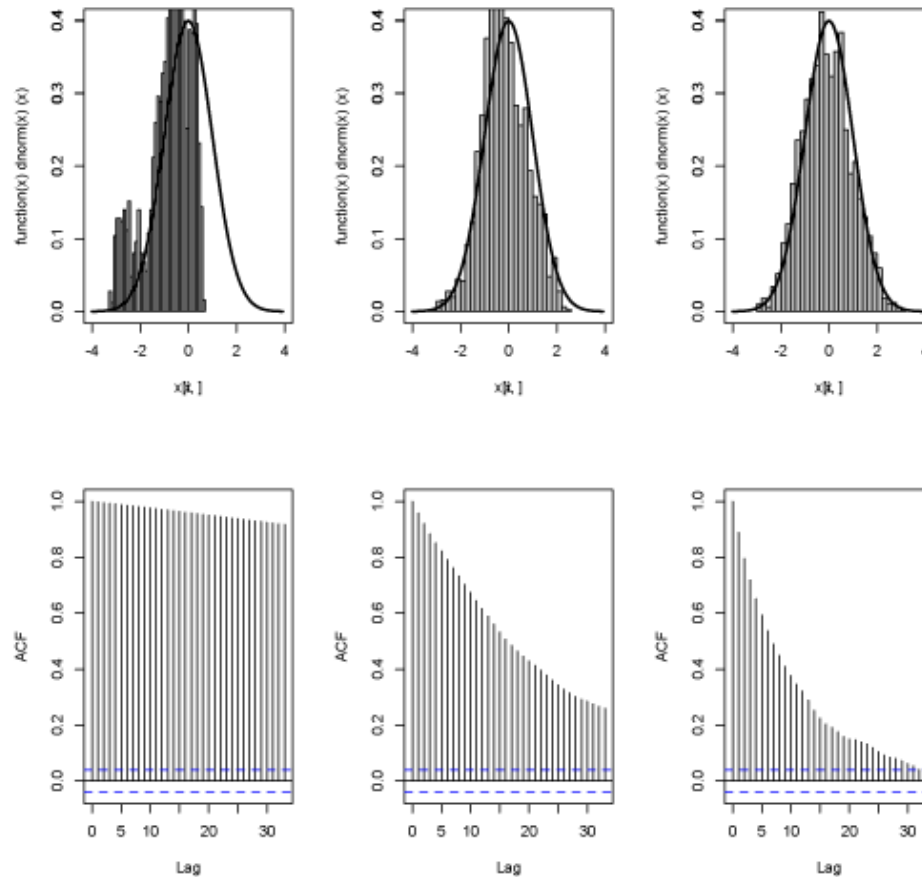
## Acceptance Rates

### Random Walk Metropolis–Hastings

- ▶ Independent Metropolis–Hastings algorithms
  - ▷ Can be optimized or compared through their acceptance rate
  - ▷ This reduces the number of replicas in the chain
  - ▷ And reduces the correlation level in the chain
- ▶ Not true for other types of Metropolis–Hastings algorithms
  - ▷ In a random walk, higher acceptance is not always better.
- ▶ The historical example of Hastings generates a  $\mathcal{N}(0, 1)$  from
  - ▷  $Y_t = X_{t-1} + \varepsilon_t$
  - ▷  $\rho(x^{(t)}, y_t) = \min\{\exp\{(x^{(t)2} - y_t^2)/2\}, 1\}, \quad \varepsilon_t \sim \mathcal{U}[-\delta, \delta]$
  - ▷  $\delta$  controls the acceptance rate

## Acceptance Rates

### Random Walk Metropolis–Hastings Example



►  $\delta = 0.1, 1, \text{ and } 10$

►  $\delta = 0.1$

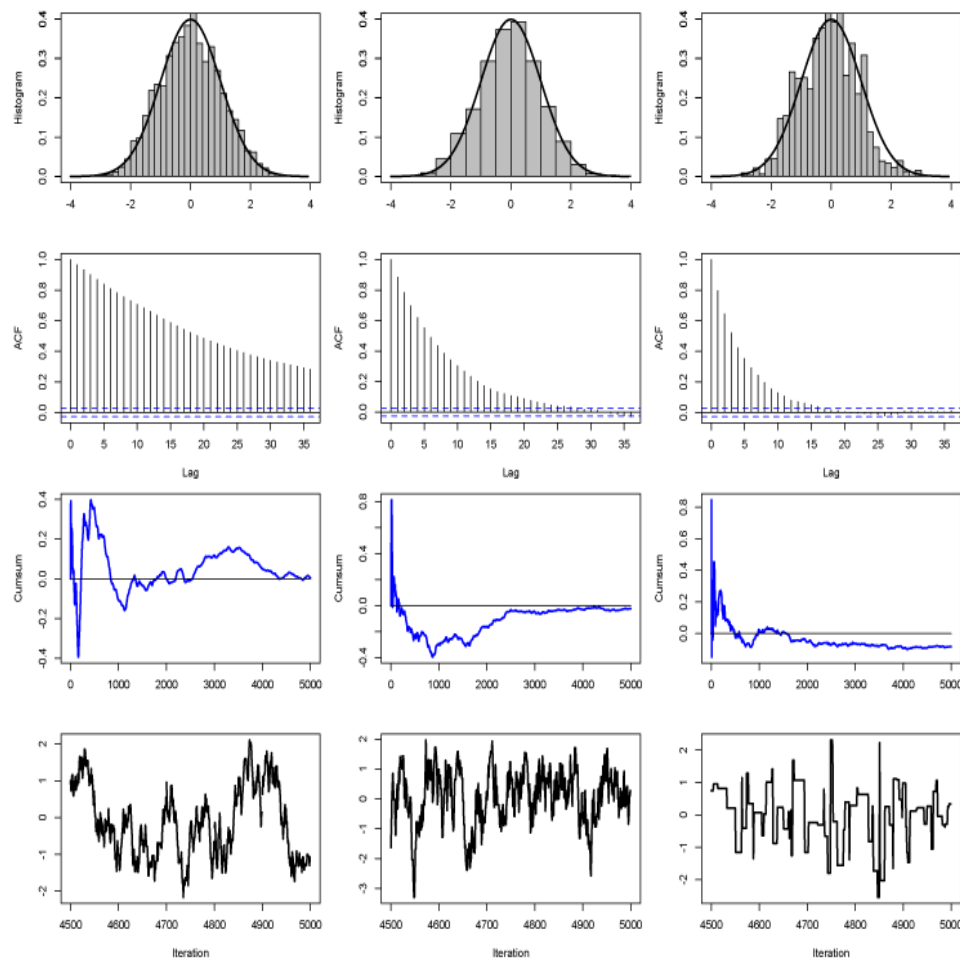
▷  $\uparrow$  autocovariance,  $\downarrow$  convergence

►  $\delta = 10$

▷  $\downarrow$  autocovariance, ? convergence

## Acceptance Rates

### Random Walk Metropolis–Hastings – All of the Details



### ► Acceptance Rates

▷  $\delta = 0.1 : 0.9832$

▷  $\delta = 1 : 0.7952$

▷  $\delta = 10 : 0.1512$

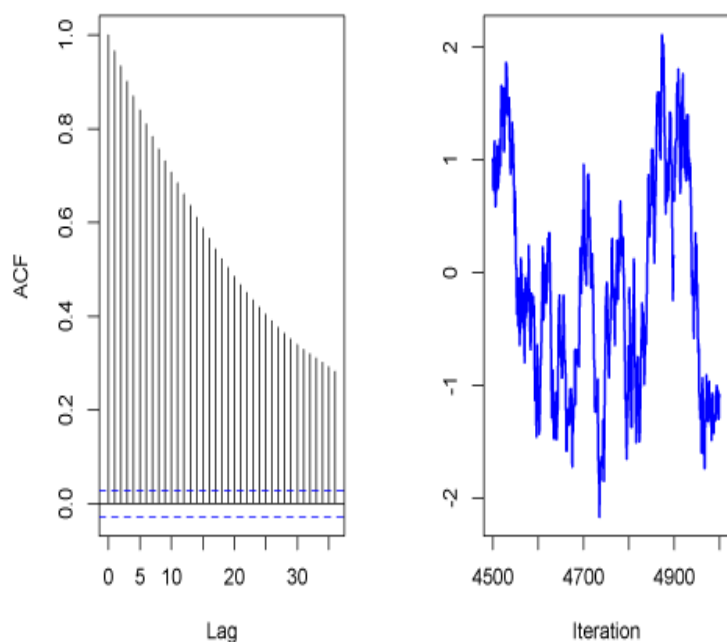
### ► Medium rate does better

▷ lowest better than the highest

## Random Walk Acceptance Rates

### Comments

- ▶ Random walk Metropolis–Hastings needs careful calibration of acceptance rates



- ▶ High acceptance rate
  - ▷ May not have satisfactory behavior
  - ▷ The chain may be moving too slowly on the surface of  $f$
- ▶ This is not always the case.
  - ▷  $f$  nearly flat  $\Rightarrow$  high acceptance OK

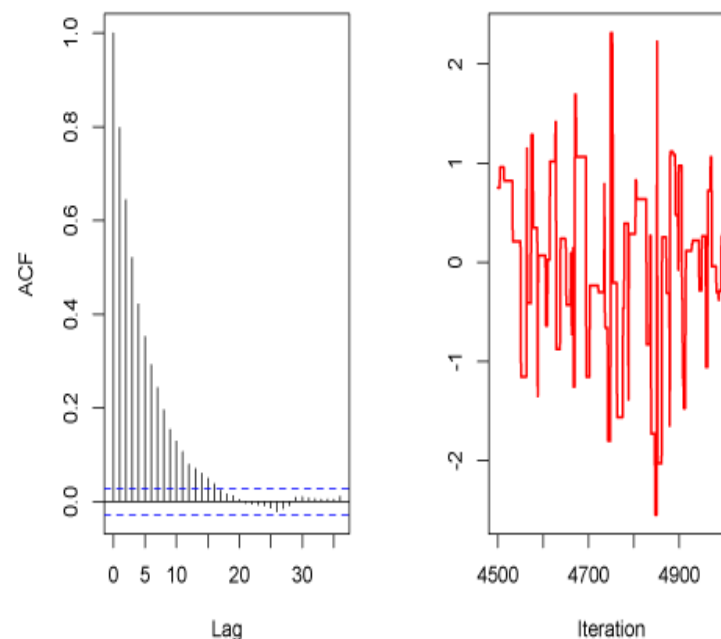
- ▶ But, unless  $f$  is completely flat, parts of the domain may be missed

## Random Walk Acceptance Rates More Comments

- ▶ In contrast, if the average acceptance rate is low
  - ▷ Successive values of  $f(y_t)$  are often are small compared to  $f(x^{(t)})$

- ▶ Low acceptance  $\Rightarrow$ 
  - ▷ The chain may not see all of  $f$
  - ▷ May miss an important but isolated mode of  $f$
- ▶ Nonetheless, low acceptance is less of an issue

- ▶ Golden acceptance rate:
  - ▷  $1/2$  for the models of dimension 1 or 2
  - ▷  $1/4$  in higher dimensions



## Chapter 7: Gibbs Samplers

*“Come, Watson , come!” he cried. “The game is afoot.”*

**Arthur Conan Doyle**

*The Adventure of the Abbey Grange*

### **This Chapter**

- ▶ We cover both the two-stage and the multistage Gibbs samplers
- ▶ The two-stage sampler has superior convergence properties
- ▶ The multistage Gibbs sampler is the workhorse of the MCMC world
- ▶ We deal with missing data and models with latent variables
- ▶ And, of course, hierarchical models



## Gibbs Samplers

### Introduction

- ▶ Gibbs samplers gather most of their calibration from the target density
  - ▶ They break complex problems (high dimensional) into a series of easier problems
    - ▷ May be impossible to build random walk Metropolis–Hastings algorithm
  - ▶ The sequence of simple problems may take a long time to converge
  - ▶ But Gibbs sampling is an interesting and useful algorithm.
- 
- ▶ *Gibbs sampling* is from the landmark paper by Geman and Geman (1984)
    - ▷ The Gibbs sampler is a special case of Metropolis–Hastings
  - ▶ Gelfand and Smith (1990) sparked new interest
    - ▷ In Bayesian methods and statistical computing
    - ▷ They solved problems that were previously unsolvable

## The Two-Stage Gibbs Sampler

### Introduction

- Creates a Markov chain from a joint distribution
- If two random variables  $X$  and  $Y$  have joint density  $f(x, y)$
- With corresponding conditional densities  $f_{Y|X}$  and  $f_{X|Y}$
- Generates a Markov chain  $(X_t, Y_t)$  according to the following steps

### Two-stage Gibbs sampler

Take  $X_0 = x_0$

For  $t = 1, 2, \dots$ , generate

1.  $Y_t \sim f_{Y|X}(\cdot | x_{t-1})$ ;

2.  $X_t \sim f_{X|Y}(\cdot | y_t)$  .

## The Two-Stage Gibbs Sampler Convergence

- ▶ The algorithm straightforward if simulating from both conditionals is feasible
- ▶ The stationary distribution is  $f(x, y)$
- ▶ Convergence of the Markov chain insured
  - ▷ Unless the supports of the conditionals are not connected

### Example: Normal bivariate Gibbs

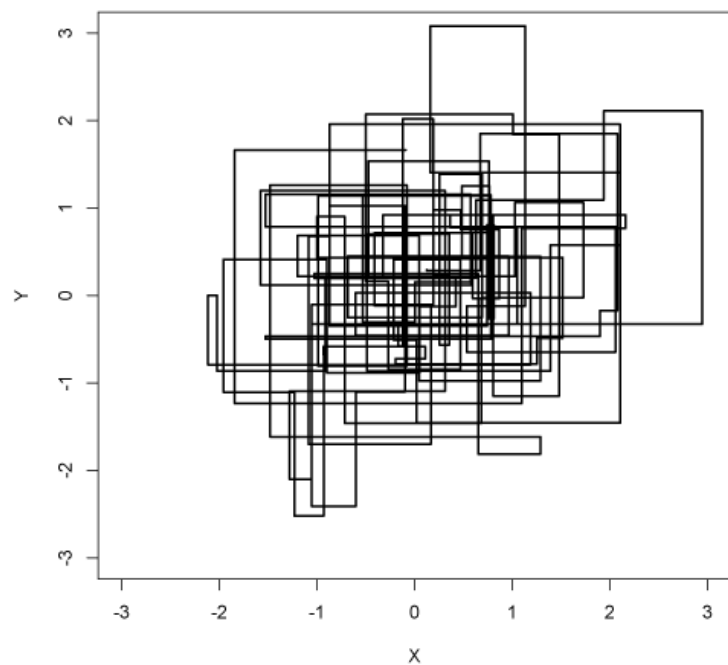
- ▶ Start with simple illustration, the bivariate normal model:

$$(X, Y) \sim \mathcal{N}_2 \left( 0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right),$$

- ▶ The the Gibbs sampler is Given  $x_t$ , generate

$$\begin{aligned} Y_{t+1} \mid x_t &\sim \mathcal{N}(\rho x_t, 1 - \rho^2), \\ X_{t+1} \mid y_{t+1} &\sim \mathcal{N}(\rho y_{t+1}, 1 - \rho^2). \end{aligned}$$

## The Two-Stage Gibbs Sampler Bivariate Normal Path



- Iterations  $(X_t, Y_t) \rightarrow (X_{t+1}, Y_{t+1})$
- Parallel to the axes
- Correlation affects mixing
- R `code`

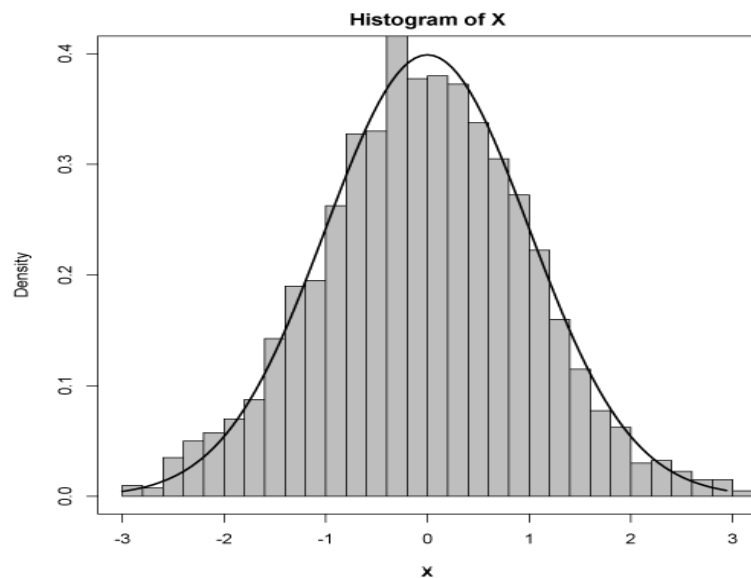
## The Two-Stage Gibbs Sampler Bivariate Normal Convergence

► The subchain  $(X_t)_t$  satisfies  $X_{t+1}|X_t = x_t \sim \mathcal{N}(\rho^2 x_t, 1 - \rho^4)$ ,

► A recursion shows that

$$X_t|X_0 = x_0 \sim \mathcal{N}(\rho^{2t} x_0, 1 - \rho^{4t}) \rightarrow \mathcal{N}(0, 1),$$

► We have converged to the *joint* distribution and both *marginal distributions*.



► Histogram of Marginal

► 2000 Iterations

## The Two-Stage Gibbs Sampler A First Hierarchical Model

- ▶ Gibbs sampling became popular
  - ▷ Since it was the perfect computational complement to hierarchical models
- ▶ A hierarchical model specifies a joint distribution
  - ▷ As successive layers of conditional distributions

Example: Generating beta-binomial random variables

- ▶ Consider the hierarchy

$$\begin{aligned} X|\theta &\sim \mathcal{B}\text{in}(n, \theta) \\ \theta &\sim \mathcal{B}e(a, b), \end{aligned}$$

- ▶ Which leads to the joint distribution

$$f(x, \theta) = \binom{n}{x} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{x+a-1} (1-\theta)^{n-x+b-1}.$$

## The Two-Stage Gibbs Sampler

### Beta-Binomial Conditionals

- The joint distribution

$$f(x, \theta) = \binom{n}{x} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{x+a-1} (1-\theta)^{n-x+b-1}.$$

- Has full conditionals

- ▷  $X|\theta \sim \mathcal{Bin}(n, \theta)$

- ▷  $\theta|X \sim \mathcal{Be}(X+a, n-X+b)$

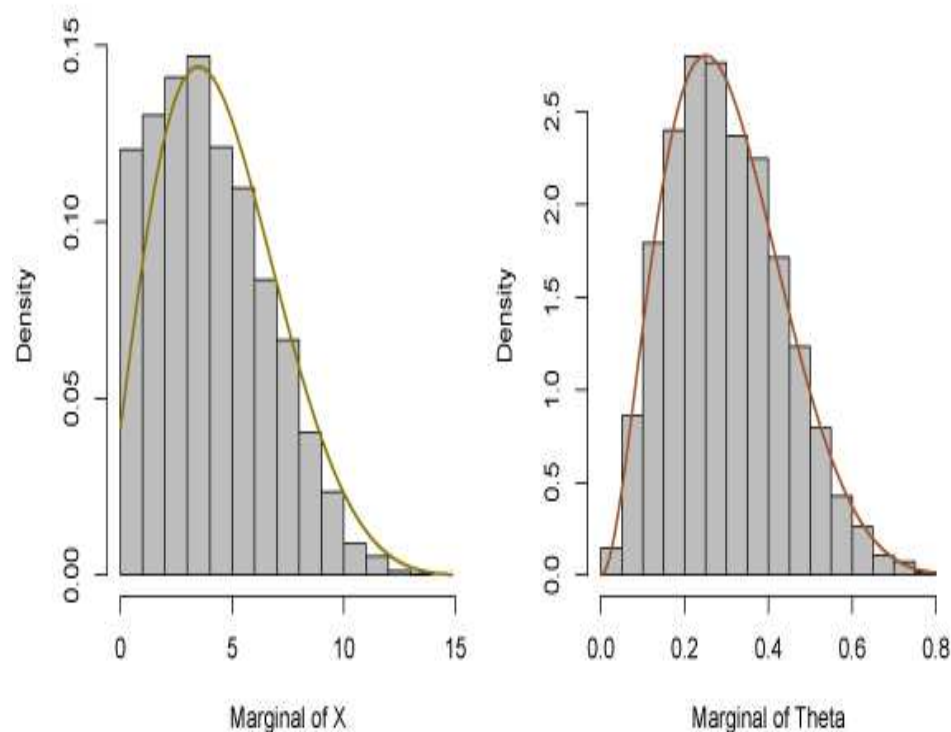
- This can be seen from

$$f(x, \theta) = \binom{n}{x} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{x+a-1} (1-\theta)^{n-x+b-1}.$$

## The Two-Stage Gibbs Sampler Beta-Binomial Marginals

- The marginal distribution of  $X$  is the [Beta-Binomial](#)

$$m(x) = \int_0^1 \binom{n}{x} \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{x+a-1} (1-\theta)^{n-x+b-1} d\theta$$



- Output from the Gibbs sampler
- $X$  and  $\theta$  marginals



## The Two-Stage Gibbs Sampler A First Normal Hierarchy

► A study on metabolism in 15-year-old females yielded the following data

```
> x=c(91,504,557,609,693,727,764,803,857,929,970,1043,
+      1089,1195,1384,1713)
```

▷ Their energy intake, measured in megajoules, over a 24 hour period.

► We model

$$\log(X) \sim \mathcal{N}(\theta, \sigma^2), \quad i = 1, \dots, n$$

▷ And complete the hierarchy with

$$\begin{aligned} \theta &\sim \mathcal{N}(\theta_0, \tau^2), \\ \sigma^2 &\sim \mathcal{IG}(a, b), \end{aligned}$$

where  $\mathcal{IG}(a, b)$  is the [inverted gamma](#) distribution.

## The Two-Stage Gibbs Sampler $\theta$ Conditional

- The posterior distribution  $\propto$  joint distribution is

$$f(\theta, \sigma^2 | \mathbf{x}) \propto \left[ \frac{1}{(\sigma^2)^{n/2}} e^{-\sum_i (x_i - \theta)^2 / (2\sigma^2)} \right] \times \left[ \frac{1}{\tau} e^{-(\theta - \theta_0)^2 / (2\tau^2)} \right] \times \left[ \frac{1}{(\sigma^2)^{a+1}} e^{1/b\sigma^2} \right]$$

(Here  $x = \log(x)$ )

- ▷ And now we can get the full conditionals

- $\theta$  conditional

$$f(\theta, \sigma^2 | \mathbf{x}) \propto \left[ \frac{1}{(\sigma^2)^{n/2}} e^{-\sum_i (x_i - \theta)^2 / (2\sigma^2)} \right] \times \left[ \frac{1}{\tau} e^{-(\theta - \theta_0)^2 / (2\tau^2)} \right] \times \left[ \frac{1}{(\sigma^2)^{a+1}} e^{1/b\sigma^2} \right]$$

$\Rightarrow$

$$\theta | \mathbf{x}, \sigma^2 \sim \mathcal{N} \left( \frac{\sigma^2}{\sigma^2 + n\tau^2} \theta_0 + \frac{n\tau^2}{\sigma^2 + n\tau^2} \bar{x}, \frac{\sigma^2 \tau^2}{\sigma^2 + n\tau^2} \right)$$

## The Two-Stage Gibbs Sampler $\sigma^2$ Conditional

- Again from the joint distribution

$$\Rightarrow f(\theta, \sigma^2 | \mathbf{x}) \propto \left[ \frac{1}{(\sigma^2)^{n/2}} e^{-\sum_i (x_i - \theta)^2 / (2\sigma^2)} \right] \times \left[ \frac{1}{\tau} e^{-(\theta - \theta_0)^2 / (2\tau^2)} \right] \times \left[ \frac{1}{(\sigma^2)^{a+1}} e^{1/b\sigma^2} \right]$$

$$\sigma^2 | \mathbf{x}, \theta \sim \mathcal{IG} \left( \frac{n}{2} + a, \frac{1}{2} \sum_i (x_i - \theta)^2 + b \right),$$

- We now have a Gibbs sampler using

$$\theta | \sigma^2 \sim \text{rnorm} \text{ and } (1/\sigma^2) | \theta \sim \text{rgamma}$$

- R code

## The Multistage Gibbs Sampler

### Introduction

- There is a natural extension from the two-stage to the multistage Gibbs sampler
- For  $p > 1$ , write  $\mathcal{X} = \mathbf{X} = (X_1, \dots, X_p)$ 
  - ▷ suppose that we can simulate from the **full conditional densities**

$$X_i | x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_p \sim f_i(x_i | x_1, x_2, \dots, x_{i-1}, x_{i+1}, \dots, x_p)$$

- The **multistage Gibbs sampler** has the following transition from  $X^{(t)}$  to  $X^{(t+1)}$ :

#### **The Multi-stage Gibbs Sampler**

At iteration  $t = 1, 2, \dots$ , given  $\mathbf{x}^{(t)} = (x_1^{(t)}, \dots, x_p^{(t)})$ , generate

1.  $X_1^{(t+1)} \sim f_1(x_1 | x_2^{(t)}, \dots, x_p^{(t)})$ ;
2.  $X_2^{(t+1)} \sim f_2(x_2 | x_1^{(t+1)}, x_3^{(t)}, \dots, x_p^{(t)})$ ;
- $\vdots$
- p.  $X_p^{(t+1)} \sim f_p(x_p | x_1^{(t+1)}, \dots, x_{p-1}^{(t+1)})$ .

## The Multistage Gibbs Sampler A Multivariate Normal Example

### Example: Normal multivariate Gibbs

- We previously saw a simple bivariate normal example
- Consider the multivariate normal density

$$(X_1, X_2, \dots, X_p) \sim \mathcal{N}_p(0, (1 - \rho)I + \rho J),$$

- ▷  $I$  is the  $p \times p$  identity matrix
- ▷  $J$  is a  $p \times p$  matrix of ones
- ▷  $\text{corr}(X_i, X_j) = \rho$  for every  $i$  and  $j$

- The full conditionals are

$$X_i | x_{(-i)} \sim \mathcal{N} \left( \frac{(p-1)\rho}{1 + (p-2)\rho} \bar{x}_{(-i)}, \frac{1 + (p-2)\rho - (p-1)\rho^2}{1 + (p-2)\rho} \right),$$

## The Multistage Gibbs Sampler

### Use of the Multivariate Normal Gibbs sampler

- ▶ The Gibbs sampler that generates from these univariate normals
  - ▷ Can then be easily derived
  - ▷ But it is not needed for this problem
- ▶ It is, however, a short step to consider
  - ▷ The setup where the components are restricted to a subset of  $\mathbb{R}^p$ .
  - ▷ If this subset is a hypercube,

$$\mathfrak{H} = \prod_{i=1} (a_i, b_i), \quad i = 1, \dots, p$$

the corresponding conditionals are the normals above restricted to  $(a_i, b_i)$

- ▶ These are easily simulated

## The Multistage Gibbs Sampler

### A Hierarchical Model for the Energy Data

- The oneway model can be a hierarchical model.
- Let  $X_{ij}$  be the energy intake,  $i = 1, 2$  (girl or boy),  $j = 1, n$ .

$$\log(X_{ij}) = \theta_i + \varepsilon_{ij}, \quad \varepsilon_{ij} \sim N(0, \sigma^2)$$

- We can complete this model with a hierarchical specification.
- There are different ways to parameterize this model. Here is one:

$$\log(X_{ij}) \sim \mathcal{N}(\theta_i, \sigma^2), \quad i = 1, \dots, k, \quad j = 1, \dots, n_i,$$

$$\theta_i \sim \mathcal{N}(\mu, \tau^2), \quad i = 1, \dots, k,$$

$$\mu \sim \mathcal{N}(\mu_0, \sigma_\mu^2),$$

$$\sigma^2 \sim \mathcal{IG}(a_1, b_1), \quad \tau^2 \sim \mathcal{IG}(a_2, b_2), \quad \sigma_\mu^2 \sim \mathcal{IG}(a_3, b_3).$$

## The Multistage Gibbs Sampler

### Full Conditionals for a Oneway Model

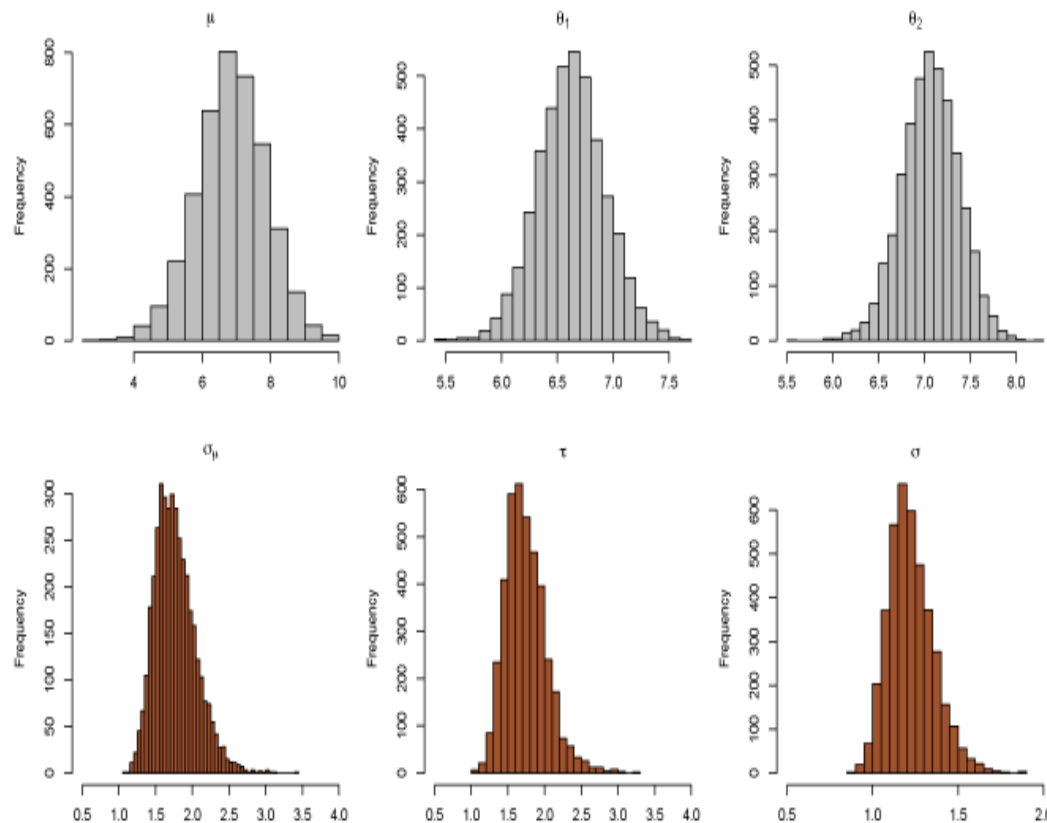
► Now, if we proceed as before we can derive the set of full conditionals

$$\begin{aligned}
 \theta_i &\sim \mathcal{N} \left( \frac{\sigma^2}{\sigma^2 + n_i \tau^2} \mu + \frac{n_i \tau^2}{\sigma^2 + n_i \tau^2} \bar{X}_i, \frac{\sigma^2 \tau^2}{\sigma^2 + n_i \tau^2} \right), \quad i = 1, \dots, k, \\
 \mu &\sim \mathcal{N} \left( \frac{\tau^2}{\tau^2 + k \sigma_\mu^2} \mu_0 + \frac{k \sigma_\mu^2}{\tau^2 + k \sigma_\mu^2} \bar{\theta}, \frac{\sigma_\mu^2 \tau^2}{\tau^2 + k \sigma_\mu^2} \right), \\
 \sigma^2 &\sim \mathcal{IG} \left( n/2 + a_1, (1/2) \sum_{ij} (X_{ij} - \theta_i)^2 + b_1 \right), \\
 \tau^2 &\sim \mathcal{IG} \left( k/2 + a_2, (1/2) \sum_i (\theta_i - \mu)^2 + b_2 \right), \\
 \sigma_\mu^2 &\sim \mathcal{IG} (1/2 + a_3, (1/2) (\mu - \mu_0)^2 + b_3),
 \end{aligned}$$

where  $n = \sum_i n_i$  and  $\bar{\theta} = \sum_i n_i \theta_i / n$ .



## The Multistage Gibbs Sampler Output From the Energy Data Analysis



► The top row:

- ▷ Mean  $\mu$  and  $\theta_1$  and  $\theta_2$ ,
- ▷ For the girl's and boy's energy

► Bottom row:

- ▷ Standard deviations.

► A variation is to give  $\mu$  a flat prior, which is equivalent to setting  $\sigma_\mu^2 = \infty$

► R code

## Missing Data and Latent Variables

### Introduction

- ▶ Missing Data Models start with the relation

$$g(x|\theta) = \int_{\mathbf{z}} f(x, z|\theta) \, dz$$

- ▷  $g(x|\theta)$  is typically the sample density or likelihood

- ▷  $f$  is arbitrary and can be chosen for convenience

- ▶ We implement a Gibbs sampler on  $f$

- ▶ Set  $y = (x, z) = (y_1, \dots, y_p)$  and run the Gibbs sampler

$$\begin{aligned} Y_1 | y_2, \dots, y_p &\sim f(y_1 | y_2, \dots, y_p), \\ Y_2 | y_1, y_3, \dots, y_p &\sim f(y_2 | y_1, y_3, \dots, y_p), \\ &\vdots \\ Y_p | y_1, \dots, y_{p-1} &\sim f(y_p | y_1, \dots, y_{p-1}). \end{aligned}$$

## Missing Data and Latent Variables Completion Gibbs Sampler

► For  $g(x|\theta) = \int_{\mathcal{Z}} f(x, z|\theta) \mathrm{d}z$

► And  $y = (x, z) = (y_1, \dots, y_p)$  with

$$Y_1|y_2, \dots, y_p \sim f(y_1|y_2, \dots, y_p),$$

$$Y_2|y_1, y_3, \dots, y_p \sim f(y_2|y_1, y_3, \dots, y_p),$$

$$\vdots$$

$$Y_p|y_1, \dots, y_{p-1} \sim f(y_p|y_1, \dots, y_{p-1}).$$

$$\triangleright Y^{(t)} = (X^{(t)}, Z^{(t)}) \rightarrow Y \sim f(x, z)$$

$$\triangleright X^{(t)} \rightarrow Y \sim f(x)$$

$$\triangleright Z^{(t)} \rightarrow Y \sim f(z)$$

►  $X^{(t)}$  and  $Z^{(t)}$  are not Markov chains

► But the subchains converge to the correct distributions

## Missing Data and Latent Variables Censored Data Models

### Example: Censored Data Gibbs

- Recall the censored data likelihood function

$$g(x|\theta) = L(\theta|x) \propto \prod_{i=1}^m e^{-(x_i-\theta)^2/2},$$

- And the complete-data likelihood

$$f(x, z|\theta) = L(\theta|x, z) \propto \prod_{i=1}^m e^{-(x_i-\theta)^2/2} \prod_{i=m+1}^n e^{-(z_i-\theta)^2/2}$$

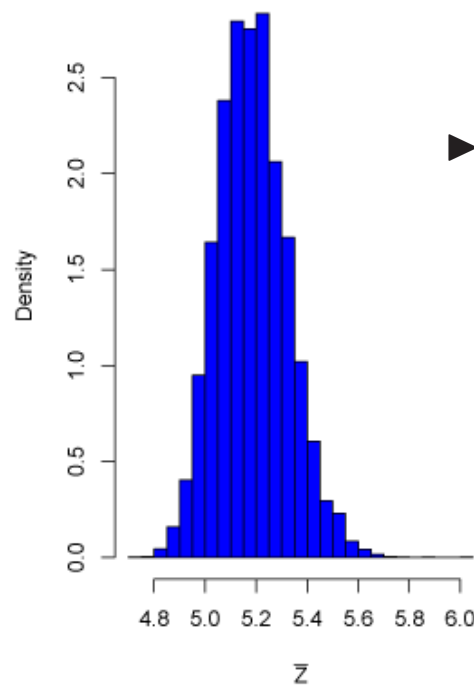
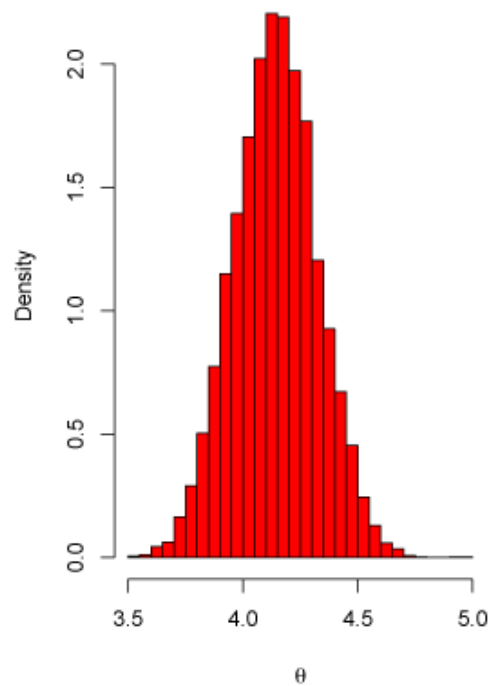
- ▷ With  $\theta \sim \pi(\theta)$  we have the Gibbs sampler

$$\pi(\theta|x, z) \quad \text{and} \quad f(z|x, \theta)$$

- ▷ With stationary distribution  $\pi(\theta, z|x)$ , the posterior of  $\theta$  and  $z$ .

## Missing Data and Latent Variables

### Censored Normal



► Flat prior  $\pi(\theta) = 1$ ,

$$\theta|x, z \sim \mathcal{N}\left(\frac{m\bar{x} + (n - m)\bar{z}}{n}, \frac{1}{n}\right),$$

$$Z_i|x, \theta \sim \frac{\varphi(z - \theta)}{\{1 - \Phi(a - \theta)\}}$$

- Each  $Z_i$  must be greater than the truncation point  $a$
- Many ways to generate  $Z$  (AR, `rtrun` from the package `bayesm`, PIT)
- R `code`

## Missing Data and Latent Variables

### Genetic Linkage

- ▶ We previously saw the classic genetic linkage data
- ▶ Such models abound
- ▶ Here is another, more complex, model

#### Observed genotype frequencies on blood type data

Genotype	Probability	Observed	Probability	Frequency
AA	$p_A^2$	A	$p_A^2 + 2p_Ap_O$	$n_A = 186$
AO	$2p_Ap_O$			
BB	$p_B^2$	B	$p_B^2 + 2p_Bp_O$	$n_B = 38$
BO	$2p_Bp_O$			
AB	$2p_Ap_B$	AB	$2p_Ap_B$	$n_{AB} = 13$
OO	$p_O^2$	O	$p_O^2$	$n_O = 284$

▶ Dominant allele  $\rightarrow$  missing data

▶ Cannot observe  $AO$  or  $BO$

▶ Observe  $X \sim \mathcal{M}_4(n; p_A^2 + 2p_Ap_O, p_B^2 + 2p_Bp_O, p_Ap_B, p_O^2)$

▷  $p_A + p_B + p_O = 1$

## Missing Data and Latent Variables

### Latent Variable Multinomial

- The observed data likelihood is

$$L(p_A, p_B, p_O | X) \propto (p_A^2 + 2p_A p_O)^{n_A} (p_B^2 + 2p_B p_O)^{n_B} (p_A p_B)^{n_{AB}} (p_O^2)^{n_O}$$

- With missing data (latent variables)  $Z_A$  and  $Z_B$ , the complete-data likelihood is

$$L(p_A, p_B, p_O | X, Z_A, Z_B) \propto (p_A^2)^{Z_A} (2p_A p_O)^{n_A - Z_A} (p_B^2)^{Z_B} (2p_B p_O)^{n_B - Z_B} (p_A p_B)^{n_{AB}} (p_O^2)^{n_O}.$$

- Giving the missing data density

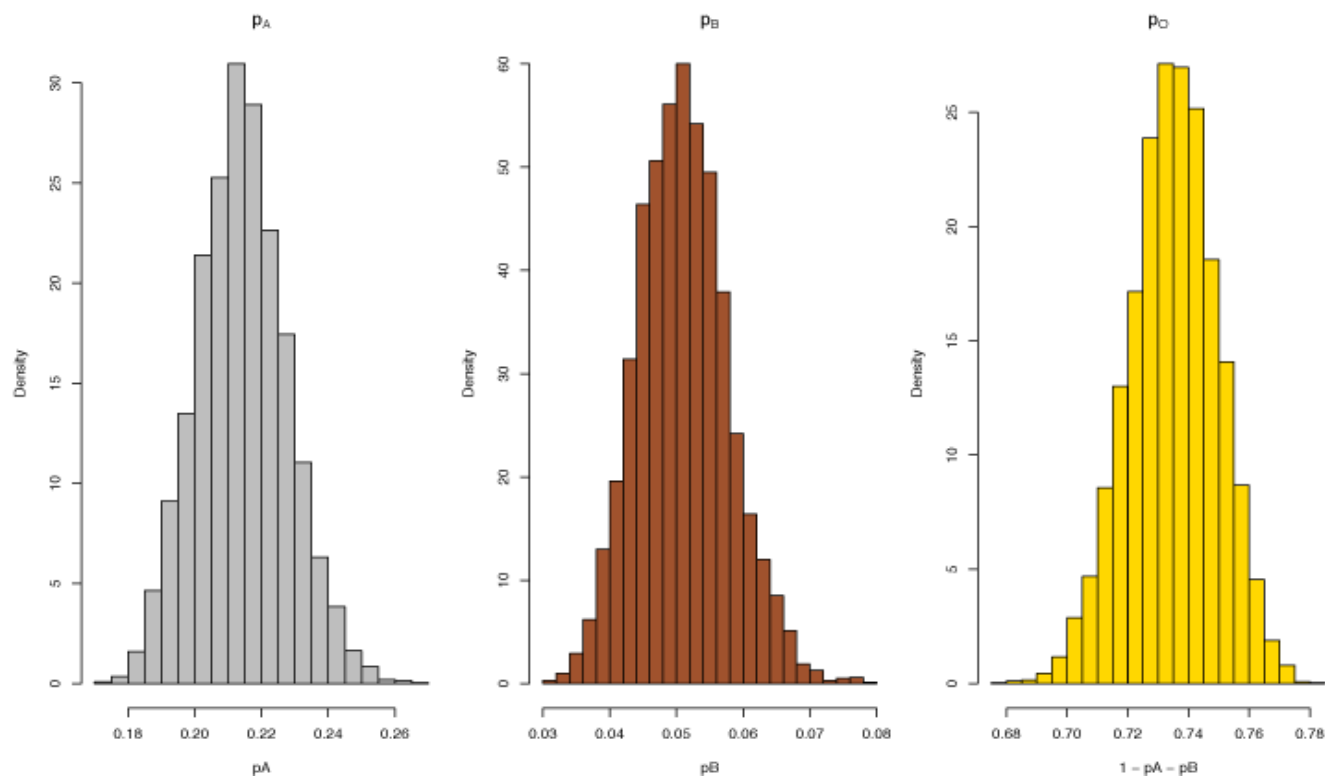
$$\left( \frac{p_A^2}{p_A^2 + 2p_A p_O} \right)^{Z_A} \left( \frac{2p_A p_O}{p_A^2 + 2p_A p_O} \right)^{n_A - Z_A} \left( \frac{p_B^2}{p_B^2 + 2p_B p_O} \right)^{Z_B} \left( \frac{2p_B p_O}{p_B^2 + 2p_B p_O} \right)^{n_B - Z_B}$$

- And the Gibbs sampler

$$p_A, p_B, p_O | X, Z_A, Z_B \sim \text{Dirichlet}, \quad Z_A, Z_B | p_A, p_B, p_O \sim \text{Independent Binomial}$$

## Missing Data and Latent Variables

### Analysis of Blood Types



- Estimated genotype frequencies
- Fisher had first developed these models
  - ▷ But he could not do the estimation: No EM, No Gibbs in 1930



## Multi-Stage Gibbs Samplers Hierarchical Structures

- ▶ We have seen the multistage Gibbs sampler applied to a number of examples
  - ▷ Many arising from missing-data structures.
- ▶ But the Gibbs sampler can sample from any hierarchical model
- ▶ A hierarchical model is defined by a sequence of conditional distributions
  - ▷ For instance, in the two-level generic hierarchy

$$\begin{aligned} X_i &\sim f_i(x|\theta), \quad i = 1, \dots, n, \quad \theta = (\theta_1, \dots, \theta_p), \\ \theta_j &\sim \pi_j(\theta|\gamma), \quad j = 1, \dots, p, \quad \gamma = (\gamma_1, \dots, \gamma_s), \\ \gamma_k &\sim g(\gamma), \quad k = 1, \dots, s. \end{aligned}$$

- ▶ The joint distribution from this hierarchy is

$$\prod_{i=1}^n f_i(x_i|\theta) \prod_{j=1}^p \pi_j(\theta_j|\gamma) \prod_{k=1}^s g(\gamma_k).$$

## Multi-Stage Gibbs Samplers Simulating from the Hierarchy

- With observations  $x_i$  the full posterior conditionals are

$$\theta_j \propto \pi_j(\theta_j|\gamma) \prod_{i=1}^n f_i(x_i|\theta), \quad j = 1, \dots, p,$$

$$\gamma_k \propto g(\gamma_k) \prod_{j=1}^p \pi_j(\theta_j|\gamma), \quad k = 1, \dots, s.$$

- ▷ In standard hierarchies, these densities are straightforward to simulate from
- ▷ In complex hierarchies, we might need to use a Metropolis–Hastings step
- ▷ Main message: full conditionals are easy to write down given the hierarchy

### ⚡ Note:

- When a full conditional in a Gibbs sampler cannot be simulated directly
  - ▷ One Metropolis–Hastings step is enough

## Multi-Stage Gibbs Samplers

### The Pump Failure Data

#### Example: Nuclear Pump Failures

- ▶ A benchmark hierarchical example in the Gibbs sampling literature
- ▶ Describes multiple failures of pumps in a nuclear plant
- ▶ Data:

Pump	1	2	3	4	5	6	7	8	9	10
Failures	5	1	5	14	3	19	1	1	4	22
Time	94.32	15.72	62.88	125.76	5.24	31.44	1.05	1.05	2.10	10.48

- ▶ Model: Failure of  $i^{\text{th}}$  pump follows a Poisson process
- ▶ For time  $t_i$ , the number of failures  $X_i \sim \mathcal{P}(\lambda_i t_i)$

## Multi-Stage Gibbs Samplers The Pump Failure Hierarchy

- The standard priors are gammas, leading to the hierarchical model

$$\begin{aligned} X_i &\sim \mathcal{P}(\lambda_i t_i), \quad i = 1, \dots, 10, \\ \lambda_i &\sim \mathcal{G}(\alpha, \beta), \quad i = 1, \dots, 10, \\ \beta &\sim \mathcal{G}(\gamma, \delta). \end{aligned}$$

- With joint distribution

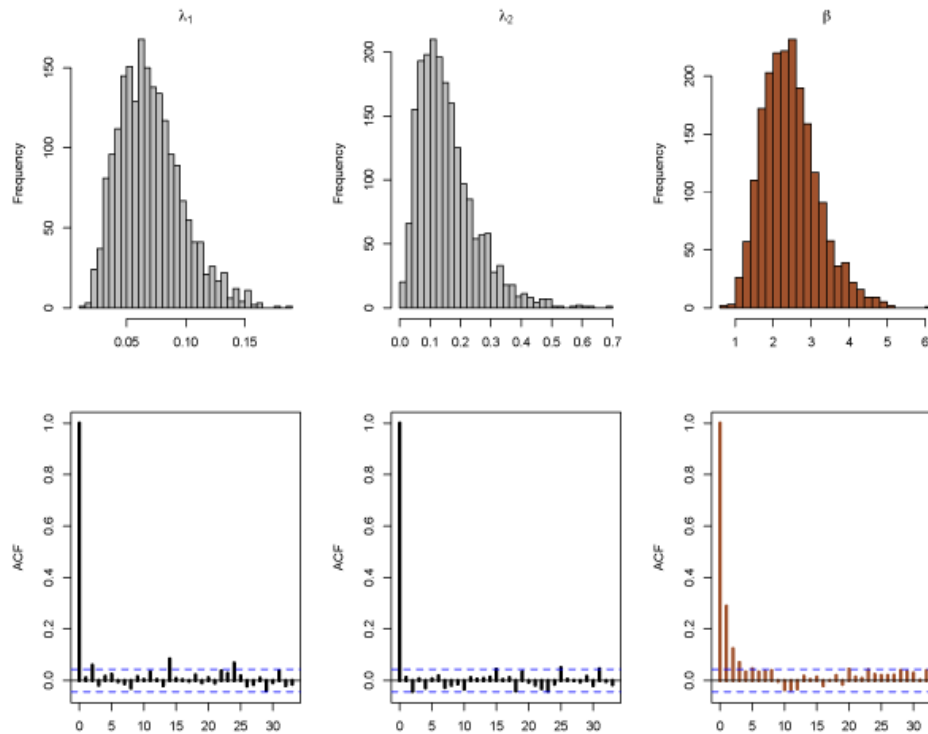
$$\prod_{i=1}^{10} \{ (\lambda_i t_i)^{x_i} e^{-\lambda_i t_i} \lambda_i^{\alpha-1} e^{-\beta \lambda_i} \} \beta^{10\alpha} \beta^{\gamma-1} e^{-\delta \beta}$$

- And full conditionals

$$\lambda_i | \beta, t_i, x_i \sim \mathcal{G}(x_i + \alpha, t_i + \beta), \quad i = 1, \dots, 10,$$

$$\beta | \lambda_1, \dots, \lambda_{10} \sim \mathcal{G} \left( \gamma + 10\alpha, \delta + \sum_{i=1}^{10} \lambda_i \right).$$

## Multi-Stage Gibbs Samplers The Pump Failure Gibbs Sampler



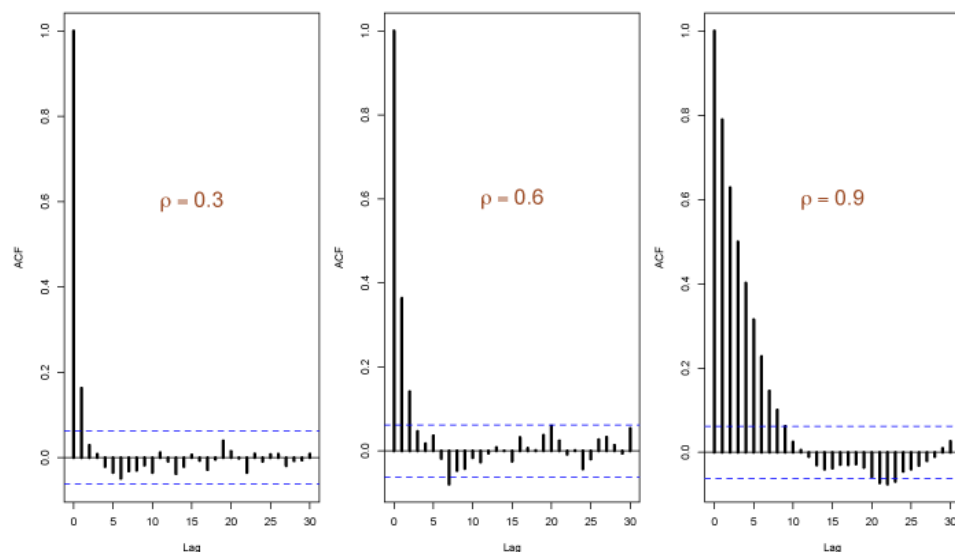
- The Gibbs sampler is easy
- Some selected output here
- Nice autocorrelations
- R [code](#)

- Goal of the pump failure data is to identify which pumps are more reliable.
  - ▷ Get 95% posterior credible intervals for each  $\lambda_i$  to assess this

## Other Considerations Reparameterization

- Many factors contribute to the convergence properties of a Gibbs sampler
- Convergence performance may be greatly affected by the parameterization
- High covariance may result in slow exploration.

### Simple Example



►  $(X, Y) \sim \mathcal{N}_2 \left( 0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right)$

►  $X + Y$  and  $X - Y$  are independent

- Autocorrelation for  $\rho = .3, .6, .9$

## Reparameterization Oneway Models

- ▶ Poor parameterization can affect both Gibbs sampling and Metropolis–Hastings
- ▶ No general consensus on a solution
  - ▷ Overall advice  $\Rightarrow$  make the components as independent as possible

▶ Example: Oneway model for the energy data

▶ Then

$$\begin{aligned}Y_{ij} &\sim \mathcal{N}(\theta_i, \sigma^2), \\ \theta_i &\sim \mathcal{N}(\mu, \tau^2), \\ \mu &\sim \mathcal{N}(\mu_0, \sigma_\mu^2),\end{aligned}$$

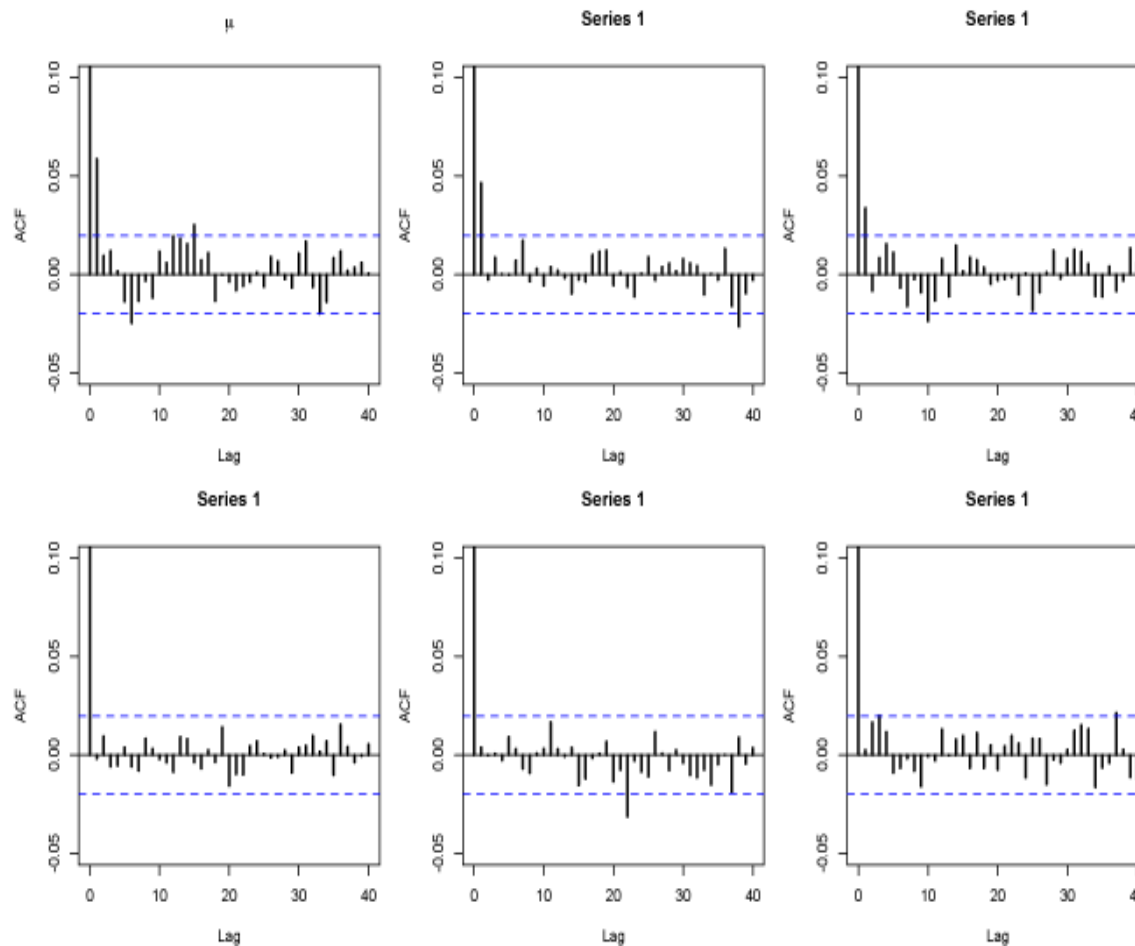
▶ Now

$$\begin{aligned}Y_{ij} &\sim \mathcal{N}(\mu + \theta_i, \sigma^2), \\ \theta_i &\sim \mathcal{N}(0, \tau^2), \\ \mu &\sim \mathcal{N}(\mu_0, \sigma_\mu^2).\end{aligned}$$

▶  $\mu$  at first level

▶  $\mu$  at second level

## Reparameterization Oneway Models for the Energy Data



- ▶ Top = Then
- ▶ Bottom = Now
- ▶ Very similar
- ▶ Then slightly better?



## Reparameterization Covariances of the Oneway Models

- But look at the covariance matrix of the subchain  $(\mu^{(t)}, \theta_1^{(t)}, \theta_2^{(t)})$

Then:  $Y_{ij} \sim \mathcal{N}(\theta_i, \sigma^2)$

Now:  $Y_{ij} \sim \mathcal{N}(\mu + \theta_i, \sigma^2)$

$$\begin{pmatrix} 1.056 & -0.175 & -0.166 \\ -0.175 & 1.029 & 0.018 \\ -0.166 & 0.018 & 1.026 \end{pmatrix}$$

$$\begin{pmatrix} 1.604 & 0.681 & 0.698 \\ 0.681 & 1.289 & 0.278 \\ 0.698 & 0.278 & 1.304 \end{pmatrix},$$

- So the new model is not as good as the old
- The covariances are all bigger
  - ▷ It will not mix as fast
- A pity: I like the new model better

## Rao–Blackwellization Introduction

- ▶ We have already seen Rao–Blackwellization in Chapter 4
  - ▷ Produced improved variance over standard empirical average
- ▶ For  $(X, Y) \sim f(x, y)$ , **parametric Rao–Blackwellization** is based on
  - ▷  $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}[\delta(Y)]$
  - ▷  $\text{var}[\delta(Y)] \leq \text{var}(X)$

### Example: Poisson Count Data

- ▶ For 360 consecutive time units
- ▶ Record the number of passages of individuals per unit time past some sensor.

Number of passages	0	1	2	3	4 or more
Number of observations	139	128	55	25	13

## Rao–Blackwellization Poisson Count Data

- ▶ The data involves a grouping of the observations with four passages or more.
- ▶ This can be addressed as a missing-data model
  - ▷ Assume that the ungrouped observations are  $X_i \sim \mathcal{P}(\lambda)$
  - ▷ The likelihood of the model is

$$\ell(\lambda|x_1, \dots, x_5) \propto e^{-347\lambda} \lambda^{128+55 \times 2+25 \times 3} \left( 1 - e^{-\lambda} \sum_{i=0}^3 \lambda^i / i! \right)^{13}$$

for  $x_1 = 139, \dots, x_5 = 13$ .

- ▶ For  $\pi(\lambda) = 1/\lambda$  and missing data  $\mathbf{z} = (z_1, \dots, z_{13})$ 
  - ▷ We have a completion Gibbs sampler from the full conditionals

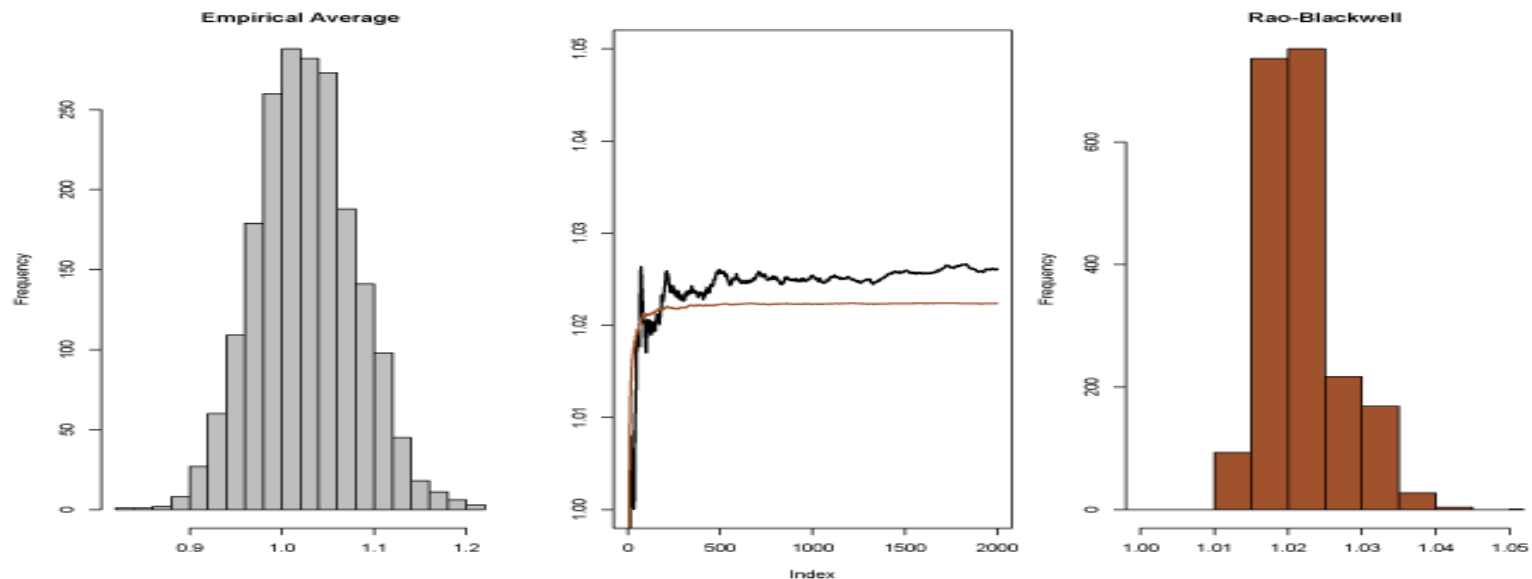
$$\begin{aligned} Z_i^{(t)} &\sim \mathcal{P}(\lambda^{(t-1)}) \mathbb{I}_{y \geq 4}, \quad i = 1, \dots, 13, \\ \lambda^{(t)} &\sim \mathcal{G} \left( 313 + \sum_{i=1}^{13} Z_i^{(t)}, 360 \right). \end{aligned}$$

## Rao–Blackwellization Comparing Estimators

- The empirical average is  $\frac{1}{T} \sum_{t=1}^T \lambda^{(t)}$
- The Rao–Blackwellized estimate of  $\lambda$  is then given by

$$\mathbb{E} \left[ \frac{1}{T} \sum_{t=1}^T \lambda^{(t)} \mid z_1^{(t)}, \dots, z_{13}^{(t)} \right] = \frac{1}{360T} \sum_{t=1}^T \left( 313 + \sum_{i=1}^{13} z_i^{(t)} \right),$$

- Note the massive variance reduction.



## Generating Truncated Poisson Variables Using While

- The truncated Poisson variable can be generated using the **while** statement

```
> for (i in 1:13){while(y[i]<4) y[i]=rpois(1,lam[j-1])}
```

or directly with

```
> prob=dpois(c(4:top),lam[j-1])
```

```
> for (i in 1:13) z[i]=4+sum(prob<runif(1)*sum(prob))
```

- Lets look at a comparison

- R **code**

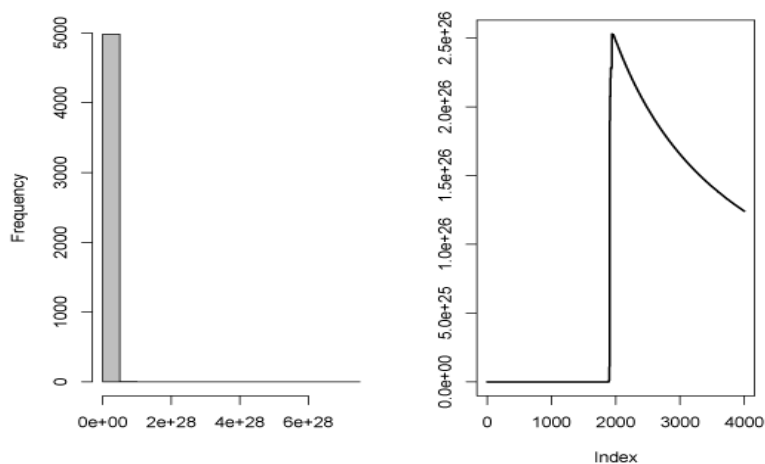
## Gibbs Sampling with Improper Priors

### Introduction

- ▶ There is a particular danger resulting from careless use of the Gibbs sampler.
  - ▶ The Gibbs sampler is based on conditional distributions
  - ▶ It is particularly insidious is that
    - (1) These conditional distributions may be well-defined
    - (2) They may be simulated from
    - (3) But may not correspond to any joint distribution!
- ▶ This problem is not a defect of the Gibbs sampler
  - ▶ It reflects use of the Gibbs sampler when assumptions are violated.
  - ▶ Corresponds to using Bayesian models with improper priors

## Gibbs Sampling with Improper Priors A Very Simple Example

- ▶ The Gibbs sampler can be constructed directly from conditional distributions
  - ▷ Leads to carelessness about checking the propriety of the posterior
- ▶ The pair of conditional densities  $X|y \sim \mathcal{Exp}(y)$ ,  $Y|x \sim \mathcal{Exp}(x)$ ,
  - ▷ Well-defined conditionals with no joint probability distribution.



- ▶ The pictures are absolute rubbish!
- ▶ Not a recurrent Markov chain
- ▶ Stationary measure =  $\exp(-xy)$
- ▶ No finite integral

- ▶ Histogram and cumulative average

## Gibbs Sampling with Improper Priors A Very Scary Example

- ▶ Oneway model  $Y_{ij} = \mu + \alpha_i + \varepsilon_{ij}$ ,
  - ▷  $\alpha_i \sim \mathcal{N}(0, \sigma^2)$  and  $\varepsilon_{ij} \sim \mathcal{N}(0, \tau^2)$
  - ▷ The Jeffreys (improper) prior for  $\mu$ ,  $\sigma$ , and  $\tau$  is  $\pi(\beta, \sigma^2, \tau^2) = \frac{1}{\sigma^2 \tau^2}$ .

### ▶ Conditional distributions

$$\alpha_i | y, \mu, \sigma^2, \tau^2 \sim \mathcal{N} \left( \frac{J(\bar{y}_i - \mu)}{J + \tau^2 \sigma^{-2}}, (J\tau^{-2} + \sigma^{-2})^{-1} \right),$$

$$\mu | \alpha, y, \sigma^2, \tau^2 \sim \mathcal{N}(\bar{y} - \bar{\alpha}, \tau^2 / IJ),$$

$$\sigma^2 | \alpha, \mu, y, \tau^2 \sim \mathcal{IG}(I/2, (1/2) \sum_i \alpha_i^2),$$

$$\tau^2 | \alpha, \mu, y, \sigma^2 \sim \mathcal{IG}(IJ/2, (1/2) \sum_{i,j} (y_{ij} - \alpha_i - \mu)^2),$$

- ▶ Are well-defined
- ▶ Can run a Gibbs sampler

- ▶ But there is no proper joint distribution
- ▶ Often this is impossible to detect by monitoring the output



## Gibbs Sampling with Improper Priors A Final Warning



- ▶ Graphical monitoring cannot exhibit deviant behavior of the Gibbs sampler.
- ▶ There are many examples, some published, of null recurrent Gibbs samplers
  - ▷ Undetected by the user
- ▶ The Gibbs sampler is valid only if the joint distribution has a finite integral.

- ▶ With improper priors in a Gibbs sampler
  - ▷ The posterior must always be checked for propriety.
- ▶ Improper priors on variances cause more trouble than those on means

## Chapter 8: Monitoring Convergence of MCMC Algorithms

*“Why does he insist that we must have a diagnosis? Some things are not meant to be known by man.”*

**Susanna Gregory**  
*An Unholy Alliance*

### This Chapter

- ▶ We look at different [diagnostics](#) to check the convergence of an MCMC algorithm
- ▶ To answer to question: “When do we stop our MCMC algorithm?”
- ▶ **We distinguish between two separate notions of convergence:**
  - ▷ Convergence to stationarity
  - ▷ Convergence of ergodic averages
- ▶ We also discuss some convergence diagnostics contained in the coda package

## Monitoring Convergence

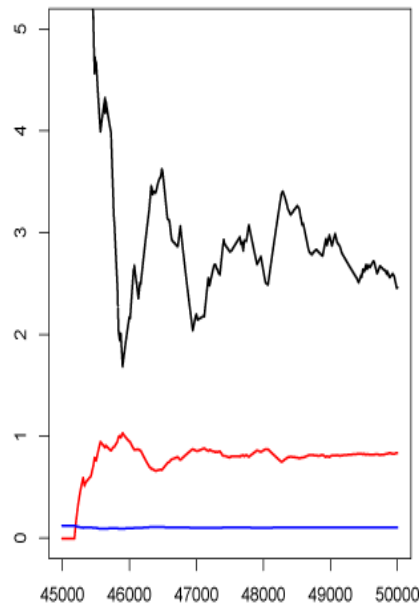
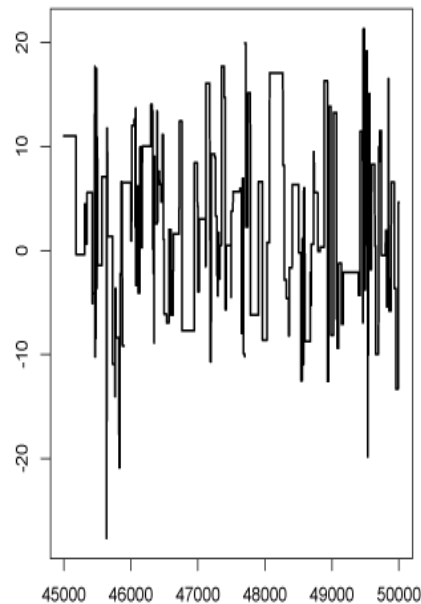
### Introduction

- ▶ The MCMC algorithms that we have seen
  - ▷ Are convergent because the chains they produce are ergodic.
- ▶ Although this is a necessary theoretical validation of the MCMC algorithms
  - ▷ It is insufficient from the implementation viewpoint
- ▶ Theoretical guarantees do not tell us
  - ▷ When to stop these algorithms and produce our estimates with confidence.
- ▶ In practice, this is nearly impossible
- ▶ Several runs of your program are usually required until
  - ▷ You are satisfied with the outcome
  - ▷ You run out of time and/or patience

## Monitoring Convergence

### Monitoring What and Why

- There are three types of convergence for which assessment may be necessary.



- Convergence to the stationary distribution
- Convergence of Averages
- Approximating iid Sampling

## Monitoring Convergence

### Convergence to the Stationary Distribution

- ▶ First requirement for convergence of an MCMC algorithm
    - ▷  $(x^{(t)}) \sim f$ , the stationary distribution
    - ▷ This sounds like a minimal requirement
  - ▶ Assessing that  $x^{(t)} \sim f$  is difficult with only a *single realization*
  - ▶ A slightly less ambitious goal: Assess the independence from the starting point  $x^{(0)}$  based on several realizations of the chain using the same transition kernel.
- ▶ When running an MCMC algorithm, the important issues are
    - ▷ The speed of exploration of the support of  $f$
    - ▷ The degree of correlation between the  $x^{(t)}$ 's

## Monitoring Convergence

### Tools for Assessing Convergence to the Stationary Distribution

- ▶ A major tool for assessing convergence: Compare performance of several chains
  - ▶ This means that the slower chain in the group governs the convergence diagnostic
  - ▶ Multiprocessor machines is an incentive for running replicate parallel chains
    - ▷ Can check for the convergence by using several chains at once
    - ▷ May not be much more costly than using a single chain
- ▶ Looking at a single path of the Markov chain produced by an MCMC algorithm makes it difficult to assess convergence
  - ▶ MCMC algorithms suffer from the major defect that
    - ▷ “you’ve only seen where you’ve been”
  - ▶ The support of  $f$  that has not yet been visited is almost impossible to detect.

## Monitoring Convergence Convergence of Averages

- ▶ A more important convergence issue is convergence of the empirical average

$$\frac{1}{T} \sum_{t=1}^T h(x^{(t)}) \rightarrow BE_f[h(X)]$$

- ▶ Two features that distinguish stationary MCMC outcomes from iid ones
  - ▷ The probabilistic dependence in the sample
  - ▷ The mixing behavior of the transition,
  - ▷ That is, how fast the chain explores the support of  $f$
- ▶ “Stuck in a mode” might appear to be stationarity
  - ▷ The missing mass problem again
- ▶ Also: The CLT might not be available

## Monitoring Convergence

### Approximating iid sampling

- ▶ Ideally, the approximation to  $f$  provided by MCMC algorithms should
  - ▷ Extend to the (approximate) production of iid samples from  $f$ .
- ▶ A practical solution to this issue is to use *subsampling* (or *batch sampling*)
  - ▷ Reduces correlation between the successive points of the Markov chain.
- ▶ Subsampling illustrates this general feature but it loses in efficiency
- ▶ Compare two estimators
  - ▷  $\delta_1$ : Uses all of the Markov chain
  - ▷  $\delta_2$ : Uses subsamples
- ▶ It can be shown that

$$\text{var}(\delta_1) \leq \text{var}(\delta_2)$$



## Monitoring Convergence

### The `coda` package

- ▶ Plummer *et al.* have written an **R** package called `coda`
- ▶ Contains many of the tools we will be discussing in this chapter
- ▶ Download and install with `library(coda)`
- ▶ Transform an MCMC output made of a vector or a matrix into an MCMC object that can be processed by coda, as in

```
> summary(mcmc(X))
```

or

```
> plot(mcmc(X))
```

## Monitoring Convergence to Stationarity Graphical Diagnoses

- ▶ A first approach to convergence control
  - ▷ Draw pictures of the output of simulated chains
- ▶ Componentwise as well as jointly
  - ▷ In order to detect deviant or nonstationary behaviors
- ▶ **coda** provides this crude analysis via the **plot** command
- ▶ When applied to an **mcmc** object
  - ▷ Produces a trace of the chain across iterations
  - ▷ And a non-parametric estimate of its density, parameter by parameter

## Monitoring Convergence to Stationarity Graphical Diagnoses for a Logistic Random Effect Model

Example: Random effect logit model

► Observations  $y_{ij}$  are modeled conditionally on one covariate  $x_{ij}$  as

$$P(y_{ij} = 1 | x_{ij}, u_i, \beta) = \frac{\exp \{ \beta x_{ij} + u_i \}}{1 + \exp \{ \beta x_{ij} + u_i \}}, i = 1, \dots, n, j = 1, \dots, m$$

▷  $u_i \sim \mathcal{N}(0, \sigma^2)$  is an unobserved random effect

▷ This is missing data

► We fit this with a Random Walk Metropolis–Hastings algorithm.

Monitoring Convergence to Stationarity  
Fitting a Logistic Random Effect Model

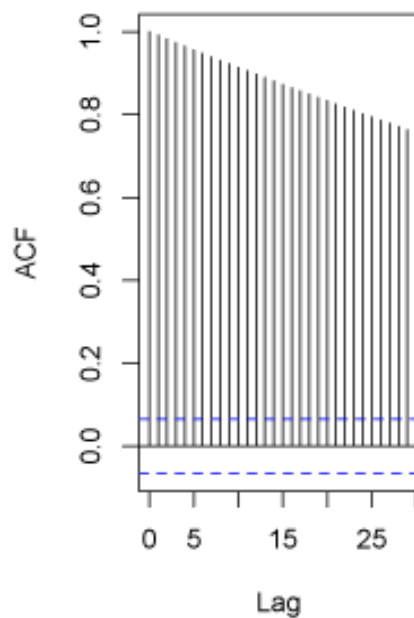
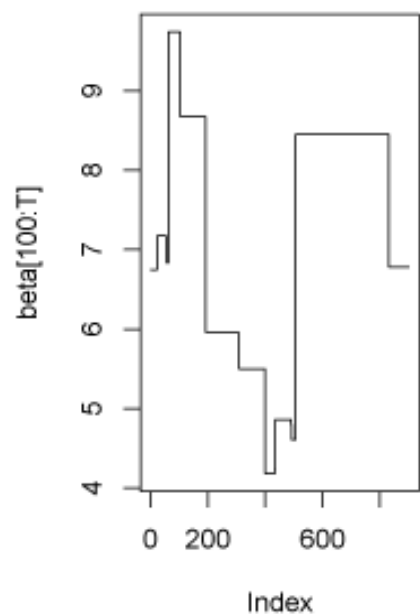
- The complete data likelihood is

$$\prod_{ij} \left( \frac{\exp \{ \beta x_{ij} + u_i \}}{1 + \exp \{ \beta x_{ij} + u_i \}} \right)^{y_{ij}} \left( \frac{1}{1 + \exp \{ \beta x_{ij} + u_i \}} \right)^{1-y_{ij}}$$

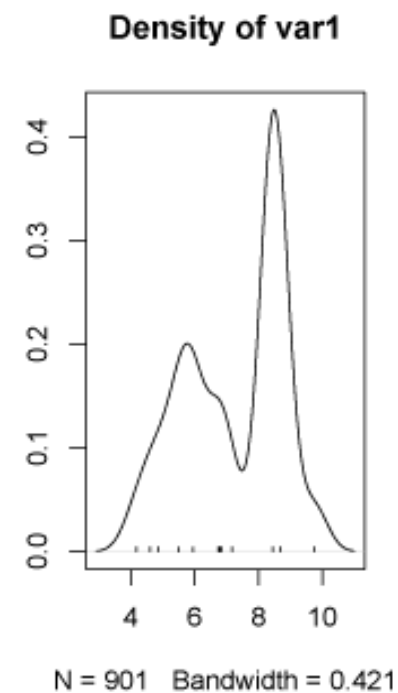
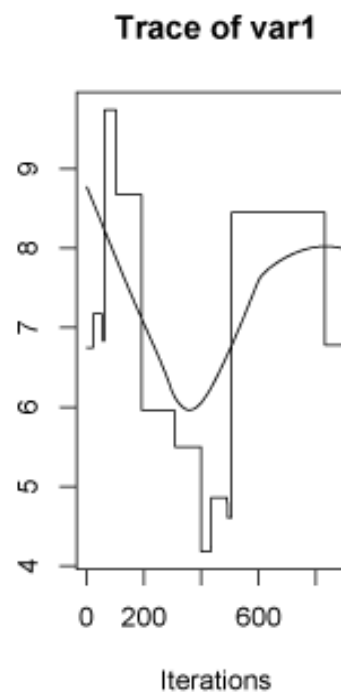
- This is the target in our Metropolis–Hastings algorithm
  - ▷ Simulate random effects  $u_i^{(t)} \sim N(u_i^{(t-1)}, \sigma^2)$
  - ▷ Simulate the logit coefficient  $\beta^{(t)} \sim N(\beta^{(t-1)}, \tau^2)$
  - ▷ Specify  $\sigma^2$  and  $\tau^2$
- $\sigma^2$  and  $\tau^2$  affect mixing

## Monitoring Convergence to Stationarity ACF and Coda

► Trace and acf:



► Coda



► R code

## Tests of Stationarity

### Nonparametric Tests: Kolmogorov-Smirnov

- ▶ Other than a graphical check, we can try to **test** for independence
- ▶ Standard non-parametric tests of fit, such as Kolmogorov–Smirnov
  - ▷ Apply to a single chain to compare the distributions of the two halves
  - ▷ Also can apply to parallel chains
- ▶ There needs to be a correction for the Markov correlation
  - ▷ The correction can be achieved by introducing a **batch** size
- ▶ We use

$$K = \frac{1}{M} \sup_{\eta} \left| \sum_{g=1}^M \mathbb{I}_{(0,\eta)}(x_1^{(gG)}) - \sum_{g=1}^M \mathbb{I}_{(0,\eta)}(x_2^{(gG)}) \right|$$

- ▷ With  $G =$  batch size,  $M =$  sample size

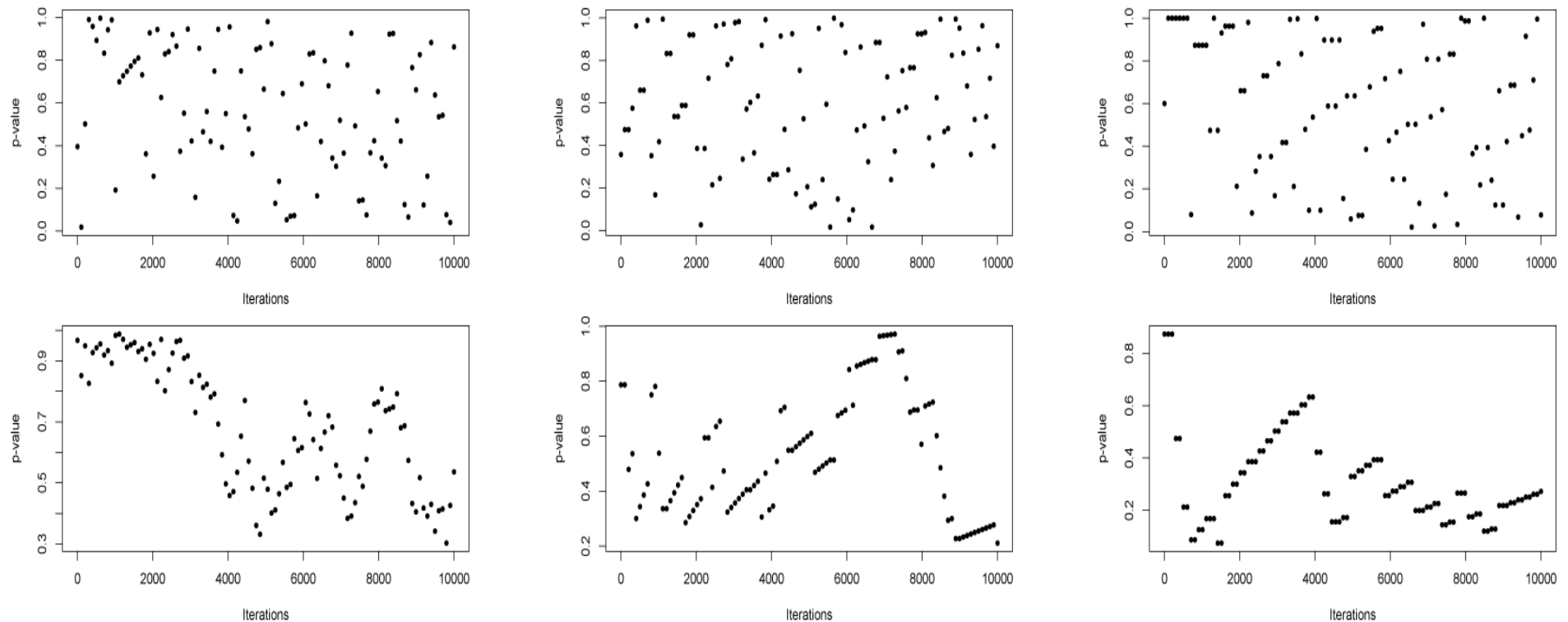
Tests of Stationarity  
Kolmogorov-Smirnov for the Pump Failure Data

Example: Poisson Hierarchical Model

- ▶ Consider again the nuclear pump failures
- ▶ We monitor the subchain  $(\beta^{(t)})$  produced by the algorithm
  - ▷ We monitor one chain split into two halves
  - ▷ We also monitor two parallel chains
- ▶ Use R command `ks.test`
- ▶ We will see (next slide) that the results are not clear

## Monitoring Convergence

### Kolmogorov-Smirnov $p$ -values for the Pump Failure Data



- ▶ Upper=split chain; Lower = Parallel chains; L → R: Batch size 10, 100, 200.
- ▶ Seems too variable to be of little use
- ▶ This is a good chain! (fast mixing, low autocorrelation)



## Monitoring Convergence Tests Based on Spectral Analysis

- ▶ There are convergence assessments spectral or Fourier analysis
- ▶ One is due to Geweke
  - ▷ Constructs the equivalent of a  $t$  test
  - ▷ Assess equality of means of the first and last parts of the Markov chain.

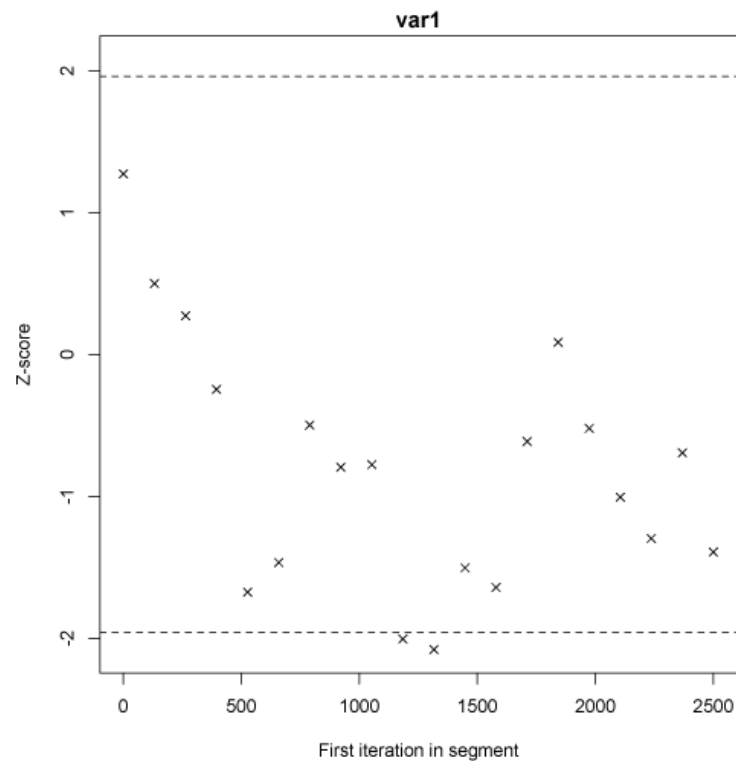
- ▶ The test statistic is

$$\sqrt{T}(\delta_A - \delta_B) / \sqrt{\frac{\sigma_A^2}{\tau_A} + \frac{\sigma_B^2}{\tau_B}},$$

- ▷  $\delta_A$  and  $\delta_B$  are the means from the first and last parts
  - ▷  $\sigma_A^2$  and  $\sigma_B^2$  are the spectral variance estimates
- ▶ Implement with `geweke.diag` and `geweke.plot`

## Monitoring Convergence

### Geweke Diagnostics for Pump Failure Data

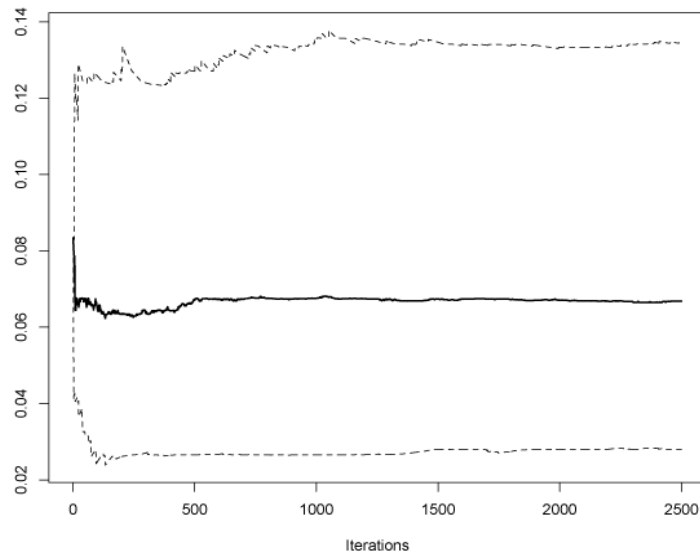


- For  $\lambda_1$ 
  - ▷  $t$ -statistic = 1.273
  - ▷ Plot discards successive beginning segments
  - ▷ Last  $z$ -score only uses last half of chain

- Heidelberger and Welch have a similar test: `heidel.diag`

## Monitoring Convergence of Averages Plotting the Estimator

- ▶ The initial and most natural diagnostic is to plot the evolution of the estimator
- ▶ If the curve of the cumulated averages has not stabilized after  $T$  iterations
  - ▷ The length of the Markov chain must be increased.
- ▶ The principle can be applied to multiple chains as well.
  - ▷ Can use `cumsum`, `plot(mcmc(coda))`, and `cumuplot(coda)`

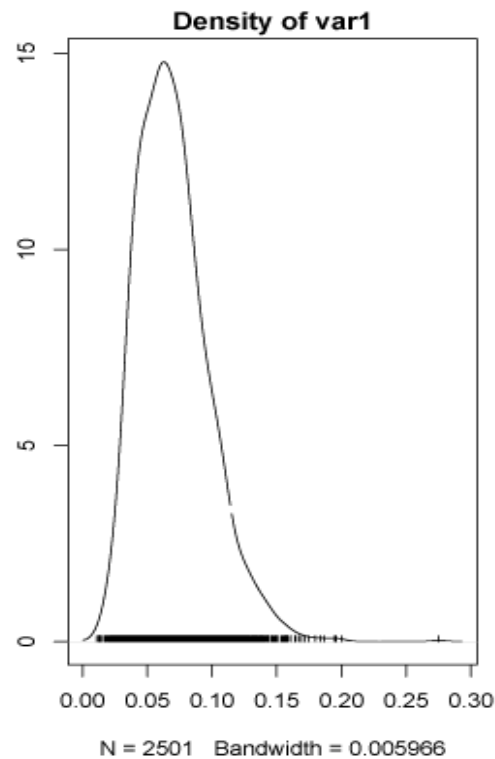
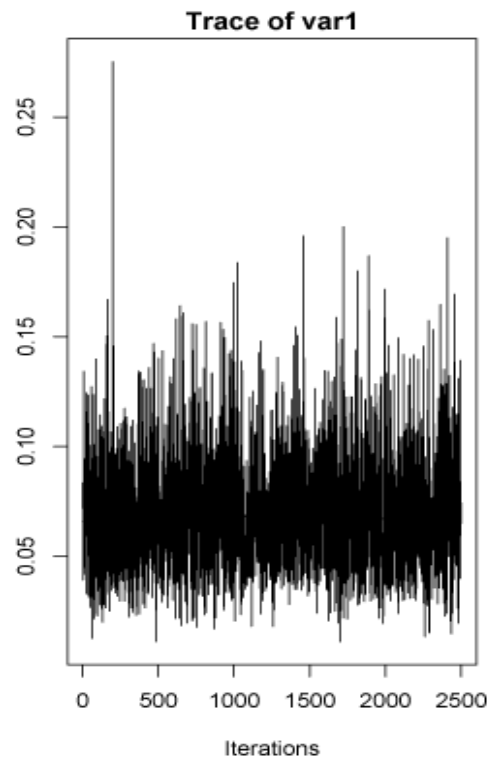


- ▶ For  $\lambda_1$  from Pump failures
- ▶ `cumuplot` of second half

## Monitoring Convergence of Averages

### Trace Plots and Density Estimates

► `plot(mcmc(lambda))` produces two graphs



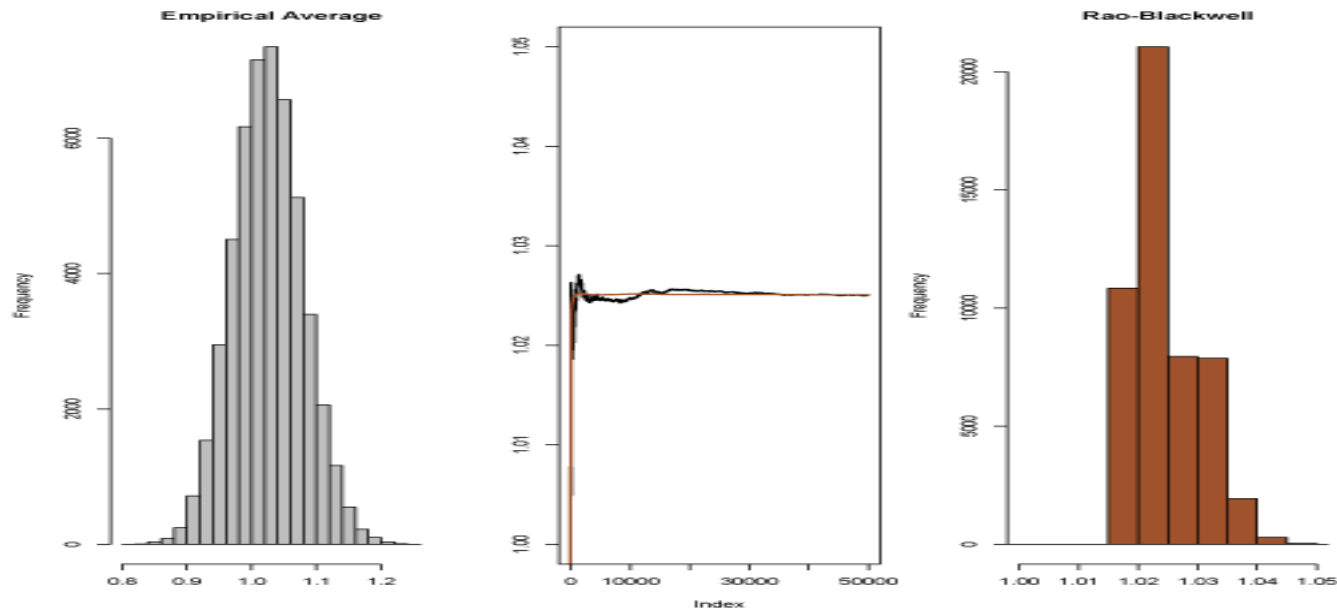
► Trace Plot

► Density Estimate

► Note: To get second half of chain `temp=lambda[2500:5000]`, `plot(mcmc(temp))`

## Monitoring Convergence of Averages Multiple Estimates

- ▶ Can use several convergent estimators of  $\mathbb{E}_f[h(\theta)]$  based on the same chain
  - ▷ Monitor until all estimators coincide
- ▶ Recall Poisson Count Data
  - ▷ Two Estimators of Lambda: Empirical Average and RB
  - ▷ Convergence Diagnostic → Both estimators converge - 50,000 Iterations



## Monitoring Convergence of Averages Computing Multiple Estimates

- ▶ Start with a Gibbs sampler  $\theta|\eta$  and  $\eta|\theta$
- ▶ Typical estimates of  $h(\theta)$ 
  - ▷ The empirical average  $S_T = \frac{1}{T} \sum_{t=1}^T h(\theta^{(t)})$
  - ▷ The Rao–Blackwellized version  $S_T^C = \frac{1}{T} \sum_{t=1}^T \mathbb{E}[h(\theta)|\eta^{(t)}]$ ,
  - ▷ Importance sampling:  $S_T^P = \sum_{t=1}^T w_t h(\theta^{(t)})$ ,
    - ▷  $w_t \propto f(\theta^{(t)})/g_t(\theta^{(t)})$
    - ▷  $f$  = target,  $g$  = candidate

## Monitoring Convergence of Multiple Estimates

### Cauchy Posterior Simulation

- The hierarchical model

$$\begin{aligned} X_i &\sim \text{Cauchy}(\theta), \quad i = 1, \dots, 3 \\ \theta &\sim N(0, \sigma^2) \end{aligned}$$

- Has posterior distribution

$$\pi(\theta | x_1, x_2, x_3) \propto e^{-\theta^2/2\sigma^2} \prod_{i=1}^3 \frac{1}{(1 + (\theta - x_i)^2)}$$

- We can use a Completion Gibbs sampler

$$\eta_i | \theta, x_i \sim \text{Exp} \left( \frac{1 + (\theta - x_i)^2}{2} \right),$$

$$\theta | x_1, x_2, x_3, \eta_1, \eta_2, \eta_3 \sim \mathcal{N} \left( \frac{\eta_1 x_1 + \eta_2 x_2 + \eta_3 x_3}{\eta_1 + \eta_2 + \eta_3 + \sigma^{-2}}, \frac{1}{\eta_1 + \eta_2 + \eta_3 + \sigma^{-2}} \right),$$

## Monitoring Convergence of Multiple Estimates Completion Gibbs Sampler

- The Gibbs sampler is based on the latent variables  $\eta_i$ , where

$$\int e^{-\frac{1}{2}\eta_i(1+(x_i-\theta)^2)} d\eta_i = \frac{2}{1 + (x_i - \theta)^2}$$

- With

$$\eta_i \sim \text{Exponential} \left( \frac{1}{2}(1 + (x_i - \theta)^2) \right)$$

- Monitor with three estimates of  $\theta$ 
  - ▷ Empirical Average
  - ▷ Rao-Blackwellized
  - ▷ Importance sample



## Monitoring Convergence of Multiple Estimates

### Calculating the Estimates

#### ► Empirical Average

$$\frac{1}{M} \sum_{j=1}^M \hat{\theta}^{(j)}$$

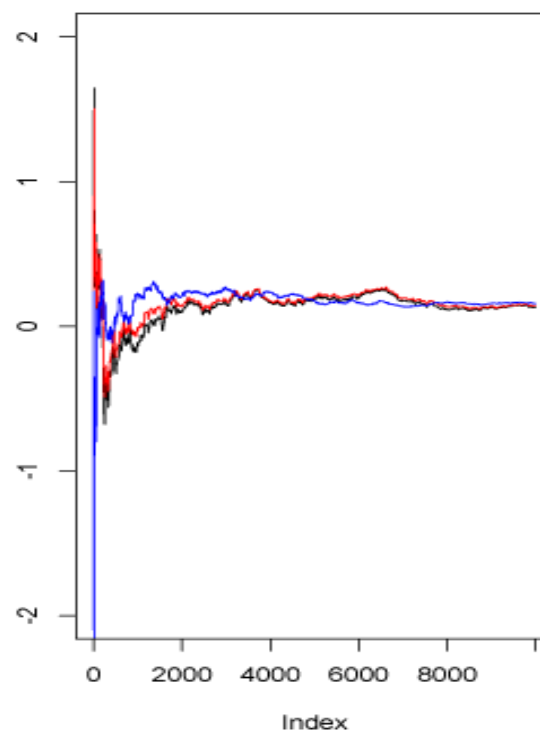
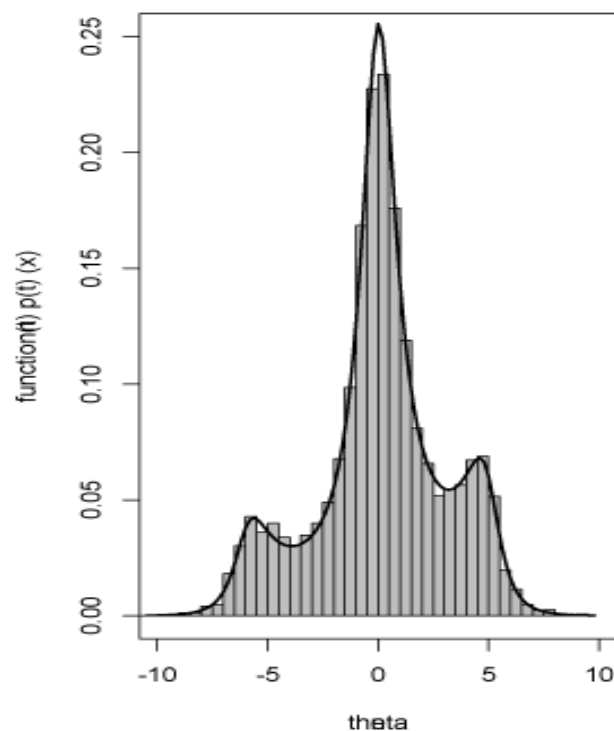
#### ► Rao-Blackwellized

$$\theta | \eta_1, \eta_2, \eta_3 \sim N \left( \frac{\sum_i \eta_i x_i}{\frac{1}{\sigma^2} + \sum_i \eta_i}, \left[ \frac{1}{\sigma^2} + \sum_i \eta_i \right]^{-1} \right)$$

#### ► Importance sampling with Cauchy candidate

## Monitoring Convergence of Multiple Estimates

### Monitoring the Estimates



► Emp. Avg

► RB

► IS

▷ Estimates converged

▷ IS seems most stable

- When applicable, superior diagnostic to single chain
- Intrinsically conservative
  - ▷ Speed of convergence determined by slowest estimate

## Monitoring Convergence of Averages Between and Within Variances

- ▶ The Gelman-Rubin diagnostic uses multiple chains
- ▶ Based on a **between-within** variance comparison (anova-like)
  - ▷ Implemented in `coda` as `gelman.diag(coda)` and `gelman.plot(coda)`.
- ▶ For  $m$  chains  $\{\theta_1^{(t)}\}, \dots, \{\theta_m^{(t)}\}$ 
  - ▷ The between-chain variance is  $B_T = \frac{1}{M-1} \sum_{m=1}^M (\bar{\theta}_m - \bar{\theta})^2$ ,
  - ▷ The within-chain variance is  $W_T = \frac{1}{M-1} \sum_{m=1}^M \frac{1}{T-1} \sum_{t=1}^T (\theta_m^{(t)} - \bar{\theta}_m)^2$
- ▶ If the chains have converged, these variances are the same (anova null hypothesis)

## Monitoring Convergence of Averages

### Gelman-Rubin Statistic

- ▶  $B_T$  and  $W_T$  are combined into an  $F$ -like statistic
- ▶ The **shrink factor**, defined by

$$R_T^2 = \frac{\hat{\sigma}_T^2 + \frac{B_T}{M}}{W_T} \frac{\nu_T + 1}{\nu_T + 3},$$

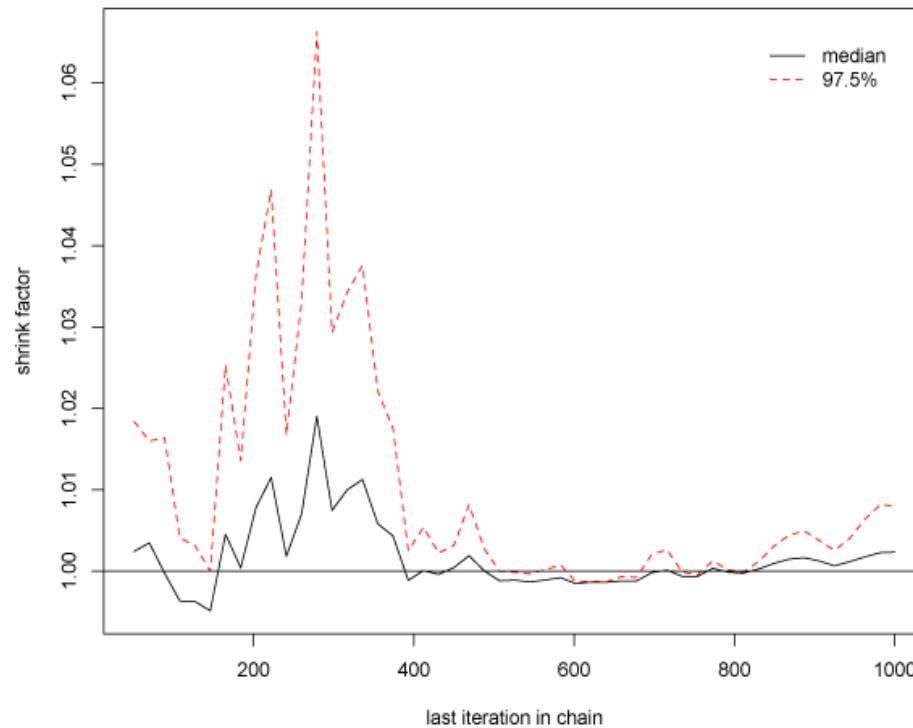
$$\triangleright \hat{\sigma}_T^2 = \frac{T-1}{T} W_T + B_T.$$

▷  $F$ -distribution approximation

- ▶ Enjoyed wide use because of simplicity and intuitive connections with anova
- ▶  $R_T$  does converge to 1 under stationarity,
- ▶ However, its distributional approximation relies on normality
- ▶ These approximations are at best difficult to satisfy and at worst not valid.

## Monitoring Convergence of Averages

### Gelman Plot for Pump Failures



- ▶ Three chains for  $\lambda_1$
- ▶ nsim=1000
- ▶ Suggests convergence
- ▶ `gelman.diag` gives

Point est.	97.5 % quantile
1.00	1.01

▶ R code

We Did All This!!!

1. Intro

2. Generating

3. MCI

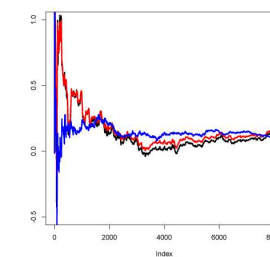
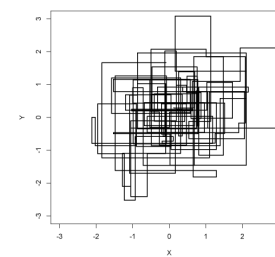
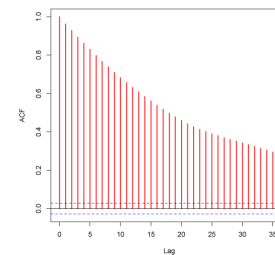
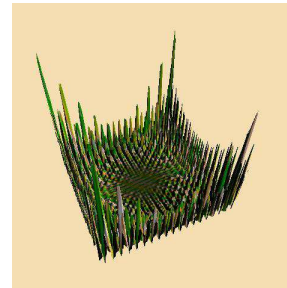
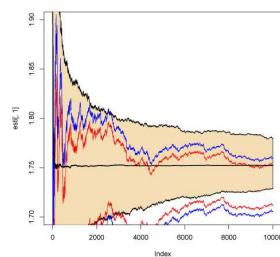
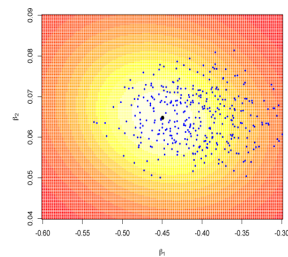
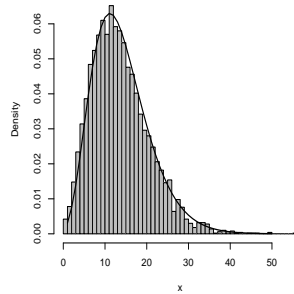
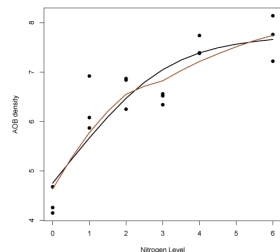
4. Acceleration

5. Optimization

6. Metropolis

7. Gibbs

8. Convergence



Thank You for Your Attention

George Casella

casella@ufl.edu

