Introducción a la estadística Bayesiana con aplicaciones de estimación en áreas pequeñas usando software STAN

Ignacio Alvarez-Castro Juan José Goyeneche

Instituto de Estadística, Facultad de Ciencias Económicas y Administración, UdelaR.

XV Congreso Latinoamericano de Sociedades de Estadística
9 al 13 de Octubre 2023
Santiago de Cali, Colombia

A partir de un modelo ($p(y|\theta)$ y $p(\theta)$) obtuvimos $p(\theta|y)$ (o al menos una función proporcional).

A partir $(\theta|y)$, podemos estar interesados en

- inferencia para θ o transformaciones $\phi(\theta)$
- hacer predicciones de nuevas observaciones
- evaluar el ajuste de nuestro modelo

En muchos casos, los objetivos anteriores se pueden expresar como:

$$\int h(\theta)p(\theta|y)d\theta \tag{1}$$

Monte Carlo: si contamos con valores simulados de la posterior:

$$\{\theta^1, \theta^2, \ldots, \theta^S\} \quad \text{con } \theta^i \sim p(\theta|y)$$

Podemos aproximar la cantidad de interés:

$$\frac{1}{S}\sum_{i=1}^{n} h(\theta^i) \longrightarrow \int h(\theta)p(\theta|y)d\theta = E(h(\theta)|y)$$

Las notas de Muestreo

Notas en la primera revisión de Muestreo, tres ejercicios que sumaban 30.

```
y <- c(21, 22, 2, 25, 3, 19, 17, 23, 6, 7, 8, 14, 22)
n <- 30
```

Como modelo para los datos se propone $y_i|\theta \sim Binomial(n = 30, \theta)$, donde el interés es realizar inferencia sobre θ, la probabilidad de "ganar un punto".

- Con previa $\theta \sim Unif(0, 1)$, calcular $Pr(\theta > 0.5|y)$ usando `pbeta()` y mediante simulación.
- Obtener la posterior usando $\theta \sim Normal(0.5, 0.2^2)I_{(0,1)}$ como previa

Con $y_i|\theta \sim Binomial(n = 30, \theta)$ y $\theta \sim Unif(0, 1)$ sabemos que

Con $y_i|\theta \sim Binomial(n = 30, \theta)$ y $\theta \sim Unif(0, 1)$ sabemos que

$$\theta|y \sim Beta(\sum y_i + 1, \sum(30 - y_i) + 1)$$

Podemos aproximar con simulaciones:

```
theta.i <- rbeta(10e3, sum(y) + 1, length(y)*n-sum(y) + 1)

c(prob = pbeta(0.5,sum(y)+1, length(y)*n-sum(y)+1, lower.tail=FALSE),
  simula = mean(theta.i > .5) )

##      prob    simula
## 0.2719972 0.2723000
```

Queremos:

$$\left\{\theta^1, \ldots, \theta^S\right\} \text{ con } \theta^i \sim p(\theta|y)$$

pero no conocemos $p(\theta|y)$!!!

- elegir un valor inicial θ^0
- hacer simulaciones **dependientes**
- las simulaciones forman una *cadena de markov*

$$\theta^{i+1} \sim f(\theta^{i+1}|\theta^i, y)$$

notemos que $\left\{\theta^1, \ldots, \theta^S\right\}$ NO son independientes y NO provienen de la posterior de interés!

Una cadena de Markov es una secuencia de variables aleatorias dependientes

$$X^1, X^2, \ldots, X^t, \ldots$$

tal que la distribución de X^t dada las variables pasadas solo depende de X^{t-1},

$$p(X^t|X^1, \ldots, X^{t-1}) = p(X^t|X^{t-1})$$

El futuro depende del pasado solo a través del presente

El método para simular debe garantizar que *existe una distribución estacionaria de la cadena*, $p(x)$, de forma que:

$$\text{Si } X^t \sim p() \longrightarrow X^{t+1} \sim p()$$

Si la cadena es *ergódica* entonces se cumple que

$$\frac{1}{S}\sum h(X^t) \longrightarrow \int h(x)p(x)dx$$

Esto implica que podemos trabajar con las simulaciones de la cadena de igual forma que con simulaciones Monte Carlo.

Los métodos que combinan simulaiones de monte carlo con cadenas de Markov son llamados Markov Chain Monte Carlo (MCMC).

- Muchos algoritmos: Gibbs, MH, HMC, SMC, ...
- Varios programas: BUGS, JAGS, STAN, ...

Desde R, podemos usar ..

```
install.packages( rstan )
library(rstan)
```

Ejemplo inicial de STAN

Dos básicos pasos:

1. definir la estructura del modelo (archivo `mimodelo.stan`)
2. obtener simulaciones de la posterior conjunta (función `stan()`)

Modelo:

$$y_i \overset{iid}{\sim} Binomial(n = 30, \theta) \quad \theta \sim Normal(0.5, 0.2^2) I_{(0,1)}$$

Escribir el modelo: `inicial.stan`

```
// bloque de datos y/o parametros
data {
  int<lower=0> N;
  int y[N];
}
// bloque para definir parametros y su espacio
parameters{
  real<lower=0, upper=1> theta;
}
// Bloque del modelo: previas y modelo para datos
model {
  theta ~ normal(.5, .2);
  y ~ binomial(30, theta);
}
```

Obtener muestras de posterior

```r
# cargamos la libreria
library(rstan)

# ponemos los datos en una lista con nombres
dt.ls <- list( N=length(y), y = y)


# Obtenemos simulaciones de la posterior
rstan_options(auto_write = TRUE)
res = stan(file = 'muestreo.stan', data = dt.ls )
```

Muestras de posterior

```
rstan::extract(res, pars='theta', permuted=FALSE) |> head()
```

```
## , , parameters = theta
##
##             chains
## iterations   chain:1   chain:2   chain:3   chain:4
##       [1,] 0.4741576 0.4733367 0.4581334 0.4888366
##       [2,] 0.4662895 0.4881370 0.4895546 0.4486647
##       [3,] 0.4850276 0.5174243 0.5004467 0.4964224
##       [4,] 0.4826960 0.5133198 0.4945396 0.4782199
##       [5,] 0.4990548 0.4531677 0.4945396 0.4814070
##       [6,] 0.4968903 0.4489703 0.4945396 0.5186371
```

Resultados del modelo

¿ Que cosas no conocemos de esta salida ?

```
res


## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rha
## theta    0.49    0.00 0.03    0.44    0.47    0.49    0.50    0.53  1640
## lp__  -272.04    0.01 0.68 -273.92 -272.22 -271.77 -271.59 -271.53  2283
##
## Samples were drawn using NUTS(diag_e) at Sun Oct  8 22:23:30 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Resultados del modelo

¿ Que cosas no conocemos de esta salida ?

```
res


## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rha
## theta    0.49    0.00 0.03    0.44    0.47    0.49    0.50    0.53  1640
## lp__  -272.04    0.01 0.68 -273.92 -272.22 -271.77 -271.59 -271.53  2283
##
## Samples were drawn using NUTS(diag_e) at Sun Oct  8 22:23:30 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

En la practica, utilizamos las simulaciones prvenientes de MCMC *como si fueran* Monte Carlo independientes. Que tan mala es esta aproximación?

Calidad de valores simulados

Recordemos que las simulaciones $\{\theta^1, \ldots, \theta^S\}$

- NO son independientes
- NO provienen de $p(\theta|y)$

PERO ... si la Cadena de Markov es ergódica y simulamos valores suficientes, los valores simulados se aproximan a la distribución estacionaria.

Evaluar la calidad de las simulaciones de MCMC

- Gráficar las simulaciones (trace plot)
- $\hat{R}$: cuanta precisión ganamos si seguimos simulando
- n_{eff}: equivalente en simulaciones independientes

Efecto de valor inicial

Sabemos que θ^0 no proviene de la distribución estacionaria
Como afecta los siguientes valores simulados ?

Ejemplo de juguete:

- Distribución estacionaria $N(0, 1)$
- Simulamos 1000 valores de 2 cadenas de Markov

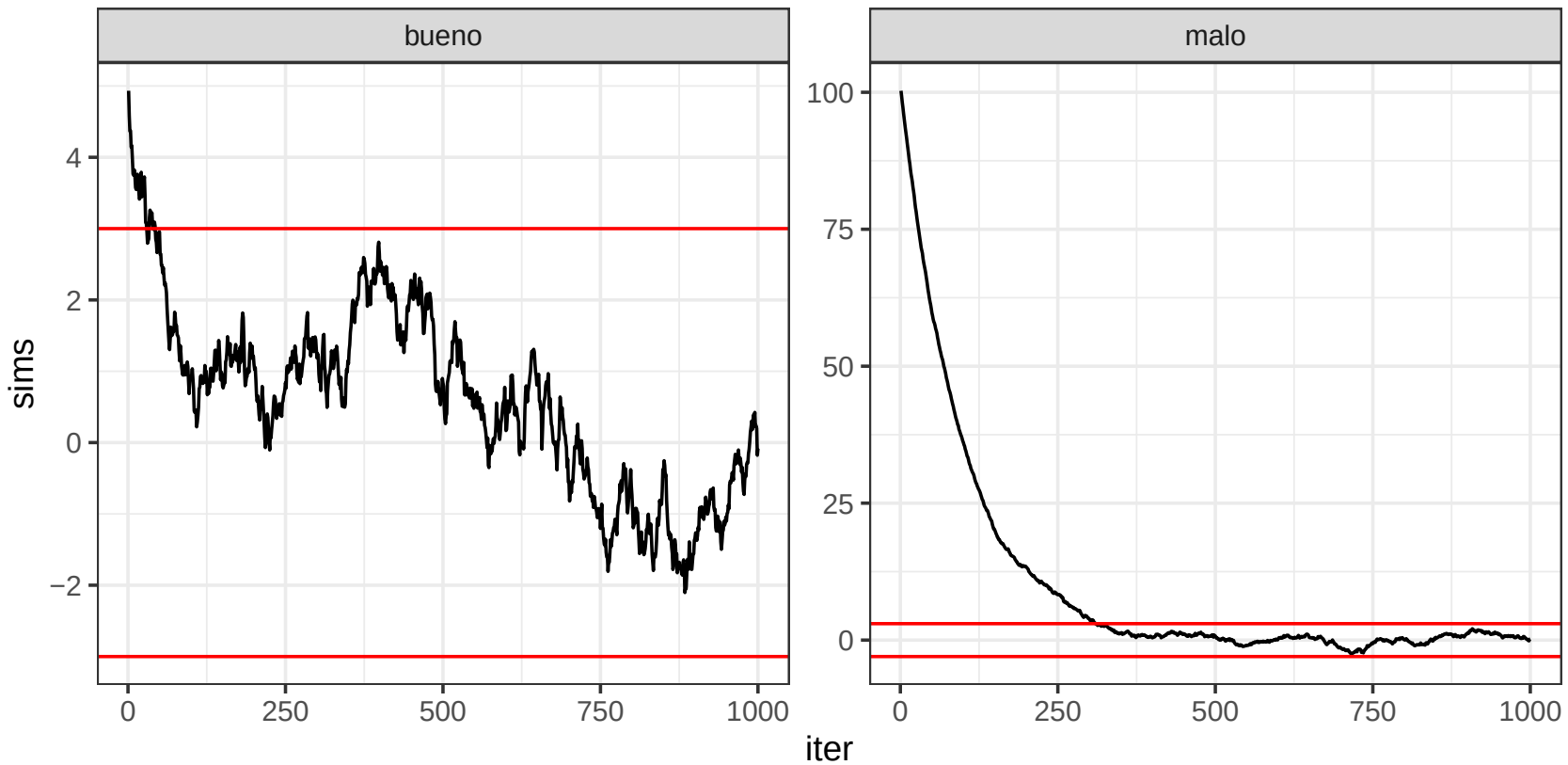
Valores iniciales:

$$\text{bueno} \quad \theta^0 \sim N(4, 1)$$

$$\text{malo} \quad \theta^0 \sim N(100, 1)$$

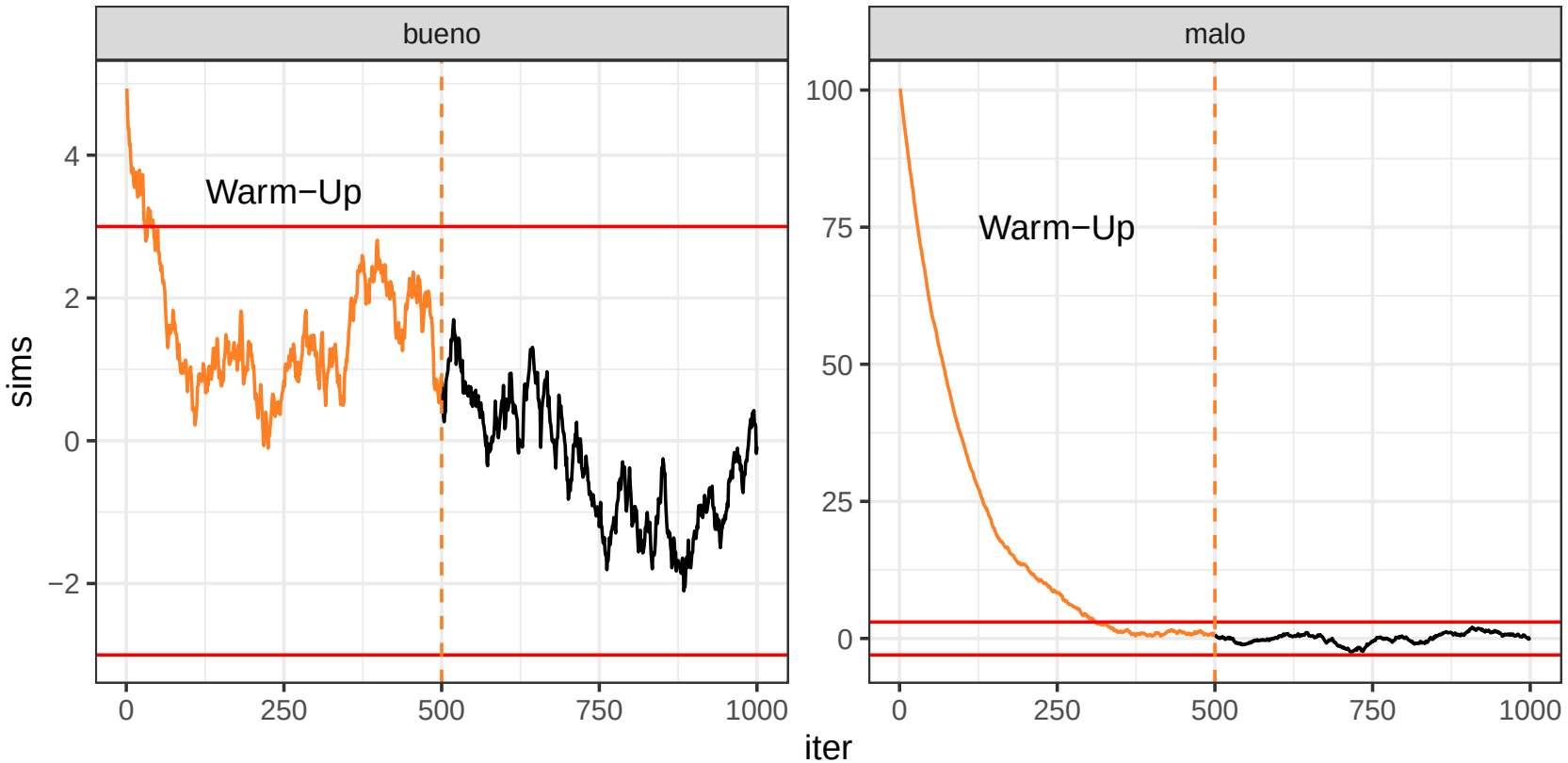
Efecto de valor inicial

Sabemos que θ^0 no proviene de la distribución estacionaria
Como afecta los siguientes valores simulados ?



Efecto de valor inicial

En la práctica *no sabemos* en que situación nos encontramos



Distribución estacionaria

Sabemos que las simulaciones convergen a su distribución estacionaria
Como se ven simulaciones que SI provienen de la distribución estacionaria de la cadena ?

Distribución estacionaria

Sabemos que las simulaciones convergen a su distribución estacionaria
Como se ven simulaciones que SI provienen de la distribución estacionaria de la cadena ?

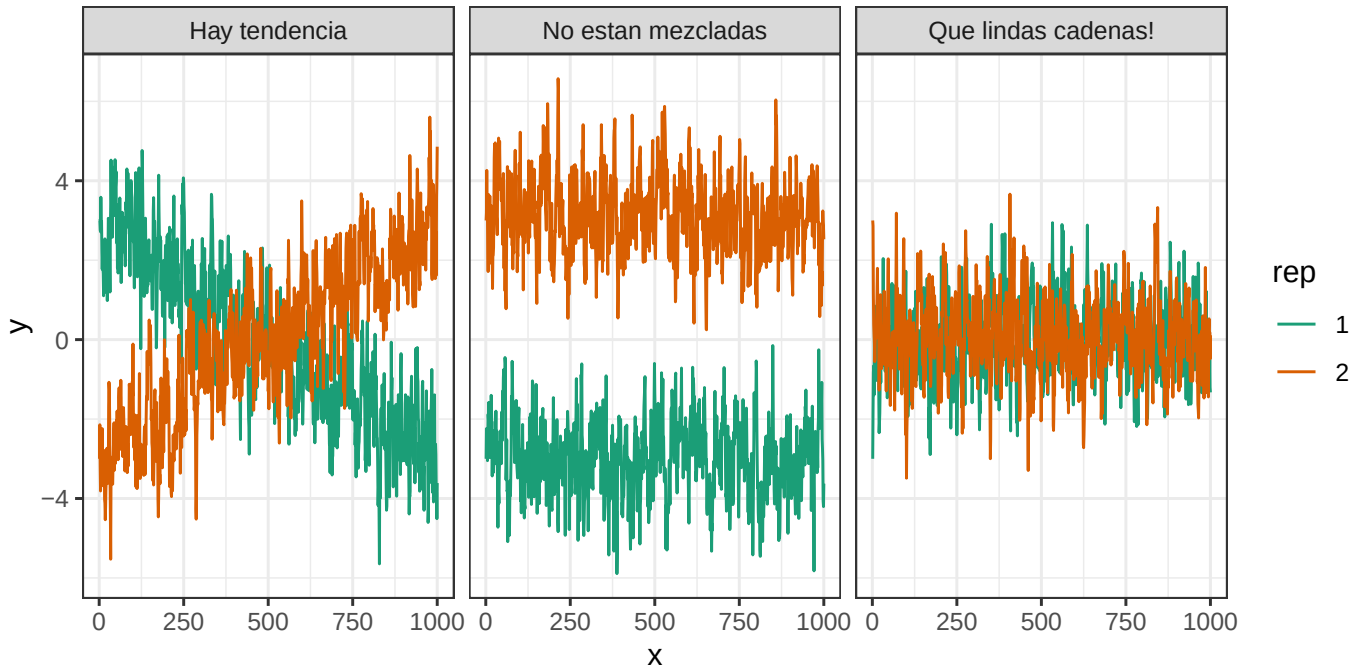
NO sabemos!

Pero si tenemos indicaciones que debemos seguir simulando.

Ejemplo, $p(x) \sim N(0, 1)$, hacemos **2 cadenas** independientes

- Problema: las cadenas no son estables
- Problema: las cadenas no se mezclan

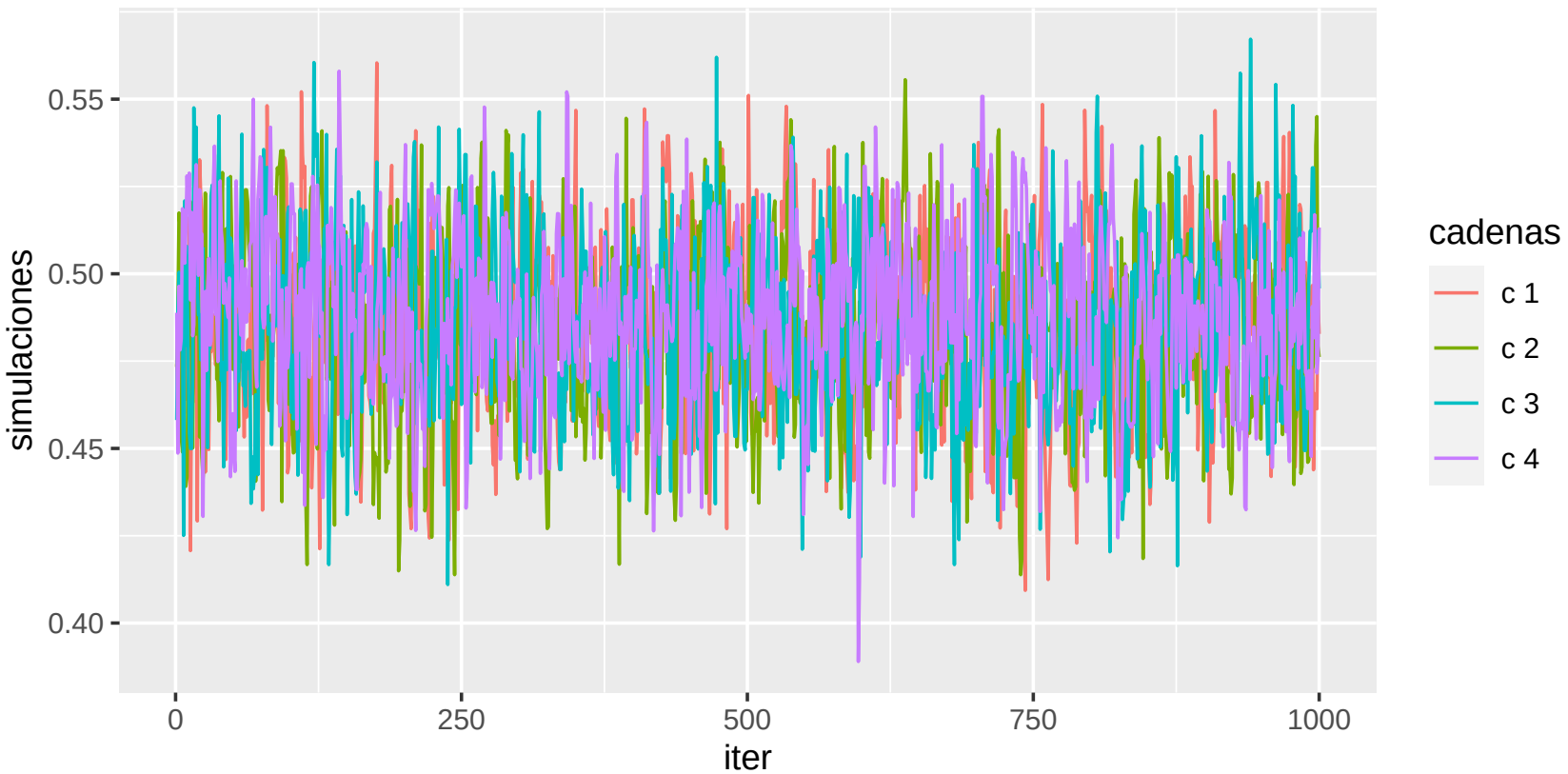
Distribución estacionaria



Si las cadenas convergen, entonces las simulaciones no deben presentar una tendencia y deben provenir de la misma distribución.

Dibujo de traza - Notas de muestreo

```
rstan::extract(res, pars='theta', permuted=FALSE) |> data.frame( ) |>
  setNames(nm=paste('c', 1:4)) |> mutate(iter = 1:1000) |>
  pivot_longer(cols=1:4, names_to = 'cadenas', values_to='simulaciones'
  ggplot() +  geom_line(aes(iter, simulaciones, color=cadenas))
```



Factor $\hat{R}$

Todas las cadenas tienen la misma distribución limite, la variabilidad *entre* cadenas debe ser chica. Cota superior de la varianza posterior:

$$var^{+}(\theta|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

- n número de iteraciones en cada cadena
- descomponemos la variabilidad en W (dentro) y B (entre)

Cuando $n \to \infty$ se cumple $var^{+}(\theta|y) \to W$

$$\hat{R} = \sqrt{\frac{var^{+}(\theta|y)}{W}}$$

representa cuanto se puede reducir la varianza si aumentamos las iteraciones.

Número de muestras efectivo

El objetivo de un algoritmo de MCMC es obtener un conjunto de simulaciones de la posterior y *tratarlas como simulaciones Monte Carlo (MC)*

PERO, las simulaciones con MCMC son **dependientes**. Si para estimar $E(h(\theta|y))$ usamos

$$\hat{h}_S = \frac{1}{S}\sum h(\theta^i)$$

entonces, *pagamos* en la varianza:

$$var(\hat{h}_S) \to var(\theta|y)\left(1 + 2\sum_{k=1}^{\infty}\rho_k\right)$$

Número de muestras efectivo

Definimos

$$n_{eff} = \frac{nm}{\left(1 + 2\sum_{k=1}^{\infty} \rho_k\right)}$$

como la cantidad de muestras equivalentes si fueran independientes.

- $S = mn$, m cadenas con n iteraciones cada una
- ρ_k representan autocorrelaciones
- Hay que estimar $\sum \rho_k$, lo que da lugar a $\hat{n}_{eff}$

Parámetros con n_{eff} chico indican que no estan siendo bien estimados.

Recomendaciones generales (en BDA)

BDA recomienda para monitorear convergencia:

1. Simular $m/2$ cadenas, con $4n$ iteraciones cada una. Fijando valores iniciales dispersos en relación a $p(\theta|y)$
2. Descartar la primer mitad de iteraciones en cada cadena (warm up)
3. Dividir cada cadena a la mitad: quedan m cadenas de largo n
4. Calcular $\hat{R}$ para cada parámetro escalar hasta que $\hat{R} < 1.1$
5. Lograr $\hat{n}_{eff} > 5m$

ALERTA !!!

Las medidas $\hat{R} < 1.1$ y $\hat{n}_{eff} > 5m$ indican que no hay indicios de falta de convergencia

NUNCA podemos estar seguros que la cadena ya convergió

Resultados del modelo

¿Qué cosas conocemos de esta salida ?

```
res


## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rha
## theta    0.49    0.00 0.03    0.44    0.47    0.49    0.50    0.53  1640
## lp__  -272.04    0.01 0.68 -273.92 -272.22 -271.77 -271.59 -271.53  2283
##
## Samples were drawn using NUTS(diag_e) at Sun Oct  8 22:23:30 2023.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```