Ejemplos

Lizbeth Naranjo Albarrán y Luz Judith Rodriguez Esparza

Paper: *Modelos ocultos de Markov: una aplicación en series de tiempo*

Authors: Lizbeth Naranjo Albarrán & Luz Judith Rodríguez Esparza

Journal: Mixba'al

Year: 2023

https://github.com/lizbethna/HMMaplica

Este archivo muestra las instrucciones para correr los códigos de R y Stan.

Cadenas de Markov

```
library(ggplot2)
library(extraDistr)
library(rstan)
```

Calcular probabilidades

```
### Datos
N = 100   #tamaño de muestra
K = 4   # estados
A = matrix(0,4,4)   # matriz de probabilidades de transicion
A[1,] = c(0.3, 0.3, 0, 0.4)   # simplex: acelerar
A[2,] = c(0.2, 0.4, 0, 0.4)   # simplex: constante
A[3,] = c(0.7, 0, 0.3, 0)   # simplex: reposo
A[4,] = c(0.4, 0.1, 0.4, 0.1)   # simplex: freno
rowSums(A)   # renglones suman 1
```

```
[1] 1 1 1 1
```

```
di1 = c(0,0,0,1) # probabilidades del estado oculto inicial
```

```
# Funcion para calcular la distribucion estacionaria delta1
distr_estac = function(A){
 n = nrow(A)
 B = A - diag(n) # Substract the identity to the input matrix
 B[,1] = rep(1,n) # Replace a column of ones
 b = c(1,rep(0,n-1)) # Create the output vector (1,0,0,...,0)
```

```r
 di1 = solve(t(B),b) # Solve the system for di1
 return(di1)
}

# distribucion estacionaria
(estac = distr_estac(A))
```

```
[1] 0.3634476 0.2253375 0.1495327 0.2616822
```

```r
# tiempo medio de recurrencia
(tiempo = 1/estac)
```

```
[1] 2.751429 4.437788 6.687500 3.821429
```

```r
# probabilidad de observaciones
prob_obs <- function(x1,A,di1){
  n = length(x1)
  px1 = rep(NA,n)
  px1[1] = di1[x1[1]]
  for(i in 2:n){
    px1[i] = A[x1[i-1],x1[i]]
  }
  prod(px1)
}
x1 = c(4,4,4,1,1,4,2,4)
prob_obs(x1,A,di1)
```

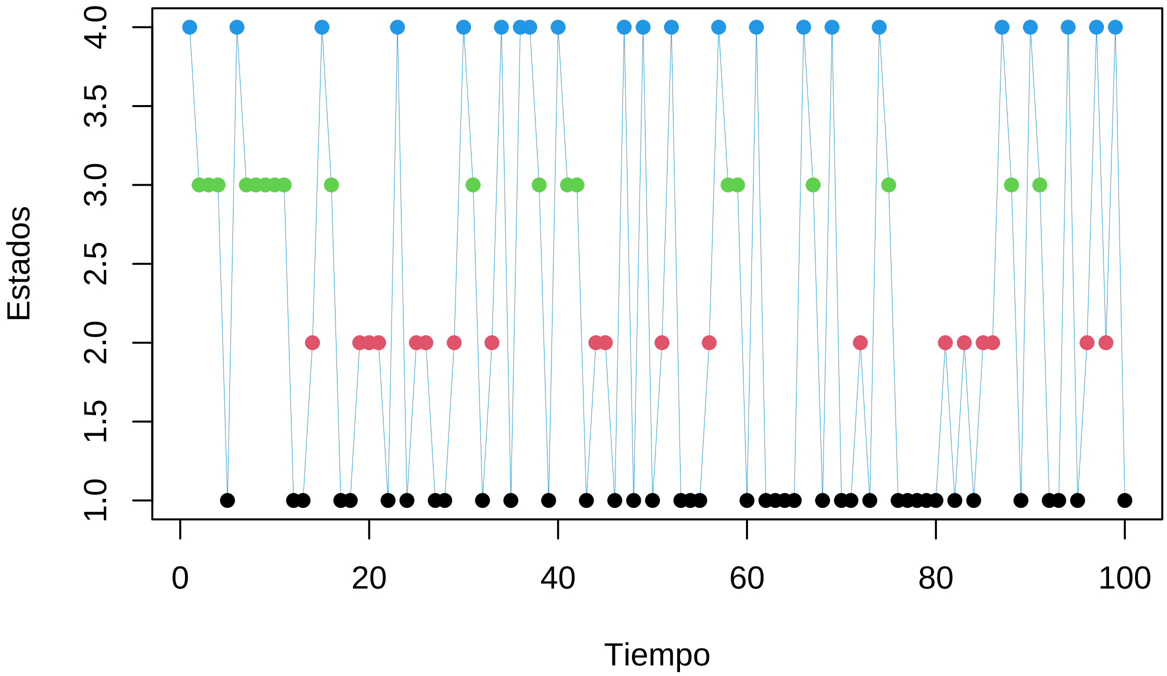
```
[1] 1.92e-05
```

Simular datos

```r
N = 100   # tamaño de muestra

# Generar muestra de una cadena de Markov
# T = tamaño de la cadena de Markov
# A = matriz de transicion
CM_genera <- function(N,A,di1) {
  K = ncol(A) #= nrow(A)
  z <- vector("numeric", N)
  z[1] <- sample(1:K, size = 1, prob = di1)
  for (t in 2:N)
    z[t] <- sample(1:K, size = 1, prob = A[z[t - 1], ])
  list(z = z,
       theta = list(di1 = di1, A = A))
}


cadena = CM_genera(N,A,di1)
plot(cadena$z, type="o",col=cadena$z,lwd=0.1,pch=19,
     xlab="Tiempo", ylab="Estados")
```



Código Stan

Dada una muestra observada, se busca estimar las probabilidades de transición.

```r
datos <- list( "z"=cadena$z, "N"=N, "K"=K,  # muestra
              "alpha"=rep(2,K))  # valores iniciales de la distribucion inicial
param  = c("theta") # parametros a estimar

fit_cm <- stan("cadenas_markov.stan", data=datos,
            chains=2, warmup=1000, iter=2000, thin=2)
```

```
SAMPLING FOR MODEL 'cadenas_markov' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.51 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
```

```
Chain 1:  Elapsed Time: 0.160562 seconds (Warm-up)
Chain 1:                0.145939 seconds (Sampling)
Chain 1:                0.306501 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'cadenas_markov' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 3.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.31 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.200104 seconds (Warm-up)
Chain 2:                0.143233 seconds (Sampling)
Chain 2:                0.343337 seconds (Total)
Chain 2:
```

Resultados

```
print(fit_cm, pars=param)
```

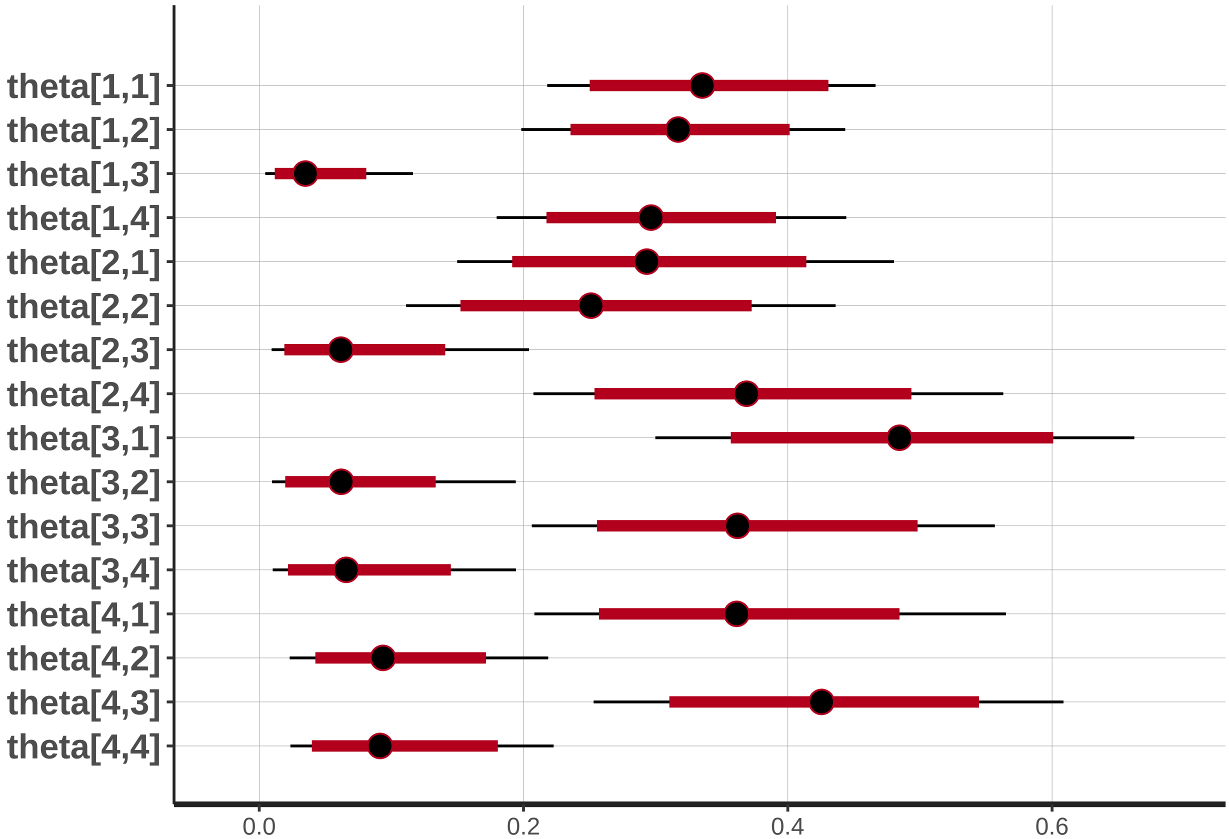
```
Inference for Stan model: cadenas_markov.
2 chains, each with iter=2000; warmup=1000; thin=2;
post-warmup draws per chain=500, total post-warmup draws=1000.

           mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
theta[1,1] 0.34       0 0.07 0.22 0.29 0.34 0.39  0.47   792 1.00
theta[1,2] 0.32       0 0.06 0.20 0.28 0.32 0.36  0.44   814 1.00
theta[1,3] 0.04       0 0.03 0.00 0.02 0.03 0.06  0.12   810 1.00
theta[1,4] 0.30       0 0.07 0.18 0.25 0.30 0.34  0.44   750 1.00
theta[2,1] 0.30       0 0.09 0.15 0.23 0.29 0.35  0.48   772 1.00
theta[2,2] 0.26       0 0.08 0.11 0.20 0.25 0.32  0.44   742 1.01
theta[2,3] 0.07       0 0.05 0.01 0.04 0.06 0.10  0.20   756 1.00
theta[2,4] 0.37       0 0.09 0.21 0.30 0.37 0.44  0.56   783 1.00
theta[3,1] 0.48       0 0.09 0.30 0.42 0.48 0.55  0.66   724 1.00
theta[3,2] 0.07       0 0.05 0.01 0.04 0.06 0.10  0.19   820 1.00
theta[3,3] 0.37       0 0.09 0.21 0.30 0.36 0.43  0.56   685 1.00
theta[3,4] 0.08       0 0.05 0.01 0.04 0.07 0.10  0.20   862 1.00
theta[4,1] 0.37       0 0.09 0.21 0.31 0.36 0.43  0.57   867 1.00
```

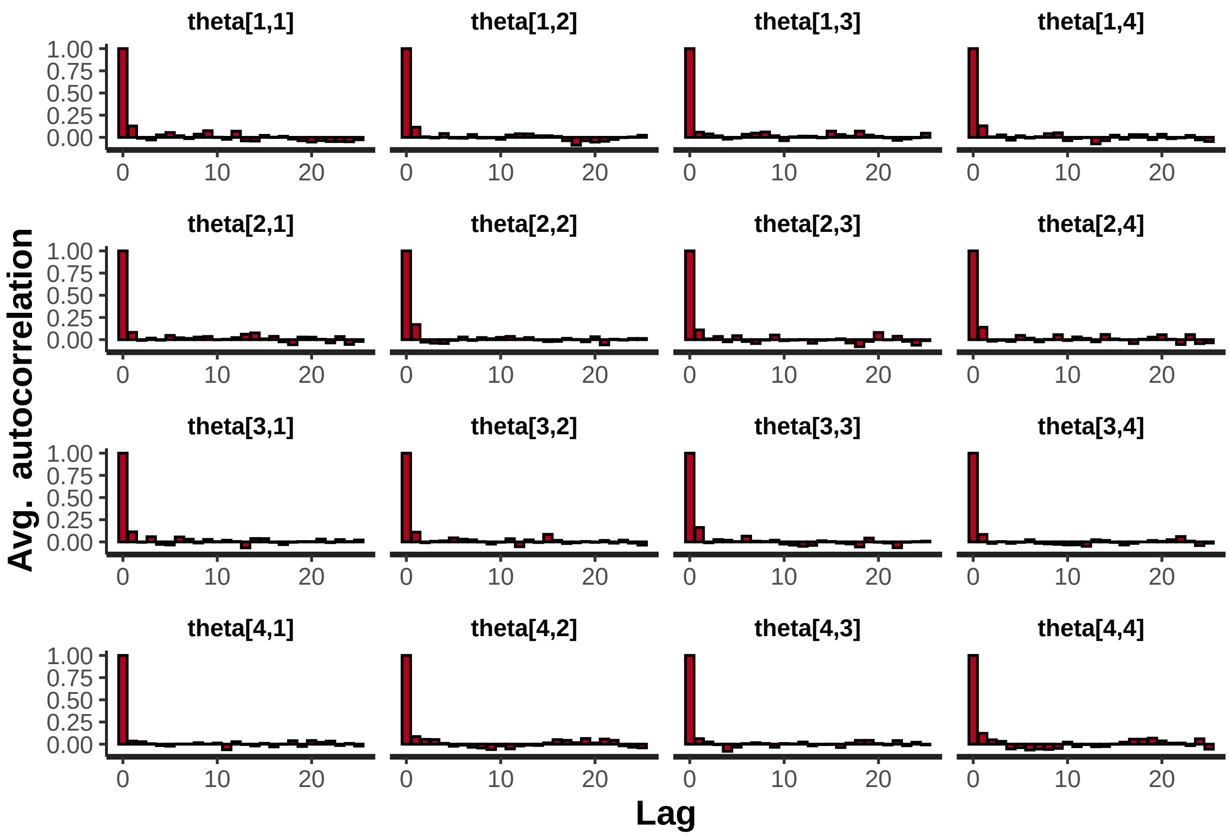
```
theta[4,2] 0.10       0 0.05 0.02 0.06 0.09 0.13  0.22   717 1.00
theta[4,3] 0.43       0 0.09 0.25 0.36 0.43 0.49  0.61   861 1.00
theta[4,4] 0.10       0 0.05 0.02 0.06 0.09 0.13  0.22   712 1.00

Samples were drawn using NUTS(diag_e) at Sat Mar 11 07:57:00 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).
```

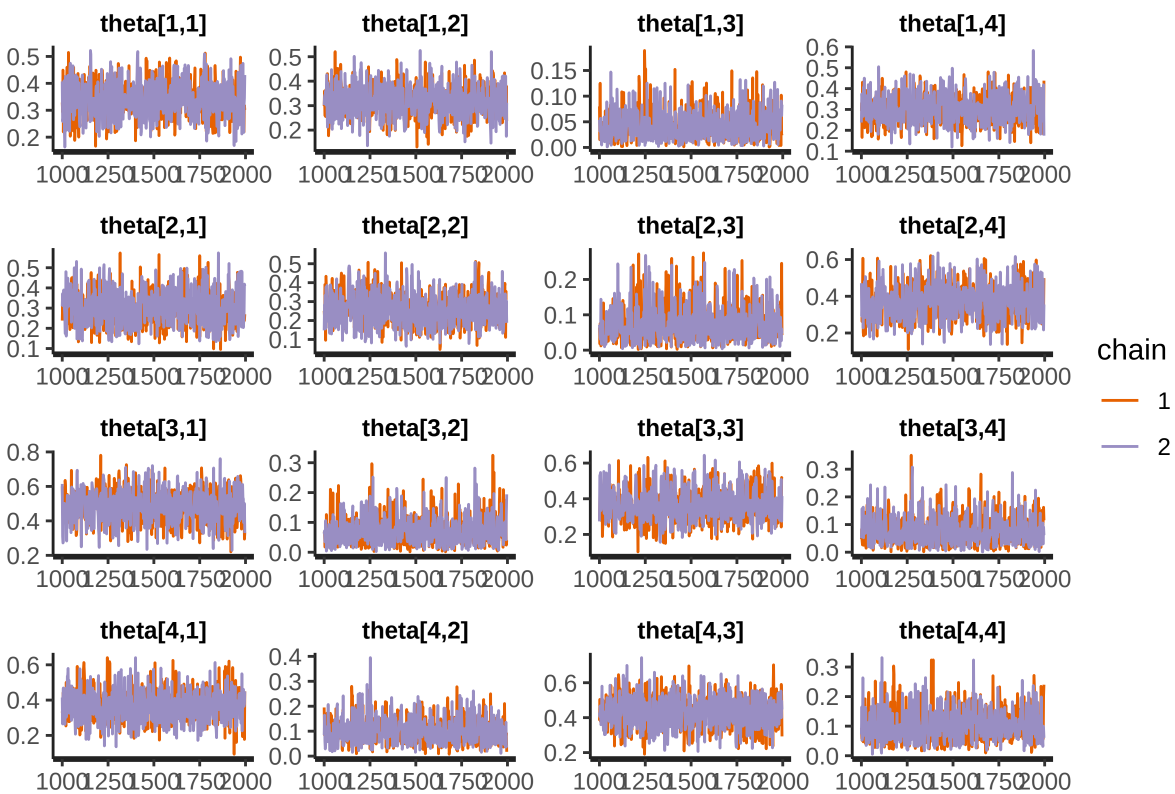
```
stan_plot(fit_cm,pars=param)
```




```
stan_ac(fit_cm,pars=param)
```




```
stan_trace(fit_cm,pars=param)
```



```r
stan_dens(fit_cm,pars="theta", point_est = "mean", show_density = TRUE) +
  ggtitle(paste("Distribución final de theta")) +
  theme(axis.title.x=element_text(size=14), axis.title.y=element_text(size=14),
        plot.title = element_text(size=16))
```