#### Simulación Gibbs Sampler Software asociado a MCMC

Jorge de la Vega Góngora

Departamento de Estadística, Instituto Tecnológico Autónomo de México

Clase 13: Semana 2 al 4 de Mayo de 2018

**GS** Software

Gibbs Sampler

### Gibbs Sampler I

El Gibbs Sampler es un caso particular del algoritmo de Metropolis-Hastings que se aplica cuando la distribución objetivo a muestrear es una distribución multivariada, y parte de la siguiente idea:

- La distribución objetivo  $\pi(\theta)$  tiene un vector de variables  $\theta$  que se puede particionar en k subpartes:  $\theta = (\theta_1, \dots, \theta_k)$
- La cadena de Markov se genera muestreando de las distribuciones condicionales de la distribución objetivo:

$$\pi(\boldsymbol{\theta}_j|\boldsymbol{\theta}_{-j},y)$$
, donde  $\boldsymbol{\theta}_{-j}=\boldsymbol{\theta}\setminus\boldsymbol{\theta}_j,\quad j=1,\ldots,k$ 

iterando sobre los k subvectores de parámetros  $\theta_{-i}$ .

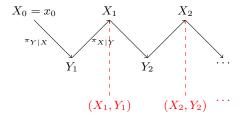
En el caso bidimensional,  $(X,Y) \sim \pi(x,y)$ . Se genera una cadena de Markov bidimensional  $(X_t, Y_t)$  del siguiente modo. Para  $n = 1, 2, \ldots$ 

- Se inicializa la cadena  $X_0 = x_0$ .
- Genera  $Y_t \sim \pi_{Y|X}(\cdot|X_{t-1})$ .
- Genera  $X_t \sim \pi_{X|Y}(\cdot|Y_t)$
- Incrementa t a t+1 hasta n

### Gibbs Sampler II

En el caso del Gibbs sampler, a diferencia del Metropolis-Hastings, cada candidato generado es aceptado.

Una representación gráfica del flujo del algoritmo se muestra a continuación:



### Ejemplo 1 GS Normal bivariada I

Consideremos una normal bivariada  $\mathcal{N}(\mu, \Sigma)$  con  $\mu = (\mu_1, \mu_2)$  y

 $\Sigma = \begin{pmatrix} \sigma_1^2 & \rho \\ \rho & \sigma_2^2 \end{pmatrix}$ . En este caso es fácil conocer las marginales:

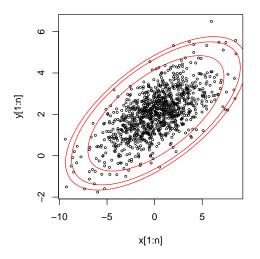
$$Y_{n+1}|X_n \sim \mathcal{N}\left(\mu_2 + \rho \frac{\sigma_2}{\sigma_1}(X_n - \mu_1), (1 - \rho^2)\sigma_2^2\right)$$
  
 $X_{n+1}|Y_{n+1} \sim \mathcal{N}\left(\mu_1 + \rho \frac{\sigma_1}{\sigma_2}(Y_{n+1} - \mu_2), (1 - \rho^2)\sigma_1^2\right)$ 

Entonces  $(X_n,Y_n) \to \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ . A continuación se muestra la simulación para el caso  $\rho=0.5,\ \mu_1=0,\ \mu_2=2,\ \sigma_1=1\ \text{y}\ \sigma_2=2$ 

### Ejemplo 1 GS Normal bivariada II

```
library (mixtools) #para la función ellipse.
mixtools package, version 1.1.0, Released 2017-03-10
This package is based upon work supported by the National Science Foundation under Grant No.
SES-0518772.
par(ptv="s") #plot cuadrado
n <- 1000; x0 <- 0 #valor incial
x <- NULL; y <- NULL
x < -append(x0,x)
rho <- 0.5; mul <- 0; mu2 <- 2; sigmal <- 1; sigma2 <- 2 #valores dados
for(i in 1:n) {
        v <- append(v, rnorm(1, mean=mu2 + rho*sigmal/sigma2*(x[i]-mul), sd=sgrt(1-rho^2))*sigmal)
        x <- append(x, rnorm(1, mean=mul + rho*sigma2/sigmal*(y[i]-mu2), sd=sqrt(1-rho^2))*sigma2)
plot (x[1:n],y[1:n],cex=0.5)
ellipse(mu = colMeans(cbind(x[1:n],v[1:n])), sigma = cov(cbind(x[1:n],v[1:n])),
ellipse(mu = colMeans(cbind(x[1:n],y[1:n])), sigma = cov(cbind(x[1:n],y[1:n])),
ellipse(mu = colMeans(cbind(x[1:n],y[1:n])), sigma = cov(cbind(x[1:n],y[1:n])),
```

### Ejemplo 1 GS Normal bivariada III



Si se particiona el vector  $\mathbf{X}$  en k componentes  $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k)$ , y podemos simular de  $f_i(\mathbf{x}_i|\mathbf{x}_{-i})$ , entonces podemos aplicar el siguiente algoritmo:

#### Algoritmo Gibbs sampler

- **1** Inicializar  $\mathbf{X}^{(0)} = (\mathbf{X}_1^{(0)}, \mathbf{X}_2^{(0)}, \dots, \mathbf{X}_L^{(0)})$  y j = 1
- ② Obtener  $\mathbf{X}^{(j)}$  a partir de  $\mathbf{X}^{(j-1)}$  generando

**3** Hacer j := j + 1 y repetir el paso 2 hasta que la sucesión  $\mathbf{X}^{(n)}$  converja a la distribución objetivo.

Supongan que X, P y N son tres variables aleatorias con densidad conjunta:

$$\pi(x, p, n) \propto \binom{n}{x} p^x (1-p)^{n-x} \frac{4^n}{n!}$$

para  $x \in \{0, 1, ..., n\}, p \in (0, 1), n \in \mathbb{N}$ . Con esta estructura, notemos que:

- $X|P=p, N=n \propto \mathbf{Bin}(n,p)$ .
- $P|X = x, N = n \propto \mathcal{B}e(x+1, n-x+1).$
- $N|X=x, P=p \propto \frac{(1-p)^{n-x}4^n}{(n-x)!}$  para  $n=x,x+1,\ldots$  Esta es una distribución Poisson desplazada con parámetro 4(1-p), es decir,  $N|P=p, X=x\sim Z+x \text{ donde } Z\sim \mathcal{P}\left(4(1-p)\right).$

De acuerdo al algoritmo dado, la implementación se realiza de la siguiente manera:

- Inicializar  $(x_0, p_0, n_0) \leftarrow (1, 0.5, 2)$
- Para m = 1, ..., M
  - $x_m \sim \text{Bin}(n_{m-1}, p_{m-1})$

### Ejemplo II

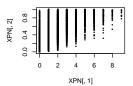
Ejemplo, Casella y George (1992)

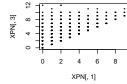
• 
$$p_m \sim \mathcal{B}e(x_m + 1, n_{m-1} - x_m + 1)$$
  
•  $z \sim \mathcal{P}(4(1 - p_m)) \text{ y } n_m = z + x_m$ 

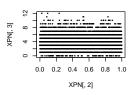
# El algoritmo anterior nos da la secuencia de valores $(X_0, P_0, N_0), (X_1, P_1, N_1), \dots (X_M, P_M, N_M)$

# Ejemplo III

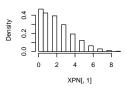
Ejemplo, Casella y George (1992)



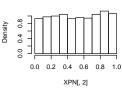




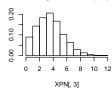
#### Histogram of XPN[, 1]



#### Histogram of XPN[, 2]



#### Histogram of XPN[, 3]



Density

- El Gibbs sampler funciona porque es un caso especial del algoritmo de Metropolis-Hastings.
- Para verlo, consideremos un sólo paso en la iteración para mostrar la lógica. Todos los pasos en cada variable siguen el mismo razonamiento.
- Sea  $i = (x_1, x_2, \dots, x_k) \in \mathcal{S}$  el estado actual y  $j = (x_1^*, x_2, \dots, x_k)$  el estado propuesto. Se toma a la distribución candidata  $q(\cdot|\cdot)$  como la distribución condicional de  $X_1$  dados  $X_2, \ldots, X_k$ .
- La probabilidad de aceptación del paso es  $\alpha(i,j) = \frac{\pi_j q(i|j)}{\pi \cdot \alpha(i|i)} = \frac{\pi_j q_{ji}}{\pi_j q_{ji}}$ . Entonces

$$\pi_{j}q_{ji} = \pi(x_{1}^{*}, x_{2}, \dots, x_{k}) f_{X_{1}|X_{2}, \dots, X_{k}}(x_{1}|x_{2}, \dots, x_{k}) 
= \pi(x_{1}^{*}, x_{2}, \dots, x_{k}) \left( \frac{\pi(x_{1}, x_{2}, \dots, x_{k})}{\int \pi(x, x_{2}, \dots, x_{k}) dx} \right) 
= \pi(x_{1}, x_{2}, \dots, x_{k}) \left( \frac{\pi(x_{1}^{*}, x_{2}, \dots, x_{k})}{\int \pi(x, x_{2}, \dots, x_{k}) dx} \right) 
= \pi(x_{1}, x_{2}, \dots, x_{k}) f_{X_{1}|X_{2}, \dots, X_{k}}(x_{1}^{*}|x_{2}, \dots, x_{k}) 
= \pi_{i}q_{ij}$$

Así que  $\alpha(i,j) = 1$ , lo que implica que con esta elección de función q el estado propuesto siempre se acepta.

#### Ejemplo 2: GS en modelos jerárquicos I Modelo jerárquico Binomial-beta

El Gibbs sampler se hizo muy popular en los 90's por sus aplicaciones en estadística Bayesiana, sobre todo en los problemas de estimación de modelos jerárquicos, ya que se cuenta con las marginales de manera explícita.

#### Ejemplo

Consideren el modelo jerárquico

$$x|\theta \sim \mathbf{Bin}(n,\theta)$$
  
 $\theta \sim \mathcal{B}e(a,b)$ 

Entonces se tiene que la densidad conjunta de X y  $\theta$  se puede escribir como

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

Entonces  $\theta | X \sim \mathcal{B}e(x+a, n-x+b)$  (beta y binomial son familias conjugada). Así que para generar de la conjunta  $(X, \theta)$  se puede generar de  $f(X|\theta)$  y de  $f(\theta|X)$ .

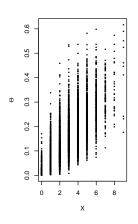
Queremos obtener muestras de la conjunta de los datos y el parámetro. Se considera una binomial con n = 15, con a = 1 y b = 8.

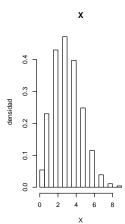
## Ejemplo 2: GS en modelos jerárquicos II

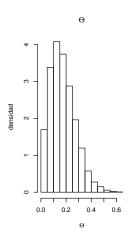
Modelo jerárquico Binomial-beta

```
nsim <- 5000
a <- 1; b <- 8 #Parámetros de Beta
X <- Theta <- array(0,dim=c(nsim,1))</pre>
Theta[1] <- rbeta(1,a,b)
X[1] <- rbinom(1,n,Theta[1])</pre>
for(i in 2:nsim) {
        X[i] <- rbinom(1,n,Theta[1])</pre>
        Theta[i] <- rbeta(1,a+X[i],n-X[i]+b)
par (mfrow=c(1,3))
plot (X, Theta, pch=16, cex=0.5, ylab=expression (Theta))
hist(X, probability = T, main = "X", ylab = "densidad")
hist (Theta, probability = T, main = expression (Theta),
     xlab = expression(Theta), ylab = "densidad")
```

Modelo jerárquico Binomial-beta







#### Aplicación: Análisis de punto de cambio I

• **Problema**: Considerar el siguiente proceso Poisson con punto de cambio:

$$X_{t} \sim \begin{cases} \mathcal{P}(\mu t) & 0 < t \leq k \\ \mathcal{P}(\lambda t) & t > k \end{cases}$$

Dada una muestra de n observaciones del proceso anterior, estimar  $\mu$ ,  $\lambda$  y k.

- Este es un proceso muy común y que ha sido ampliamente estudiado. Hay varias opciones de especificación de modelos Bayesianos para resolverlo.
- Una aplicación concreta del problema anterior se relaciona con los datos publicados en el artículo de R.G. Jarret: A note on the intervals between coal-mining disasters. Biometrika, 66:191-193, 1979. Los datos se encuentran en coal, en el paquete de R boot y corresponden a las fechas de 191 explosiones en minas de carbón que resultaron en más de 10 fatalidades desde marzo 15, 1851 hasta marzo 22, 1962.

#### Aplicación: Análisis de punto de cambio II

```
library(boot)
par(mar=c(2,2,3,1))
data(coal)
fead(coal) #fecha en numeros consecutivos desde 1851

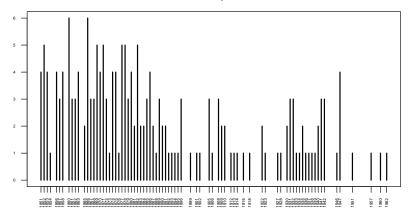
date

1851.203
2 1851.632
3 1851.969
4 1851.975
5 1852.314
6 1852.347

año <- floor(coal) #tomamos el año del evento
y <- table(año) #número de eventos por año
plot(y,las=2,main="Accidentes en Minas por año 1851-1962",cex.axis=0.4)
```

### Aplicación: Análisis de punto de cambio III

#### Accidentes en Minas por año 1851-1962



Los datos muestran un cambio en el número promedio de desastres por año alrededor de 1900. Los números anuales de accidentes se crean a continuación:

#### Aplicación: Análisis de punto de cambio IV

```
año <- floor(coal[[]]) #extrae las fechas como vector
y <- tabulate(año) #usa tabulate para obtener los años con frecuencia 0
y <- y[1851:length(y)]
y

[1] 4 5 4 1 0 4 3 4 0 6 3 3 4 0 2 6 3 3 5 4 5 3 1 4 4 1 5 5 3 4 2 5 2 2 3
[36] 4 2 1 3 2 2 1 1 1 1 3 0 0 1 0 1 1 1 0 0 3 1 0 3 2 2 0 1 1 1 0 1 0 1 0 0
[71] 0 2 1 0 0 0 1 1 1 0 2 3 3 1 1 2 1 1 1 1 2 3 3 0 0 0 1 4 0 0 0 1 0 0 0 0
[106] 0 1 0 0 1 0 1
```

#### Modelado:

• Sea  $Y_i$  = número de desastres en el año i (1851=1). Entonces, si k es el año punto de cambio,

$$Y_i \sim \mathcal{P}(\mu), \qquad i = 1, \dots k,$$
  
 $Y_i \sim \mathcal{P}(\lambda), \qquad i = k + 1, \dots, n$ 

Hay n=112 observaciones terminando en el año 1962. El parámetro a estimar es  $\theta=(k,\mu,\lambda)$ .

• Se requiere estimar como distribución objetivo la posterior  $\pi(\theta|y) = \pi(k,\mu,\lambda|y)$ , y en particular, la distribución posterior de k,  $\pi(k|y,\mu,\lambda)$ .

### Aplicación: Análisis de punto de cambio V

 Se puede construir un modelo Bayesiano con las siguientes distribuciones iniciales independientes:

$$k \sim U\{1, 2, \dots, n\},$$
  
 $\mu \sim \mathcal{G}(a_1, b_1),$   
 $\lambda \sim \mathcal{G}(a_2, b_2)$ 

donde  $a_1$ ,  $a_2$ ,  $b_1$  y  $b_2$  son hiperparámetros que se pueden fijar o bien, se pueden considerar a su vez aleatorios.

• Sean  $S_k = \sum_{i=1}^k Y_i$  y  $S_k^c = S_n - S_k$ . Para aplicar el GS, se necesita especificar las distribuciones condicionales posteriores. Las densidades condicionales para k,  $\mu$ ,  $\lambda$ , están dadas por:

$$\mu|y,k \sim \mathcal{G}(a_1 + S_k, k + b_1)$$

$$\lambda|y,k \sim \mathcal{G}(a_2 + S_k^c, n - k + b_2)$$

$$k|y,\mu,\lambda \sim \frac{L(Y|k,\mu,\lambda)}{\sum_{j=1}^n L(Y|j,\mu,\lambda)}$$

#### Aplicación: Análisis de punto de cambio VI

con función de verosimilitud

$$L(Y|k, \mu, \lambda) = e^{k(\lambda - \mu)} \left(\frac{\mu}{\lambda}\right)^{S_k}.$$

Por ejemplo, para encontrar la distribución de  $\mu|y,k$  usamos la siguiente relación:

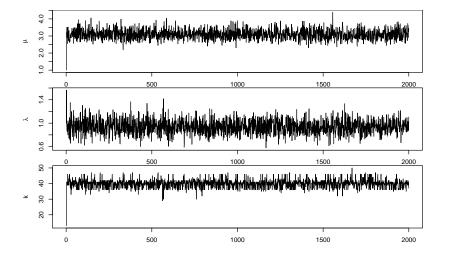
$$\begin{array}{ll} \pi(\mu|y,k) & \propto & \pi(y|k,\mu)\pi(k,\mu) = \pi(y|k,\mu)\pi(k)\pi(\mu) \\ \\ & \propto & \prod_{i=1}^k e^{-\mu}\mu^{y_i}e^{-b_1\mu}\mu^{a_1-1} \\ \\ & \propto & e^{-k\mu-b_1\mu}\mu^{S_k+a_1-1} \end{array}$$

que corresponde al kernel de una distribución  $\mathcal{G}(S_k + a_1, k + b_1)$ , por lo que  $\mu|y, k \sim \mathcal{G}(S_k + a_1, k + b_1)$ .

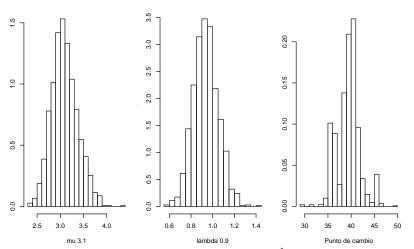
#### Aplicación: Análisis de punto de cambio VII

```
n <- length(v) #longitud de los datos
m <- 2000
mu <- lambda <- k <- numeric(m)
L <- numeric(n)
k[1] <- sample(1:n,1) #valor inicial del punto de cambio (seleccionado al azar)
mu[1] <- 1
lambda[1] <- 1
b1 <- b2 <- 1
for (i in 2:m) {
            mu[i] \leftarrow rgamma(1, shape = al + sum(y[1:k[i-1]]), rate = k[i-1] + bl) #genera mu
        lambda[i] \leftarrow rgamma(1, shape = a2 + sum(y) - sum(y[1:k[i-1]]), rate = n - k[i-1] + b2) #genera lambda
for(| in 1:n) {
L[j] \leftarrow exp((lambda[i]-mu[i])*j) * (mu[i]/lambda[i])^sum(y[1:j])
L <- L/sum(L)
k[i] <- sample(1:n,prob=L,size=1)
```

```
par (mfrow=c(3,1))
par (mar=c(2,4,0,1))
plot (mu, type="1", ylab=expression (mu))
plot(lambda, type="1", ylab=expression(lambda))
plot(k, type="1", ylab="k")
```



```
# histograms from the Gibbs sampler output
b <- 500 #burn-in
par(mar=c(4,3,0,1))
par(mfrow=c(1,3))
label1 <- paste("mu", round(mean(mu[b:m]), 1))
label2 <- paste("lambda", round(mean(lambda[b:m]), 1))
hist(mu[b:m], main="", xlab=label1, breaks = "scott", prob=TRUE) #mu posterior
hist(lambda[b:m], main="", xlab=label2, breaks = "scott", prob=TRUE) #lambda posterior
hist(k[b:m], breaks = min(k[b:m]):max(k[b:m]), prob=TRUE, main="", xlab = "Punto de cambio")</pre>
```



Entonces el punto de cambio está alrededor de  $\hat{k}=40$ , que corresponde al año 1851+40=1891. De 1851 a 1890 la media

#### Aplicación: Análisis de punto de cambio XI

Poisson es alrededor de  $\hat{\mu}\approx 3.1$  y del año 1891 hacia adelante la media es  $\hat{\lambda}\approx 0.93$ 

**GS Software** 

- BUGS (Bayesian Analysis using Gibbs Sampler) es un programa que permite realizar el análisis Bayesiano de modelos estadísticos complicados, utilizando métodos MCMC y extraer muestras de distribuciones posteriores. Fue desarrolladodel 89 al 97 por un equipo en Cambridge dirigido por David Spiegehalter.
- De este modo, nos podemos enfocar en el modelado estadístico, que es el principal interés, y dejar los cálculos a la computadora.
   La sintaxis de \*BUGS es muy similar a la de R, pero hay algunas diferencias que hay que notar.
- BUGS tiene dos sabores: WinBUGS, versión para Windows con interfaz gráfica y OpenBUGS, que es la versión de consola, abierta y extendida a más plataformas. Ahora también tiene su versión GUI.
- WinBUGS ya no se actualiza y se ha sustituído principalmente por OpenBUGS. BRugs y R20penBUGS son paquetes interfaz entre OpenBUGS y R.
- Una buena manera de aprender lo básico de WinBUGS es ver: WinBUGS - The Movie

#### Software relacionado al GS II

- JAGS (Just Another Gibbs Sampler) es otro proyecto independiente de BUGS para hacer análisis de modelos jerárquicos bayesianos usando MCMC.
- Stan Stan es un lenguaje y librerias para obtener inferencia Bayesiana utilizando muestreo MCMC, utilizando como interfase a R, pero también puede ser Python, MATLAB, JULIA y Stata.

#### Modelado en BUGS

- Para llevar a cabo un análisis en BUGS, se requiere considerar la siguiente secuencia de acciones
  - Definir el modelo
  - Incluir los datos observados
  - O Dar valores iniciales para los parámetros.
  - Estimar el modelo
  - 6 Realizar evaluación de diagnósticos.
  - Analizar los resultados
- A continuación seguiremos el procedimiento en WinBUGS a través de un ejemplo sencillo, y lo haremos también a través de la interfase a R utilizando BRugs.

• Paso 1: Consideremos un modelo normal conjugado:

$$y|\theta \sim \mathcal{N}\left(\theta, \sigma^2\right)$$

donde  $\sigma^2$  es conocido, y la prior de la media  $\theta$  también es normal,

$$\theta | \eta \sim \mathcal{N}\left(m, \tau^2\right)$$

donde m y  $\tau$  son hiperparámetros conocidos ( $\eta=(m,\tau)$ ), entonces la posterior de  $\theta|y$  también es normal:

$$\theta|y \sim \mathcal{N}\left(\frac{\sigma^2 m + \tau y}{\sigma^2 + \tau^2}, \frac{\sigma^2 \tau^2}{\sigma^2 + \tau^2}\right)$$
$$= \mathcal{N}\left(Bm + (1 - B)y, B\tau^2\right)$$

donde  $B=\frac{\sigma^2}{\sigma^2+\tau^2}\in(0,1)$ , se puede interpretar como un factor de compresión de la media.

#### Ejemplo 3: Modelo normal conjugado II

• En estadística Bayesiana, es común pensar más en términos del concepto de *precisión* que de varianza.

#### Precisión

La precisión es el recíproco de la varianza: si  $\sigma^2$  es la varianza, la precisión es  $1/\sigma^2$ .

• Si ahora se toma una muestra aleatoria  $y_1, \ldots, y_n$ , entonces la información de la muestra se puede combinar en la estadística suficiente  $\bar{y}$ , y

$$\theta | \bar{y} \sim \mathcal{N} \left( \frac{\sigma^2 m + \tau^2 \bar{y}}{\sigma^2 / n + \tau^2}, \frac{\frac{\sigma^2}{n} \tau^2}{\sigma^2 / n + \tau^2} \right)$$
$$= \mathcal{N} \left( \frac{\sigma^2 m + n \tau \bar{y}}{\sigma^2 + n \tau^2}, \frac{\sigma^2 \tau^2}{\sigma^2 + n \tau^2} \right)$$

• Paso 2 y 3: Por ejemplo, si  $m=2, \ \tau=\sigma=1$  y  $\bar{y}=6$ , podemos obtener la distribución posterior para diferentes tamaños de la muestra n.

### Ejemplo 3: Modelo normal conjugado III

Aunque este modelo está resuelto de manera analítica, se puede usar como referencia para aprender WinBUGS.

Para ejecutar este modelo desde R usando BRugs, seguimos el mismo proceso.

Paso 4: Escribimos el modelo a un archivo, y se "checa" el modelo:

### Ejemplo 3: Modelo normal conjugado IV

② Cargamos los datos y los valores iniciales a un archivo. Cargamos los datos del modelo. Aquí suponemos que n=10.

```
datos <- "list(ybar=6, mu=2, sigma2=1, tau2=1, n=10)"
writeLines(datos, con = "datos.txt")
vinit <- "list(theta=0)"
writeLines(vinit, con = "vinit.txt")
modelData("datos.txt")
data loaded</pre>
```

A continuación se compila el modelo y se inicializa la cadena.

```
modelCompile() #compilación del modelo
model compiled
modelInits("vinit.txt") #Se da un valor inicial para el modelo o se puede usar modelGenInits() para valor
Initializing chain 1:
model is initialized
```

Gibbs Sampler

### Ejemplo 3: Modelo normal conjugado V

• Ahora ejecutamos el modelo. Por ejemplo, consideramos un burn-in de 1,000 observaciones y generamos 10,000 datos adicionales. Necesitamos monitorear a  $\theta$ :

```
modelUpdate (1000) #burn-in

1000 updates took 0 s
samplesSet("theta") #Define el monitor de la variable
monitor set for variable 'theta'
modelUpdate (10000) #Generamos n iteraciones

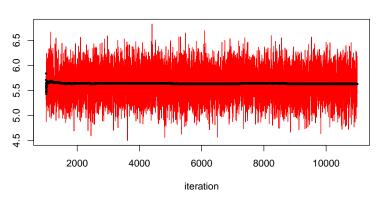
10000 updates took 0 s
samplesStats("theta") #podemos usar "*" o un vector con los nombres de las variables a monitorear entre co
mean sd MC_error val2.5pc median val97.5pc start sample
theta 5.635 0.3028 0.003174 5.046 5.638 6.228 1001 10000
```

samplesStats devuelve los valores de la muestra obtenida

#### Paso 5: Podemos ahora generar las gráficas del modelo

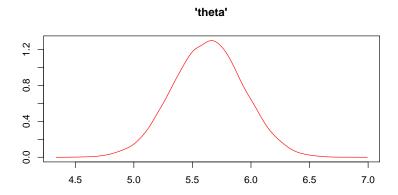
```
a <- samplesHistory("*",mfrow=c(1,1),ask = F)
points(1001:11000,cumsum(a$theta)/(1:length(a$theta)),pch=16,cex=0.5)</pre>
```





samplesDensity("theta", mfrow=c(1,1))

## Ejemplo 3: Modelo normal conjugado VII



#### Ejemplo 3 con Modelo normal conjugado con R20penBUGS I

Ejecutamos el mismo ejemplo previo usando el paquete R20penBUGS. El modelo se escribe con la misma sintaxis que antes y está en el archivo Modelo.txt.

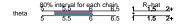
Este programa da una salida ligeramente diferente a la de OpenBUGS con información adicional.

## Ejemplo 3 con Modelo normal conjugado con R20penBUGS II

```
library (R2OpenBUGS)
setwd("~/Dropbox/Public/SimS18-I/bugs/modelo1/") #define el directorio de trabajo.
datos <- list(vbar=6, mu=2, sigma2=1, tau2=1, n=10) #define los valores de los datos
inits <- function(){list(theta = 0)} #se requiere función para tener inicializadas múltiples cadenas
mod.sim <- bugs(datos, inits, model.file = "Modelo.txt",</pre>
                                parameters = c("theta"), n.chains = 20,
print (mod.sim)
Inference for Bugs model at "Modelo.txt",
Current: 20 chains, each with 5000 iterations (first 1000 discarded)
Cumulative: n.sims = 80000 iterations saved
         mean sd 2.5% 25% 50% 75% 97.5% Rhat n.eff
         5.6 0.3 5.0 5.4 5.6 5.8 6.2 1 80000
deviance 1.8 2.6 -0.5 -0.1 0.9 2.8 8.7
                                            1 80000
For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
DIC info (using the rule, pD = Dbar-Dhat)
pD = 0.9 \text{ and } DIC = 2.7
DIC is an estimate of expected predictive error (lower deviance is better).
plot (mod.sim)
```

# Ejemplo 3 con Modelo normal conjugado con R20penBUGS III

Bugs model at "Modelo.txt", 20 chains, each with 5000 iterations (first 1000 discarded)



medians and 80% intervals





- La interpretación de esta salida es muy similar a la que se obtendría en Stan también.
- En la salida de los resultados del modelo, se pueden ver los siguientes elementos para cada parámetro estimado:
  - mean es la media posterior estimada, calculada como el promedio de las salidas de la cadena.
  - sd es la desviación estándar de las observaciones, que conforme el tamaño de muestra tiende a infinito, se acerca a la desviación estándar posterior del parámetro.
  - 2.5%, 25%, 50%, 75%, 97.5% son cuantiles de la distribución posterior
  - Rhat corresponde al factor de reducción potencial de escala  $\hat{R}$  que se mencionó antes, que vale  $\hat{R}=1$  cuando todas las cadenas se han mezclado

## Explicación de los elementos de la salida de OpenBUGS II

 n.eff es el tamaño de muestra efectivo, que tiene una fórmula dada por:

$$\hat{n}_{eff} = \frac{mn}{1 + 2\sum_{i=1}^{T} \hat{\rho}_t}$$

 $\rho_t$  es la autocorrelación de la sucesión  $\{X_n\}$  en el rezago t, n es el número de simulaciones de cada cadena y m es el número de cadenas simuladas, y T es el primer entero impar para el cual  $\hat{\rho}_{T+1} + \hat{\rho}_{T+2} < 0$ . (ver detalles en Gelman, et.al BDA3, sección 11.5) Basta tomar como referencia que  $\hat{n}_{eff} \geq 10m$  es un síntoma de que la cadena se ha mezclado adecuadamente

- En la práctica, la selección estadística apropiada de un mejor modelo en una colección de modelos jerárquicos es problemática, debido a la ambigüedad del tamaño de esos modelos.
- Spiegelhater et al. (2002) sugiere una generalización del Criterio de Información de Akaike (AIC) que se basa en la distribución posterior de la estadística devianza:

$$D(\theta) = -2\log f(y|\theta) + 2\log h(y)$$

donde  $f(y|\theta)$  es la verosimilitud de los datos observados, dado el parámetro  $\theta$  y h(y) es alguna función estandarizada que sólo depende de los datos (y que no tiene importancia en la selección del modelo).

• En este enfoque, el ajuste de un modelo queda resumido por la esperanza posterior de la devianza,  $\bar{D} = E_{\theta|y}[D]$ , mientras que la complejidad del modelo se captura en el número efectivo de parametros  $p_D$  que hay en el modelo:

$$p_D = E_{\theta|y}[D] - D(E_{\theta|y}[\theta]) = \bar{D} - D(\bar{\theta})$$

#### Selección de Modelo II

• El Criterio de Información de la Devianza (DIC) se define como

$$DIC = \bar{D} + p_D = 2\bar{D} - D(\bar{\theta})$$

- Valores pequeños de DIC indican un modelo que ajusta mejor.
- Comparando modelos, los que tienen las máximas diferencias son los que tienen mayor cambio y mejoran más.

## Ejemplo: Modelo normal conjugado y coda. I

- coda significa Convergence and diagnostics analysis for MCMC. El paquete coda, permite hacer análisis de salida y diagnósticos para MCMC. coda trata de responder a la pregunta: ¿ha tenido el sampler suficiente adaptación (burn-in) para justificar que la muestra proviene de la posterior de interés?
- Un paquete similar es boa que sirve para evaluar la convergencia de cadenas MCMC. boa surge de reescribir de las funciones y la interfase de CODA, que es la versión original del paquete de R coda. El orden es:

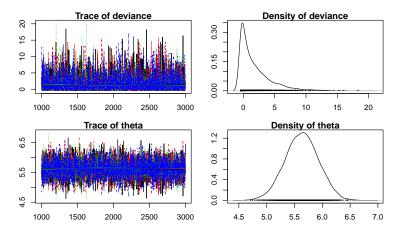
$$\mathtt{CODA} o \mathtt{boa} o \mathtt{coda}$$

 Usualmente tenemos que analizar las muestras después del periodo burn-in para detectar algún tipo de anomalía. El paquete coda ofrece algunas funciones de diagnóstico para para facilitar el análisis.

### Ejemplo: Modelo normal conjugado y coda. II

 Podemos obtener la traza y las densidades de las variables de interés del modelo. Estas ya las obteníamos desde antes.

#### Ejemplo: Modelo normal conjugado y coda. III



• Otra herramienta de diagnóstico que ya mencionamos antes son los gráficos de Gelman, que grafican el valor de  $\hat{R}$ .

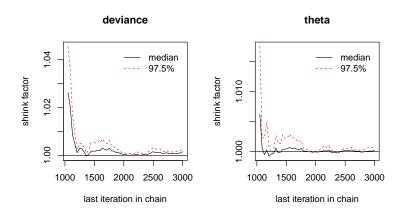
### Ejemplo: Modelo normal conjugado y coda. IV

- La gráfica de Gelman mide si hay una diferencia significativa entre la varianza al interior de varias cadenas y la varianza entre las cadenas.
- gelman.diag devuelve el factor de reducción para cada parámetro. Un factor de 1 significa que la intravarianza y la intervarianza son iguales. Valores grandes denotan que hay diferencias notables entre las cadenas.

# Ejemplo: Modelo normal conjugado y coda. V

```
gelman.diag(read.bugs(mod.sim2))
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Potential scale reduction factors:
        Point est. Upper C.I.
deviance 1 1
Multivariate psrf
gelman.plot(read.bugs(mod.sim2))
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting theta ... 2000 valid values
```

#### Ejemplo: Modelo normal conjugado y coda. VI



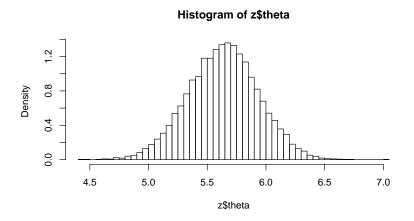
De acuerdo al profesor Charles Geyer, de la Universidad de Minnesota, el periodo de burn-in es completamente innecesario

#### Ejemplo 3: Modelo normal conjugado con JAGS y rjags I

Para ejecutar el programa ahora usamos JAGS El archivo con el modelo siguie teniendo la misma especificación y está en Modelo.txt. Obviamente primero se requiere tener instalado JAGS en la computadora.

```
library (rjags)
Linked to JAGS 4.3.0
Loaded modules: basemod, bugs
setwd("~/Dropbox/Public/SimS18-I/bugs/modelo1/") #define el directorio de trabajo.
iags <- iags.model("Modelo.txt",</pre>
                    data = list(ybar=6, mu=2, sigma2=1, tau2=1, n=10), #define los valores de los datos
Compiling model graph
  Allocating nodes
   Observed stochastic nodes: 1
   Total graph size: 9
Initializing model
update (jags, 1000)
z <- jags.samples(jags,c("theta"),1000)</pre>
hist (z$theta, breaks = 50, prob=T)
```

#### Ejemplo 3: Modelo normal conjugado con JAGS y rjags II



Información adicional y ejemplos se pueden encontrar en este tutorial.

#### Ejemplo 4: Simulación de transformación de variable aleatoria I

Problema: Si tenemos  $Z \sim N(0,1)$  y transformamos a  $Y = (2Z+1)^3$ . Queremos obtener la distribución de Y, su media y P(Y>k) para alguna k, por ejemplo k=10. El problema es difícil de resolver de manera analítica pero es fácil de simular.

#### Ejemplo 4: Simulación de transformación de variable aleatoria II

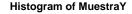
```
Modelo2 <- "
Z \sim dnorm(0,1) #Hay que considerar que en BUGS se lee la precisión, no la varianza
P10 <- step(Y-10) \#step es la indicadora de la condición > 0 Suponemos k=10
writeLines (Modelo2, con="Modelo2.txt")
modelCheck("Modelo2.txt")
model is syntactically correct
modelCompile()
model compiled
modelGenInits() #inicializa al azar
initial values generated, model initialized
modelUpdate(10000) #burn-in
10000 updates took 0 s
samplesSet(c("Y", "P10"))
monitor set for variable 'Y'
monitor set for variable 'P10'
modelUpdate (50000)
50000 updates took 0 s
```

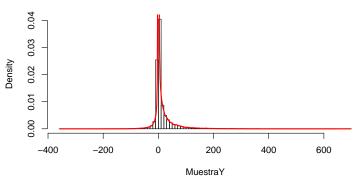
## Ejemplo 4: Simulación de transformación de variable aleatoria III

#### Podemos ver las respuestas a las preguntas:

Distribución:

```
MuestraY <- samplesSample ("Y")
hist(MuestraY)prob=T,breaks=100)
lines(density(MuestraY),col="red",lwd=2)</pre>
```





#### Ejemplo 4: Simulación de transformación de variable aleatoria IV

Media y varianza

```
        samplesStats("*")

        mean
        sd MC_error val2.5pc median val97.5pc start sample

        P10
        0.283
        0.4505
        0.001822
        0.00
        0.0000
        1.0
        10001
        50000

        Y
        13.000
        39.3000
        0.163500
        -24.52
        0.9742
        119.2
        10001
        50000
```

• y la Probabilidad de que sea mayor que 10:

```
samplesStats("P10")$mean
[1] 0.283
```

#### Simplificación de proceso I

Como pueden ver, varios pasos se repiten con frecuencia. Entonces, podemos tratar de escribir una función que haga todo el proceso sin tener que reescribirlo todo el tiempo.

Entonces el ejercicio previo lo pudimos haber ejecutado de la siguiente manera:

## Simplificación de proceso II

```
run.model(Modelo2, muestras=c("Y","P10"),longcadena = 50000)

model is syntactically correct
model compiled
initial values generated, model initialized
5000 updates took 0 s
monitor set for variable 'Y'
monitor set for variable 'P10'
50000 updates took 0 s
```

### Ejemplo 5: Estimación de número de reparaciones I

Problema: Tenemos 100 computadoras que requieren reparación. El costo unitario de reparación depende de las piezas que tenga descompuesta la computadora, y usualmente el costo es una variable aleatoria que se puede modelar como una gamma con media 100 y desviación estándar 50. Si tenemos un presupuesto de 10,000 para reparaciones, ¿cuántas reparaciones en promedio podemos hacer?

Recuerden que la distribución  $\mathcal{G}\left(a,b\right)$  tiene media  $\frac{a}{b}$  y varianza  $\frac{a}{b^2}$  Entonces la costo en el ejercicio tiene distribución  $\mathcal{G}\left(4,0.04\right)$ .

Para simular el ejercicio, generamos 100 datos de costos con la distrbución dada. Entonces obtenemos una muestra  $Y_1,\ldots,Y_{100}$  donde  $Y_i$  es el costo de reparar la computadora i. El problema consiste en estimar M tal que  $\sum_{i=1}^M Y_i <= 10,000$  Como podemos ver M es aleatorio completamente.

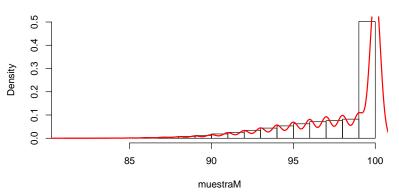
### Ejemplo 5: Estimación de número de reparaciones II

#### Para ver los resultados

## Ejemplo 5: Estimación de número de reparaciones III

#### Ejemplo 5: Estimación de número de reparaciones IV

#### Histogram of muestraM



#### Ejemplo 6: Modelo de Regresión Lineal I

En el siguiente modelo será  $\tau=\frac{1}{\sigma^2}$ . Se supone que la precisión tiene una distribución inicial de la forma gamma $(\epsilon,\epsilon)$ , en la parametrización de la Gamma en BUGS, tiene media  $\frac{\epsilon}{\epsilon}=1$  y varianza  $\frac{\epsilon}{\epsilon^2}=\frac{1}{\epsilon}=10$ .

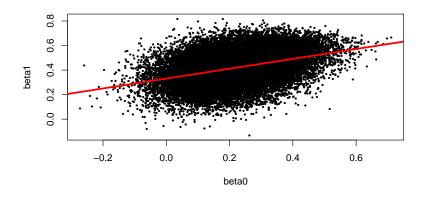
# Ejemplo 6: Modelo de Regresión Lineal II

```
modelo4 <- "
# priors para los parámetros de regresión
beta1 ~ dnorm(0,100)
# prior para la precisión del parámetro
sigma <- 1/sqrt(tau)
run.model (modelo4, muestras=c("beta0", "beta1", "sigma"),
              datos = list(x = c(1.0, 1.5, 1.5, 1.5, 2.5, 4.0, 5.0, 5.0, 7.0,
                                                         8.0, 8.5, 9.0, 9.5, 9.5, 10.0, 12.0, 12.0, 13.0,
                                                         13.0, 14.5, 15.5, 15.5, 16.5, 17.0, 22.5, 29.0,31.5),
                                           v = c(1.80, 1.85, 1.87, 1.77, 2.02, 2.27, 2.15, 2.26, 2.47,
                                                          2.19, 2.26, 2.40, 2.39, 2.41, 2.50, 2.32, 2.32, 2.43,
                                                          2.47, 2.56, 2.65, 2.47, 2.64, 2.56, 2.70, 2.72, 2.57),
model is syntactically correct
data loaded
model compiled
initial values generated, model initialized
6000 updates took 0 s
monitor set for variable 'beta0'
monitor set for variable 'betal'
monitor set for variable 'sigma'
30000 updates took 0 s
```

#### Modelo de Regresión Lineal I

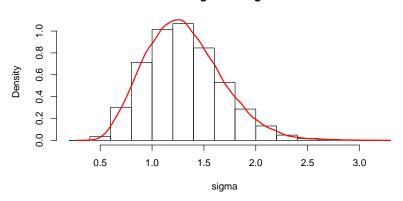
Después de obtener las iteraciones analizamos los resultados. Noten en la gráfica conjunta de  $(\beta_0,\beta_1)$  la fuerte correlación que hay entre los dos parámetros La correlación se puede eliminar si la variable x se centra alrededor de su propia media.

## Modelo de Regresión Lineal II



hist(sigma, probability = T)
lines(density(sigma),col="red",lwd=2)

#### Histogram of sigma



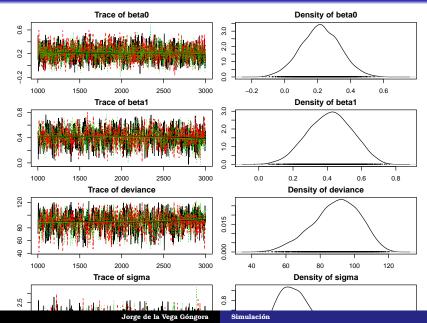
Versión con OpenBUGS para incluir análisis de coda

#### Modelo de Regresión Lineal IV

```
par (mar=c(2,2,2,1))
plot(read.bugs (mod.siml))

Abstracting beta0 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting sigma ... 2000 valid values
```

## Modelo de Regresión Lineal V



## Ejemplo 6: Modelo de Regresión Lineal I

```
Par (mar=c(3,3,1,1))
gelman.plct (read.bugs (mod.sim1))

Abstracting beta0 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting beta0 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting deviance ... 2000 valid values
Abstracting beta0 ... 2000 valid values
Abstracting beta0 ... 2000 valid values
Abstracting beta1 ... 2000 valid values
Abstracting beta0 ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting sigma ... 2000 valid values
Abstracting sigma ... 2001 valid values
```

## Ejemplo 6: Modelo de Regresión Lineal II

