Ejemplos

Lizbeth Naranjo Albarrán y Luz Judith Rodriguez Esparza

Paper: *Modelos ocultos de Markov:*

una aplicación de estimación Bayesiana para series de tiempo financieras

Authors: Lizbeth Naranjo Albarrán & Luz Judith Rodríguez Esparza

Journal: Mixba'al

Year: 2023

https://github.com/lizbethna/HMMBayes.git

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
```

```r
 b = c(1,rep(0,n-1)) # Create the output vector (1,0,0,...,0)
 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)
```

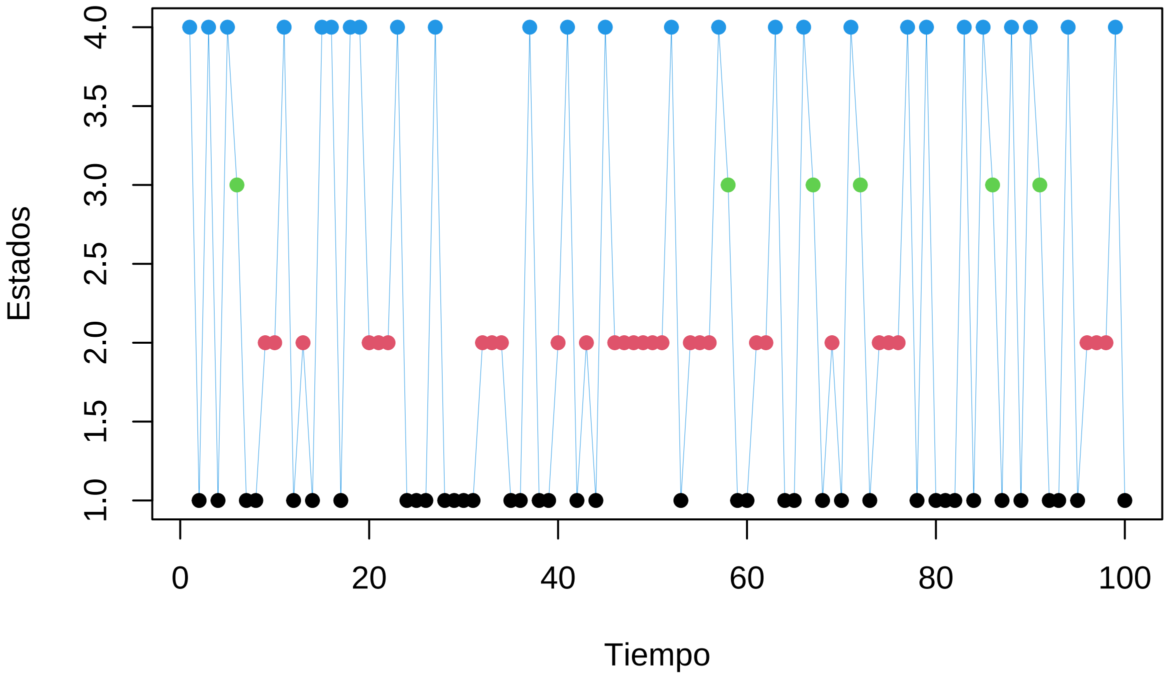
```
[1] 2.751429 4.437788 6.687500 3.821429
```

```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.

```
datos <- list( "z"=cadena$z, "N"=N, "K"=K,  # muestra
              "alpha"=rep(1,K))  # valores iniciales de la distribucion inicial
param  = c("gama") # parametros a estimar

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

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/I
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/I
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Mac
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Mac
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/I
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal
#include <complex>
         ^~~~~~~~~
3 errors generated.
```

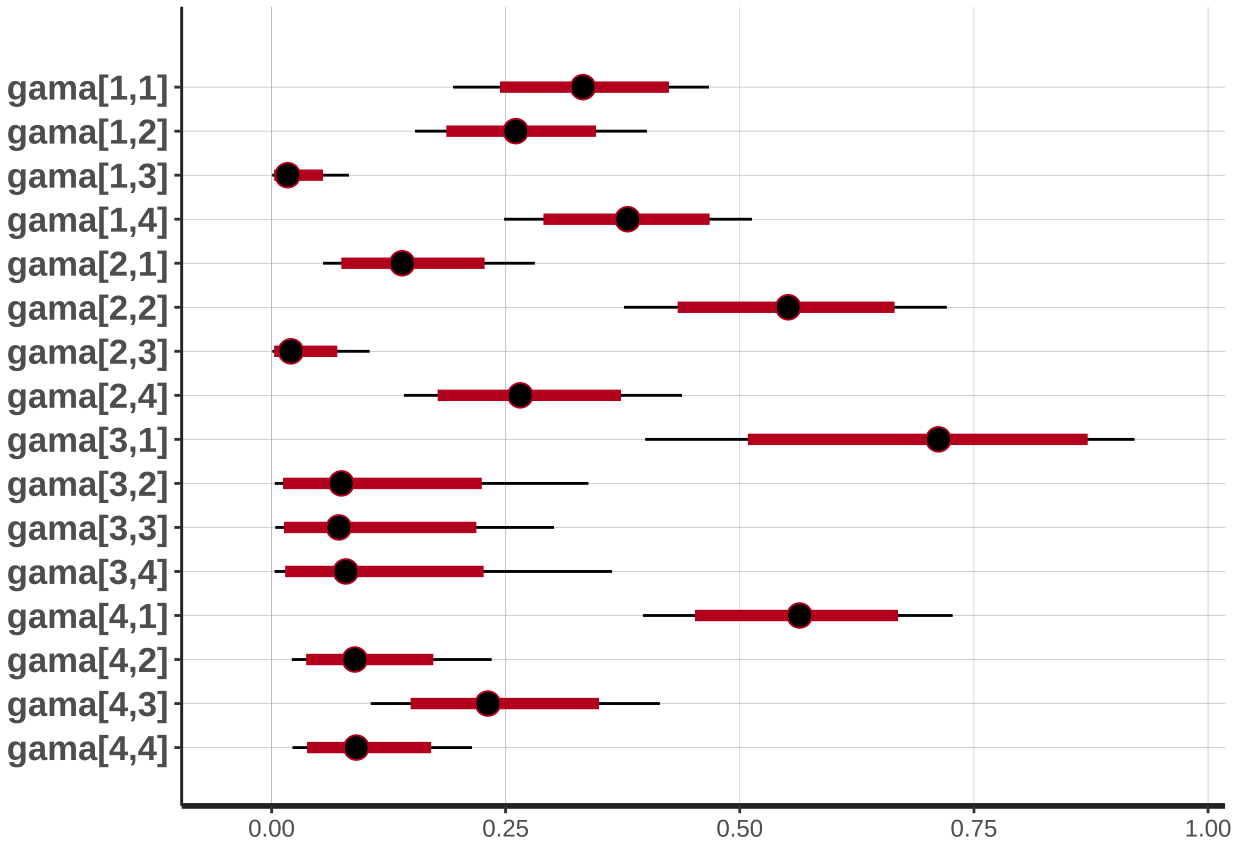
```
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'cadenas_markov' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.46 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: 0.182916 seconds (Warm-up)
Chain 1:                0.151923 seconds (Sampling)
Chain 1:                0.334839 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'cadenas_markov' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 2.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.173797 seconds (Warm-up)
Chain 2:                0.15055 seconds (Sampling)
Chain 2:                0.324347 seconds (Total)
Chain 2:
```

Resultados

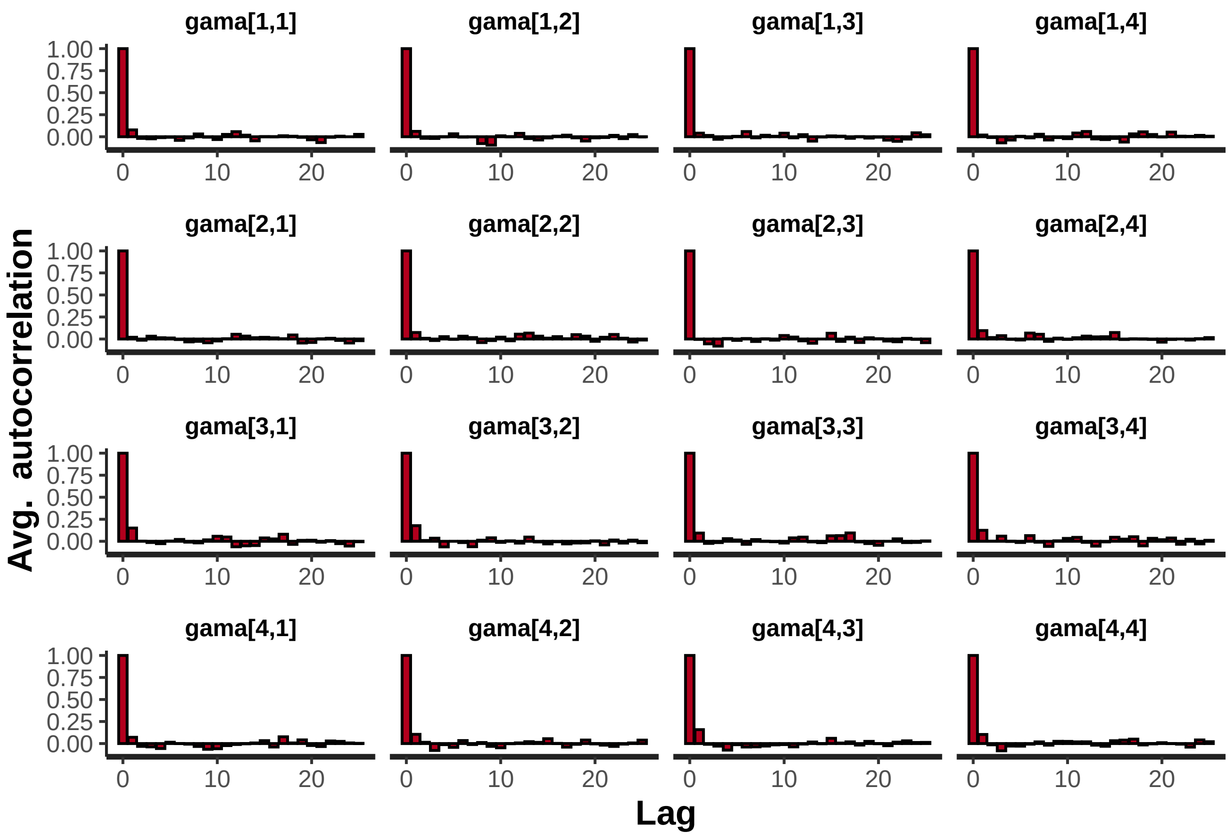
```
print(fit_cm, pars=param)
```

```
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
gama[1,1] 0.33    0.00 0.07 0.19 0.29 0.33 0.38  0.47   854    1
gama[1,2] 0.26    0.00 0.06 0.15 0.22 0.26 0.31  0.40   886    1
gama[1,3] 0.02    0.00 0.02 0.00 0.01 0.02 0.03  0.08   937    1
gama[1,4] 0.38    0.00 0.07 0.25 0.33 0.38 0.42  0.51   958    1
gama[2,1] 0.15    0.00 0.06 0.05 0.10 0.14 0.18  0.28   865    1
gama[2,2] 0.55    0.00 0.09 0.38 0.49 0.55 0.61  0.72   869    1
gama[2,3] 0.03    0.00 0.03 0.00 0.01 0.02 0.04  0.10   989    1
gama[2,4] 0.27    0.00 0.08 0.14 0.22 0.27 0.32  0.44   773    1
gama[3,1] 0.70    0.01 0.14 0.40 0.61 0.71 0.80  0.92   766    1
gama[3,2] 0.10    0.00 0.09 0.00 0.03 0.07 0.14  0.34   701    1
gama[3,3] 0.10    0.00 0.09 0.00 0.03 0.07 0.14  0.30   835    1
gama[3,4] 0.10    0.00 0.09 0.00 0.04 0.08 0.15  0.36   736    1
gama[4,1] 0.56    0.00 0.08 0.40 0.50 0.56 0.62  0.73   851    1
gama[4,2] 0.10    0.00 0.06 0.02 0.06 0.09 0.13  0.23   816    1
gama[4,3] 0.24    0.00 0.08 0.11 0.18 0.23 0.28  0.41   756    1
gama[4,4] 0.10    0.00 0.05 0.02 0.06 0.09 0.13  0.21   829    1

Samples were drawn using NUTS(diag_e) at Tue Mar 14 18:46:25 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).
```

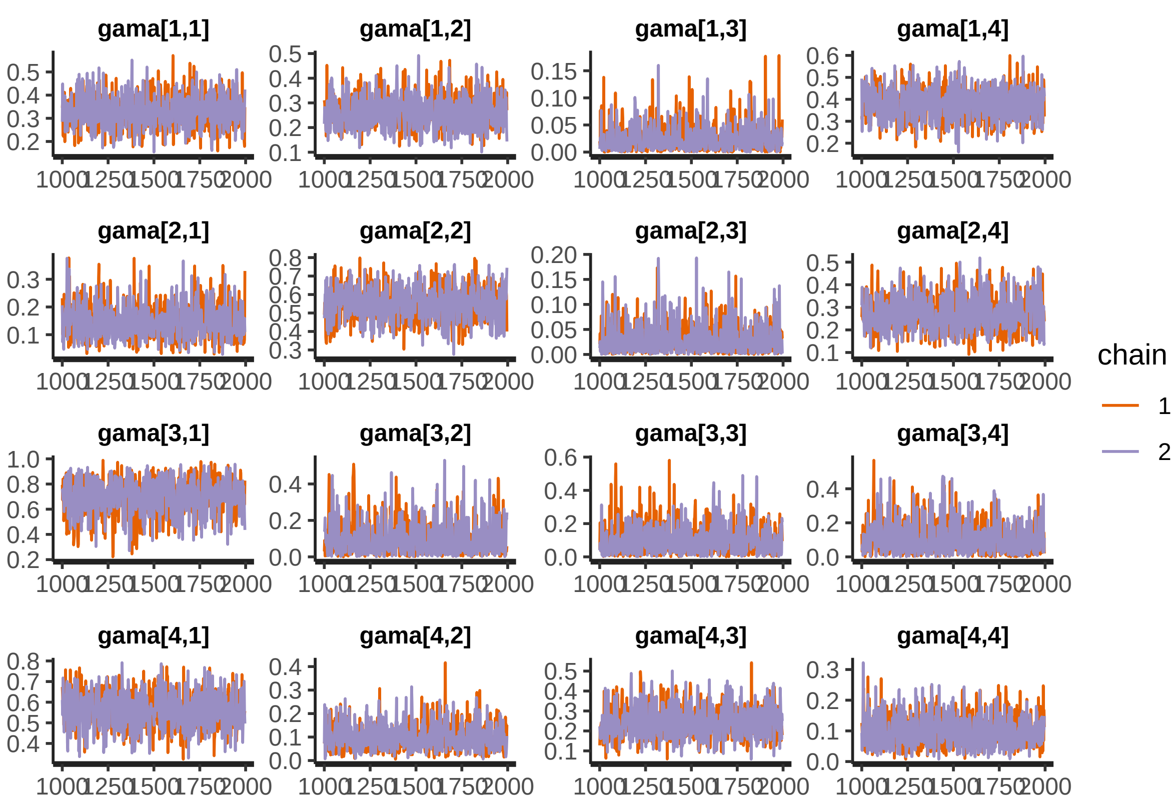
```
stan_plot(fit_cm,pars=param)
```




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




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



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