

Introducción a los métodos estadísticos

bayesianos en Ecología

Pablo Inchausti

Programa del curso:

1. Introducción general
2. Elementos básicos del análisis bayesiano ←
3. Análisis bayesiano I
4. Análisis bayesiano II
5. Modelos bayesianos jerárquicos

2. Elementos básicos del análisis bayesiano

- a. Máximo de verosimilitud. Práctico 01
- b. Markov Chain Monte Carlo (MCMC).
- c. Algunas simulaciones ilustrativas.
- d. ¿Cómo vamos a trabajar en este curso?
- e. ¿Por qué $Pr(\text{parámetros} | \text{datos}) \approx Pr(\text{datos} | \text{parámetros}) * Pr(\text{parámetros})$?

a. Máximo de verosimilitud. $Pr(A|B)*Pr(B)=Pr(B|A)*Pr(A)$

Aplicar el teorema de Bayes al análisis de datos requiere postular un modelo por ej: Poisson (μ_Y).

Cuando A: valor de μ_Y y B: datos, la ecuación anterior se transforma en: $Pr(\mu_Y|datos) = \frac{Pr(datos|\mu_Y)}{Pr(datos)} * Pr(\mu_Y)$

$Pr(\mu_Y|datos)$: Distr. posterior $Pr(\mu_Y)$: Distr previa.

$Pr(datos|\mu_Y)$: prob. de obtener los datos
observados dado el valor de μ_Y . VEROSIMILITUD

Y un poco después llegamos a que:

Dist posterior

Verosimilitud

Dist previa

$$Pr(\mu_Y | \text{datos}) \approx Pr(\text{datos} | \mu_Y) * Pr(\mu_Y)$$

Evidencia
actual

Información
en los datos

Evidencia
previa

Permite “refinar” el conocimiento previo sobre el valor de μ_Y empleando la información de los datos para generar la distr. a posteriori del parámetro μ_Y .

La dist posterior de μ_Y contiene el estado de conocimiento actual de sus valores plausibles.

Si los datos no contienen mucha/nueva evidencia,

la Dist. posterior ~ Dist previa



Cuando los datos tengan nueva evidencia,

la Dist. posterior será diferente de Dist previa

En esta ecuación:

$$Pr(\mu_Y | \text{datos}) = \frac{Pr(\text{datos} | \mu_Y)}{Pr(\text{datos})} * Pr(\mu_Y)$$

La verosimilitud $Pr(\text{datos} | \mu_Y)$ se refiere al valor de μ_Y que más probablemente podría haber generado las observaciones (si el modelo fuese correcto).

Es un método genérico de estimación de parámetros de modelos también usado en estadística frecuentista.

La verosimilitud es la solución al problema general:
Dado los datos y suponiendo un modelo (Poisson),
¿cómo estimar el valor del parámetro (μ_Y) que podría
“haber generado los k datos observados”?

La solución:

Fisher (1922) definió una función $L(\mu_Y; y_1, y_2, \dots, y_k)$ en lugar de la probabilidad de obtener los datos observados que NO puede calcularse.



La verosimilitud se interpreta como una medida de la “creencia racional” de que μ_Y tenga un valor ej. 2.3) dados todos los datos y el modelo Poisson.

Para entenderlo intuitivamente, veamos un ejemplo.

Se tienen k conteos de abundancia de algún insecto.

Supongamos como razonable al modelo Poisson:

$$Pr(Y = y_i) = \frac{e^{-\mu_Y} * \mu^{y_i}}{y_i!}$$

La probabilidad de haber obtenido simultáneamente los k valores de conteos es: $Pr((Y_1 = y_1) \cap (Y_2 = y_2) \cap, \dots, \cap (Y_k = y_k))$

Suponiendo que las muestras son independientes:

$$\prod_{i=1}^k Pr(Y_i = y_i) \equiv \prod_{i=1}^k \text{Poisson}(\mu_Y; y_i)$$

→ probabilidad de obtener todos los datos, dado el modelo Poisson y el valor de su parámetro μ_Y .

Fisher postuló que L_{Tot} (likelihood) era proporcional a la probabilidad de obtener los datos observados dado el valor del parámetro μ_Y , i.e. a la $Pr(\text{Posterior}(\mu_Y | \text{datos}))$.

$$Pr(\mu_Y | \text{datos}) \approx Pr(\text{datos} | \mu_Y) * Pr(\mu_Y) \Rightarrow Pr(\mu_Y | \text{datos}) \approx Pr(\text{datos} | \mu_Y)$$

Dada esta proporcionalidad, maximizar L_{Tot} permite estimar los valores del parámetro del modelo Poisson que más probablemente podrían haber generado los datos observados.

subtle

Recordemos : $Pr(\text{datos} | \mu_Y) \propto L_{\text{Tot}} = \prod_i^k \text{Likelihood}(\mu_Y; y_i)$

Por tanto: $L_{\text{Tot}} = \prod_i^k \text{Likelihood}(\mu_Y; y_i) = \prod_i^k \text{Poisson}(\mu_Y; y_i)$

Es equivalente (y más simple) maximizar $\log L(\mu_Y | \text{data})$:

$$L_{\text{Tot}} = \prod \text{Likelihood}(\mu_Y; y_i) \rightarrow \log(L_{\text{Tot}}) = \sum \log(\text{Likelihood}(\mu_Y; y_i))$$

Como \log es una monotónica creciente, $\log(\text{Poisson}(\mu_Y | \text{datos}))$ se maximiza para el mismo valor de μ que $\text{Poisson}(\mu_Y | \text{datos})$.

Reemplazando $\Pr(Y=y_i) = \frac{e^{-\mu_Y} * \mu_Y^{y_i}}{y_i!}$ en $\text{Likelihood}(\mu_Y; y_i)$:

$$\log(L_{\text{Tot}}) = \sum \log(\text{Poisson}(\mu_Y; y_i)), \text{ se obtiene: } \sum \log\left(\frac{e^{-\mu_Y} * \mu_Y^{y_i}}{y_i!}\right)$$

$$\text{y: } L_{\text{Tot}} \approx \sum -\mu_Y \log(e) + y \log(\mu_Y) - \log(y!)$$

Para buscar el valor de μ_Y que maximiza $\log(L_{\text{Tot}})$, se iguala su derivada a cero.

$$\frac{d \log(L_{\text{Tot}}(\mu))}{d \mu_Y} = 0$$

Raramente (ésta es una) la maximización anterior produce una expresión simple.

$$\mu_Y = \frac{\sum y}{k}$$

Fisher (1922) demostró que a partir de la **2da derivada de L_{tot}** $\frac{d^2 L_{\text{Tot}}(\mu)}{d\mu^2} = 0$ se puede calcular el SE del parámetro estimado. ←ésta es una propiedad genérica de ML.

Además, también demostró que los estimadores de máximo de verosimilitud son insesgados, consistentes, eficientes y tienen asintóticamente distr. Normal.

Pero en general estas ecuaciones MaxLik se resuelven de forma numérica, especialmente en el contexto de la estimación bayesiana. Veámoslo.

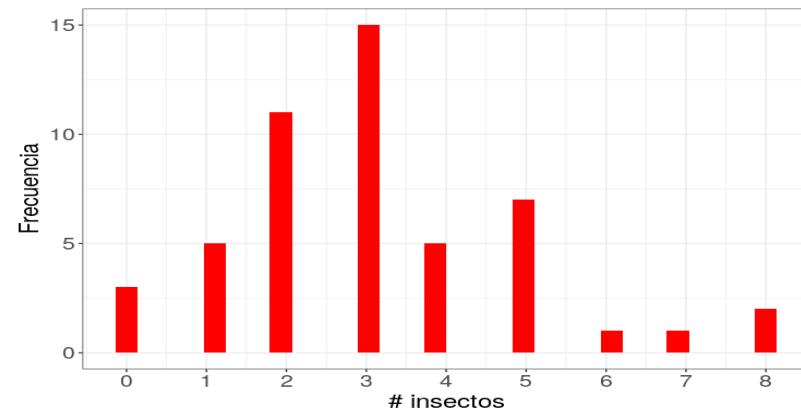
Fuerza bruta numérica en R:

```
> DF=read.csv("Practico 01 counts.csv", header=T)
> str(DF)
'data.frame': 50 obs. of 1 variable:
 $ insectos: int 2 3 5 2 1 3 3 4 3 3 ...
> table(DF$insectos)
```

0	1	2	3	4	5	6	7	8
3	5	11	15	5	7	1	1	2

Vamos a suponer:
 $Y \sim \text{Poisson}(\mu_Y)$

Abundancias de insectos en cada una de k=50 plantas.



y a estimar numéricamente el valor de μ_Y que maximiza L_{tot} : el valor que más probablemente podría haber generado estos conteos.

$$L_{\text{Tot}} = \sum \left(\frac{e^{-\mu_Y} * \mu^{y_i}}{y_i!} \right)$$

Esto es la Prob. Poisson para cada dato Y_i para un valor de $\mu_Y=1$.

```
> dpois(DF$insectos, lambda=1) # dens Poisson para mu=1
[1] 1.839397e-01 6.131324e-02 3.065662e-03 1.839397e-01 3.678794e-01 6.131324e-02
[7] 6.131324e-02 1.532831e-02 6.131324e-02 6.131324e-02 1.532831e-02 1.532831e-02
[13] 3.678794e-01 1.532831e-02 1.839397e-01 3.065662e-03 6.131324e-02 3.065662e-03
[19] 1.839397e-01 6.131324e-02 1.839397e-01 3.065662e-03 6.131324e-02 1.839397e-01
[25] 9.123994e-06 3.678794e-01 6.131324e-02 6.131324e-02 6.131324e-02 6.131324e-02
[31] 5.109437e-04 3.065662e-03 1.839397e-01 9.123994e-06 1.839397e-01 3.678794e-01
[37] 6.131324e-02 1.532831e-02 3.678794e-01 3.678794e-01 3.678794e-01 3.065662e-03
[43] 1.839397e-01 3.678794e-01 6.131324e-02 1.839397e-01 1.839397e-01 6.131324e-02
[49] 7.299195e-05 3.065662e-03
```

Para $\mu_Y=1$

log prob Poisson para cada dato y $\mu_Y=1$.

```
> log(dpois(DF$insectos, lambda=1)) # log dens Poisson para mu=1
[1] -1.693147 -2.791759 -5.787492 -1.693147 -1.000000 -2.791759 -2.791759 -4.178054
[9] -2.791759 -2.791759 -4.178054 -4.178054 -1.000000 -4.178054 -1.693147 -5.787492
[17] -2.791759 -5.787492 -1.693147 -2.791759 -1.693147 -5.787492 -2.791759 -1.693147
[25] -11.604603 -1.000000 -2.791759 -2.791759 -2.791759 -2.791759 -7.579251 -5.787492
[33] -1.693147 -11.604603 -1.693147 -1.000000 -2.791759 -4.178054 -1.000000 -1.000000
[41] -1.000000 -5.787492 -1.693147 -1.000000 -2.791759 -1.693147 -1.693147 -2.791759
[49] -9.525161 -5.787492
> sum(log(dpois(DF$insectos, lambda=1)))
[1] -170.2173
```

Cuya suma es:

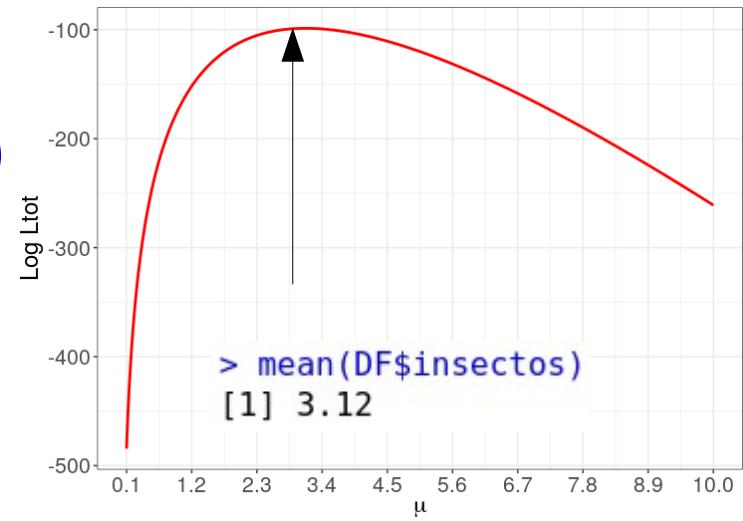
Ahora creamos una función para hacer los mismos cálculos para muchos valores de μ_Y :

```
L_Poisson=function(lambda)
  {sum(log(dpois(DF1$insectos,lambda)))}
```

Buscamos el valor de μ_Y que maximiza L_{tot} :

```
DF1=data.frame(media=
  seq(from=0.1,to=10,by=0.01))
DF1$Ltot=sapply(DF1$media,L_Poisson)
```

Hay métodos mejores para encontrar el valor de μ_Y que maximiza L_Poisson.



b. Markov Chain Monte Carlo (MCMC): Recordemos que el objetivo es:

$$Pr(\text{parámetros} | \text{datos}) \approx Pr(\text{datos} | \text{parámetros}) * Pr(\text{parámetros})$$

Ej. conteos modelados como Poisson. $Pr(Y = y_i) = \frac{e^{-\mu_Y} * \mu_Y^{y_i}}{y_i!}$

MCMC: es una clase de métodos que permite obtener la distr. posterior del parámetro μ a partir de la distr. previa (candidatos” plausibles de μ **antes** de ver los datos) y la verosimilitud (información contenida en los datos).

Propuesto por Metropolis et al. (1953) y “redescubierto” por Gelfand & Smith (1990)



Se llama **MCMC** porque contiene una **cadena de Markov** genera los valores candidatos de μ_Y y porque el criterio de aceptación de valores de plausibles de μ_Y tiene un componente probabilístico (**Monte Carlo**).

Cadena de Markov: proceso estocástico que genera una secuencia de valores posibles en el que el valor de μ_k solo está relacionado con el de μ_{k-1} (short memory):

Ej. Proceso AR(1): $\mu_{k+1} = \mu_k + \text{random shift} \sim \text{Normal}(0, \sigma)$

σ : regula cuan lejos estaría el siguiente valor posible de μ .

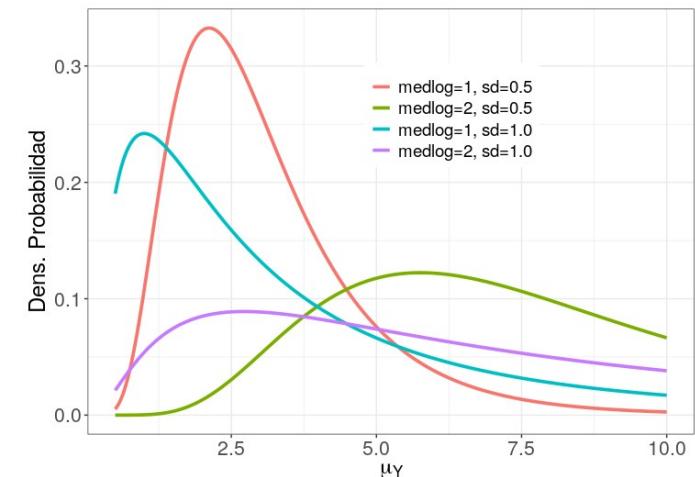
La cadena no converge... ← falta criterio de aceptación de valores

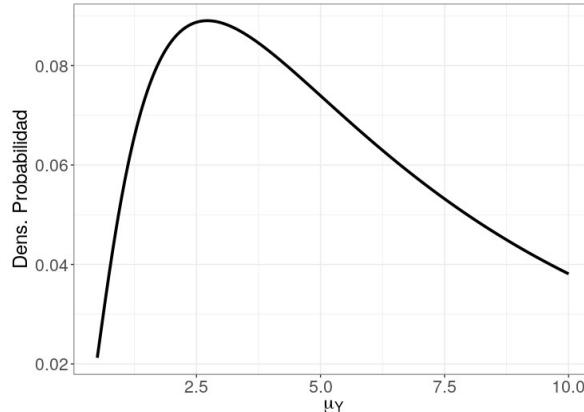
Para conteos Poisson: $Pr(Y = y_i) = \frac{e^{-\mu} * \mu^{y_i}}{y_i!}$

¿Qué sabemos de los valores plausibles de μ ANTES de observar los datos? ← Distribución previa.

- * $\mu_Y > 0$, cualquiera que sean los datos.
- * Datos previos de la misma población/especie o sitio (similares) proporcionarían valores de la media y acaso una distr de frecuencias de ésta.

**Ejemplos de distrib.
previas posibles de μ_Y :**





Por ahora, optemos “a ciegas” por una de estas distr previas:
lognormal(meanlog=2, sdlog=1)

¿Qué modelo emplearemos para la verosimilitud que es
 $\propto \Pr(\mu_Y | \text{data})$? ← Poisson (podría ser otro).



$Y \sim \text{Poisson}(\mu_Y)$.

Distr previa de μ_Y .

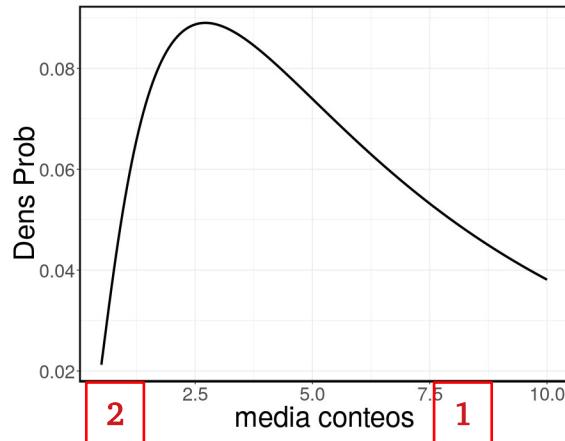
Verosimilitud

$\mu_Y \sim \text{lognormal}(\text{meanlog}=2, \text{sdlog}=1)$

(hiperparámetros)

A partir de todo esto, MCMC genera (“muestrea”) por simulación estocástica la dist posterior de μ_Y .

La versión original de MCMC del algoritmo de Metropolis et al (1953) consiste en :



a. generar 1er candidato de la cadena:

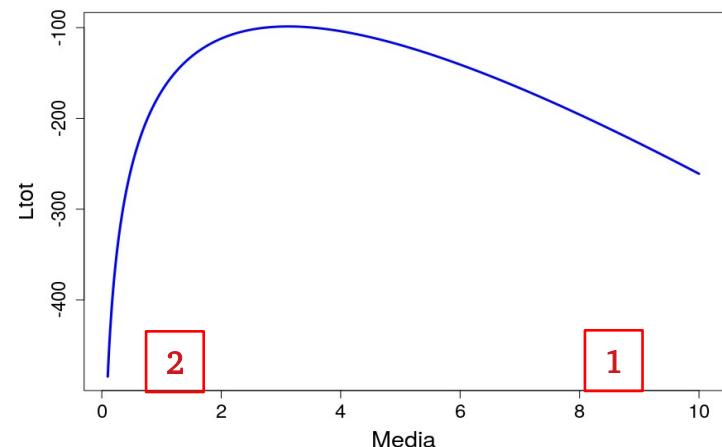
```
> rlnorm(n=1,meanlog=2,sdlog=1)  
[1] 8.485576
```

b. Log.verosimilitud de los datos de conteos para el valor de $\mu_Y=8.845$

```
> LPoisson(lambda=8.485)  
[1] -214.7464
```

c. generar 2do. candidato de la cadena: $\mu_Y=1.353$ y calcular su verosimilitud:

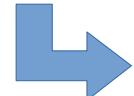
```
> LPoisson(lambda=1.353)  
[1] -172.0982
```



d. calculemos log prob. de obtener los dos valores de $\mu_{Y_1}=8.485$ y $\mu_{Y_2}=1.353$.

```
> log(dlnorm(8.485, meanlog=2, sdlog=1))  
[1] -3.066802  
> log(dlnorm(1.353, meanlog=2, sdlog=1))  
[1] -2.662314
```

$$Pr(\text{Posterior}) \approx \text{Verosimilitud} * Pr(\text{Previa})$$



$$\log(Pr(\text{Posterior})) \approx \log(\text{Verosimilitud}) + \log(Pr(\text{Previa}))$$

e. log prob.
posterior para $\mu_{Y_1}=8.485$ y $\mu_{Y_2}=1.353$.

```
> LPoisson(lambda=8.485) + log(dlnorm(8.485, meanlog=2, sdlog=1))  
[1] -213.9594  
> LPoisson(lambda=1.353) + log(dlnorm(1.353, meanlog=2, sdlog=1))  
[1] -143.3671
```

f. cociente entre log prob posterior de valores de μ_Y :

```
> (LPoisson(lambda=1.353) + log(dlnorm(1.353, meanlog=2, sdlog=1)))/  
+ (LPoisson(lambda=8.485) + log(dlnorm(8.485, meanlog=2, sdlog=1)))  
[1] 0.6700668
```

← combina soporte de los datos y evidencia previa sobre valores de μ_Y .

Ahora que con 2 valores, se necesita un **criterio de aceptación** para decidir si la cadena “se mueve” (acepta o no acepta) de $\mu_1=8.845$ a $\mu_2=1.353$.

g. El criterio de
aceptación es:

$$R = \min\left(1, \frac{\log[\text{verosim}(\mu_2) * \text{Pr previa}(\mu_2)]}{\log[\text{verosim}(\mu_1) * \text{Pr previa}(\mu_1)]}\right)$$

```
> (LPoisson(lambda=1.353) + log(dlnorm(1.353, meanlog=2, sdlog=1)))/  
+   (LPoisson(lambda=8.485) + log(dlnorm(8.485, meanlog=2, sdlog=1)))  
[1] 0.6700668
```

Se acepta μ_2 si: `runif(1, min=0, max=1) < R`

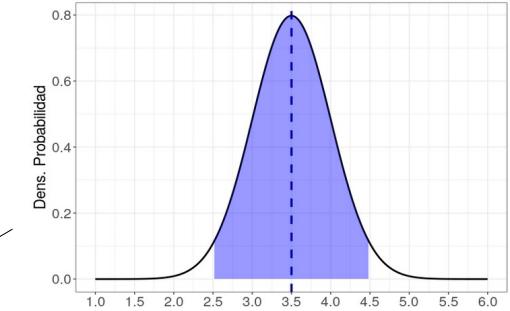
```
> runif(1,min=0, max=1)  
[1] 0.5407793
```

El nuevo valor de la cadena (set de estimados de μ_Y) pasa a ser $\mu_2 = 1.353$.

Por supuesto, se repiten estos 8 pasos muuuchas veces....

Un algoritmo MCMC "básico":

```
MCMC1= data.frame(matrix(nrow=5000,ncol=1));
names(MCMC1)="mean"
MCMC1$mean[1]=3.5
```



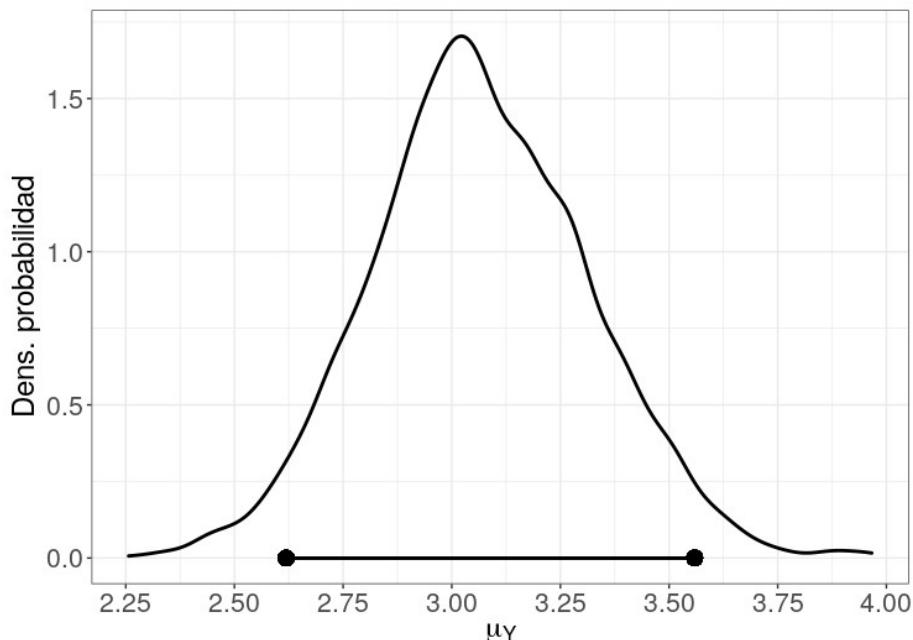
$$\log(Pr(\text{Posterior})) = \log(\text{Verosimilitud}) + \log(\text{Pr}(\text{Previa}))$$

```
for(i in 2:5000){
  current=MCMC1$mean[i-1]
  prop=rnorm(n=1,mean=current, sd=0.5)
  post.prop=sapply(prop, LPoisson) +
    dlnorm(prop,meanlog=0.5,sdlog=0.5, log=T)
  post.curr=sapply(current, LPoisson) +
    dlnorm(current,meanlog=0.5,sdlog=0.5, log=T)
  R=min(1,exp(post.prop)/exp(post.curr))
  MCMC1$mean[i]=ifelse(R > runif(n=1, min=0, max=1), prop, current)}
```

```

> summary(MCMC1$mean)
   Min. 1st Qu. Median     Mean 3rd Qu.    Max.
 2.26    2.90   3.06    3.07   3.23   3.97
> quantile(MCMC1$mean, probs=c(0.025,0.975))
 2.5% 97.5%
2.59  3.57

```



Esta distr posterior μ_Y resume todo lo que es posible conocer sobre este parámetro para estos datos, dado el conocimiento previo.

Int. Cred 95% de μ_Y :
 $\Pr(2.59 < \mu_Y < 3.57) = 0.95$

Se puede demostrar que una cadena de Markov (i.e. la secuencia de valores aceptados del parámetro) así construida converge a la distr posterior del problema, aún cuando NO conozcamos su expresión analítica.

Obtuvimos una solución numérica (particular) del problema, hay que realizar ciertos “aspectos de control de calidad” que veremos luego.

Las variantes mejoradas de MCMC que incluyen Metropolis-Hastings, Gibbs, INLA y Hamiltonian MonteCarlo siguen estos principios básicos.



2. Elementos básicos del análisis bayesiano

- a. Máximo de verosimilitud. Práctico 01
- b. Markov Chain Monte Carlo (MCMC).
- c. Algunas simulaciones ilustrativas.
- d. ¿Cómo vamos a trabajar en este curso?
- e. ¿Por qué $Pr(\text{parámetros} | \text{datos}) \approx Pr(\text{datos} | \text{parámetros}) * Pr(\text{parámetros})$?

c. Algunas simulaciones ilustrativas:

The Markov-chain Monte Carlo

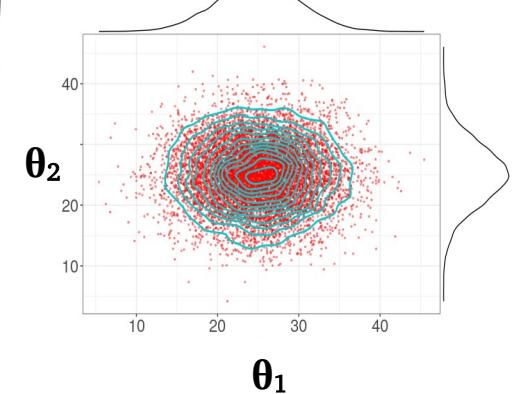
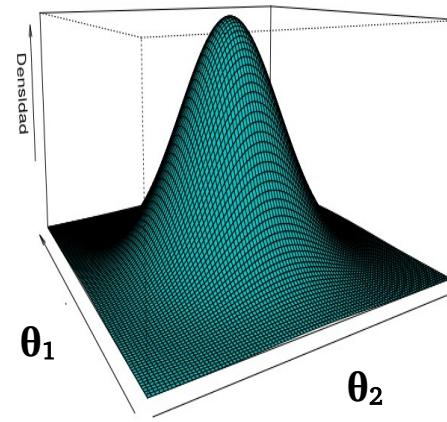
Click on an algorithm below to view interactive demo:

- Random Walk Metropolis Hastings
- Adaptive Metropolis Hastings [1]
- Hamiltonian Monte Carlo [2]
- No-U-Turn Sampler [2]
- Metropolis-adjusted Langevin Algorithm (MALA) [3]
- Hessian-Hamiltonian Monte Carlo (H2MC) [4]
- Gibbs Sampling
- Stein Variational Gradient Descent (SVGD) [5]
- Nested Sampling with RadFriends (RadFriends-NS) [6]
- Differential Evolution Metropolis (Z) [7]

 <https://chi-feng.github.io/mcmc-demo/>

Sea el modelo estadístico BlaBla(θ_1, θ_2)

El objetivo es obtener por simulación las dos distr posteriores marginales de θ_1 y θ_2 .





Acaban de usar el algoritmo Metropolis-Hastings.

El algoritmo de Metropolis et al (1953) que implementamos generaba cambios simétricos

$\Pr(\Theta_k \rightarrow \Theta_{k+1}) = \Pr(\Theta_{k+1} \rightarrow \Theta_k)$ en los potenciales valores de los parámetros a evaluar si aceptar o no.

Esta opción es problemática cuando hay parámetros con límites precisos, por ej. $\sigma > 0$, $-1 < \text{correl.} < 1$.

Hastings (1970) modificó de la regla de aceptación R y resolvió este problema:

$$R = \min\left(1, \frac{\text{Lik}(\text{data}|\text{Par}^B) \text{Previa}(\text{Par}^B) \Pr(\text{Par}^A \rightarrow \text{Par}^B)}{\text{Lik}(\text{data}|\text{Par}^A) \text{Previa}(\text{Par}^A) \Pr(\text{Par}^B \rightarrow \text{Par}^A)}\right)$$

Geman & Geman (1984) propusieron el algoritmo de Gibbs para modelos con muchos parámetros y/o muy correlacionados.

Es un caso particular de M-H.

Supongamos el modelo $Y \sim \text{Normal}(\mu_Y, \sigma_Y^2)$.

Habría que: $\text{Posterior}(\mu_Y, \sigma_Y^2 | \text{data}) \propto \text{Likelihood}(\text{data} | \mu_Y, \sigma_Y^2) * \text{Previa}(\mu_Y, \sigma_Y^2)$

Pero suponiendo que μ_Y y σ_Y^2 fuesen independientes, el problema más complejo se simplifica a:

$\text{Posterior}(\mu_Y, \sigma_Y^2 | \text{data}) \propto \text{Likelihood}(\text{data} | \mu_Y, \sigma_Y^2) * \text{Previa}(\mu_Y) \text{Previa}(\sigma_Y^2)$

We know. We've seen it all before.

el algoritmo de Gibbs

Simulation options

Algorithm **GibbsSampling**

Target distribution **standard**

Autoplay

Autoplay delay

Tweening delay

Step

Reset

Visualization Options

Animate proposal

Show target

Show samples

Show histogram

Histogram bins

Algorithm Options



Algoritmo HMC

Simulation options

Algorithm	HamiltonianMC
Target distribution	standard
Autoplay	<input type="checkbox"/>
Autoplay delay	0
Tweening delay	0
Step	<input type="button" value="Step"/>
Reset	<input type="button" value="Reset"/>

Visualization Options

Animate proposal	<input type="checkbox"/>
Show target	<input checked="" type="checkbox"/>
Show samples	<input checked="" type="checkbox"/>
Show histogram	<input checked="" type="checkbox"/>
Histogram bins	50

Algorithm Options

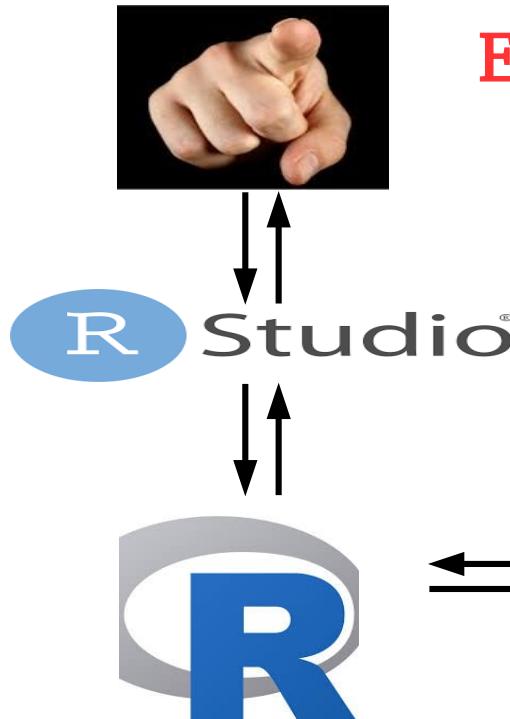
Leapfrog Steps	18
Leapfrog Δt	0.1

About this algorithm

[Close Controls](#)



d. ¿Cómo vamos a trabajar en este curso?

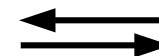


El flujo de interacciones entre el Usuario
y el software se puede resumir así:

Paquete de R que
interactúa con:

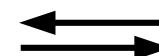
Generador de
MonteCarlo

R2WinBUGS



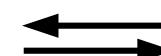
WinBUGS

R2Openbugs



OpenBugs

rjags,runjags,R2jags



JAGS

INLA



INLA

rstan, brms,rstanarm

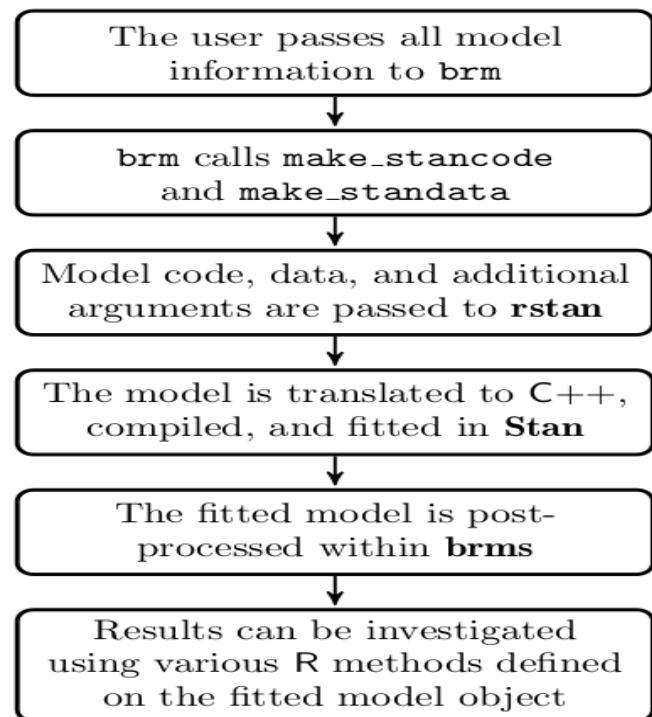


STAN

Y hay otras opciones como NIMBLE, LaplaceDemon, etc...

En el caso de Stan, además de usarlo a través del paquete **rstan**, **rstanarm** y **brms** permiten estimar parámetros de Modelos Lineales Generalizados Mixtos (y otros similares en un sentido muuuy amplio) a través de **rstan**.

La idea es:



JAGS: “Just Another Gibbs Sampler” implementa una variante del MCMC original (Geman & Geman 1984).

Ésta es la base los venerables BUGS,
Winbugs y JAGS.

Recordemos:

$$Post(\beta_0, \beta_1, \sigma | datos) \approx Verosim(datos | \beta_0, \beta_1, \sigma) * Previa(\beta_0, \beta_1, \sigma)$$

Funciona suponiendo una distr posterior (ej. normal multiv.) y reemplazando el cálculo complejo en 3 dimensiones por una secuencia de 3 cálculos más simples de 1 dimensión.

$$Post(\beta_0 | datos, \beta_1, \sigma) \approx Verosim(datos | \beta_0, \beta_1, \sigma) * Previa(\beta_0, \beta_1, \sigma)$$

$$Post(\beta_1 | datos, \beta_0, \sigma) \approx Verosim(datos | \beta_0, \beta_1, \sigma) * Previa(\beta_0, \beta_1, \sigma)$$

Etc.

Particularmente útil para modelos jerárquicos.

Stan: emplea otro algoritmo tipo MCMC (Hamiltonian MCMC) para obtener una distr. posterior multivariada sin caer en mínimos locales (Hooten & Hefley 2019).

El algoritmo genera así “grandes saltos” y una “trayectoria onduladante” entre valores consecutivos posibles de los parámetros.

Integrated Nested Laplace Approximation: usa una serie de Taylor para aproximar una distr. posterior normal multivariada (Wang et al. 2017).

INLA y Stan: mucho más rápidos que JAGS.

Comparación:

glmmTMB

Bürkner (2017)

	brms	lme4	glmmTMB	rstanarm
Supported model types:				
Linear models	yes	yes	yes	yes
Robust linear models	yes	no	no	yes ¹
Binomial models	yes	yes	yes	yes
Categorical models	yes	no	no	no
Multinomial models	no	no	no	no
Count data models	yes	yes	yes	yes
Survival models	yes ¹	yes	no	yes
Ordinal models	various	no	no	cumulative ³
Zero-inflated and hurdle models	yes	no	yes	no
Generalized additive models	yes	no	no	yes
Non-linear models	yes	no	no	no
Additional modeling options:				
Variable link functions	various	various	various	various
Weights	yes	yes	yes	yes
Offset	yes	yes	yes	yes
Multivariate responses	limited	no	no	no
Autocorrelation effects	yes	no	some	no
Category specific effects	yes	no	no	no
Standard errors for meta-analysis	yes	no	no	no
Censored data	yes	no	no	no
Truncated data	yes	no	no	no
Customized covariances	yes	no	some	no
Missing value imputation	no	no	no	no
Bayesian specifics:				
parallelization	yes	—	—	yes
population-level priors	flexible	— ³	—	normal, Student-t
group-level priors	normal	— ³	—	normal
covariance priors	flexible	— ³	—	restricted ⁵
Other:				
Estimator	HMC, NUTS	ML, REML	HMC, NUTS	
Information criterion	WAIC, LOO	AIC, BIC	AIC, LOO	
C++ compiler required	yes	no	no	
Modularized	no	yes	no	

La ventaja es que la sintaxis de los modelos (GLM(M) es casi idéntica a paquetes como lme4 y glmmTMB.

(¿que espero uds. conozcan al menos un poco?)

Los GLM(M) no son todo en la vida....

pero casi

Hamiltonian Monte Carlo

Simulation options	
Algorithm	<input type="button" value="HamiltonianMC"/> <input checked="" type="button"/>
Target distribution	<input type="button" value="standard"/> <input checked="" type="button"/>
Autoplay	<input type="checkbox"/>
Autoplay delay	<input type="text" value="0"/>
Tweening delay	<input type="text" value="0"/>
Step	<input type="button"/>
Reset	<input type="button"/>
Visualization Options	
Animate proposal	<input type="checkbox"/>
Show target	<input checked="" type="checkbox"/>
Show samples	<input checked="" type="checkbox"/>
Show histogram	<input checked="" type="checkbox"/>
Histogram bins	<input type="text" value="50"/>
Algorithm Options	
Leapfrog Steps	<input type="text" value="18"/>
Leapfrog Δt	<input type="text" value="0.1"/>
About this algorithm	
<input type="button" value="Close Controls"/>	



e. ¿Por qué $Pr(\text{parámetros} | \text{datos}) \approx Pr(\text{datos} | \text{parámetros}) * Pr(\text{parámetros})$?

El teorema de Bayes establece que:

$$Pr(\mu_Y | \text{datos}) = \frac{Pr(\text{datos} | \mu_Y) * Pr(\mu_Y)}{Pr(\text{datos})}$$

Pr(datos): prob. de obtener los datos

para todos los valores posibles de μ_Y .

$$Pr(\mu_Y | \text{data}) = \frac{Pr(\text{data} | \mu_Y) * Pr(\mu_Y)}{\int Pr(\text{data} | \mu_Y) * Pr(\mu_Y) d\mu_Y}$$

$\int Pr(\text{data} | \mu_Y) * Pr(\mu_Y) d\mu_Y$: suma de productos de Lik(data | μ_Y) * Pr_{Previa}(μ_Y)

Pr_{Posterior}($\mu_Y | \text{datos}$)

Mágicamente, imaginemos que μ_Y solo

puede tener dos valores posibles: 1.32 y 1.45



$$Pr(\text{datos}) = Pr(\text{datos} | \mu_Y = 1.32) * Pr(\mu_Y = 1.32) + Pr(\text{datos} | \mu_Y = 1.45) * Pr(\mu_Y = 1.45)$$

Pr(datos) es solo una estandarización difícil de calcular!

El cálculo numérico de $\text{Pr}(\text{datos})$ es aún más difícil para modelos con muchos parámetros: $\text{Y} \sim \text{BlaBla}(\theta_1, \theta_2, \theta_3)$.

$$\text{Pr}(\theta_1, \theta_2, \theta_3 | \text{data}) = \frac{\text{Pr}(\text{data} | \theta_1, \theta_2, \theta_3) * \text{Pr}(\theta_1) \text{Pr}(\theta_2) \text{Pr}(\theta_3)}{\int \int \int \text{Pr}(\text{data} | \theta_1, \theta_2, \theta_3) \text{Pr}(\theta_1) \text{Pr}(\theta_2) \text{Pr}(\theta_3) d\theta_1 d\theta_2 d\theta_3}$$



Seleccionando 1000 valores posibles por parámetro, habría $1000^3 = 10^9$ tripletes a evaluar ← “curse of dimensionality”

Además no es necesario:

Al final de cuentas, $\text{Pr}(\text{datos})$ es un número (laboriosamente calculado) que estandariza el numerador, lo único importante para la inferencia!

2. Elementos básicos del análisis bayesiano

- a. Máximo de verosimilitud. Práctico 01
- b. Markov Chain Monte Carlo (MCMC).
- c. Algunas simulaciones ilustrativas.
- d. ¿Cómo vamos a trabajar en este curso?
- e. ¿Por qué $Pr(\text{parámetros} | \text{datos}) \approx Pr(\text{datos} | \text{parámetros}) * Pr(\text{parámetros})$?