Una primera aproximación al **Muestreador de Gibbs**

Teoría de Procesos Estocásticos. 2023-I

Michel Mendivenson Barragán Zabala

21 de Junio de 2023

Introducción del problema:

Se tiene que en un juego hay dos monedas, una trucada y una moneda justa. Dos amigos acostumbran a apostar de la siguiente forma:

1. Cada jugador hace dos tiros por turno.
2. El jugador elige si su primer turno es con la moneda justa o con la trucada. Si en su primer tiro obtiene cara el segundo tiro lo hará con la moneda justa, si en cambio obtiene sello debe hacer su segunda lanzada con la moneda trucada.
3. En el segundo tiro también se decide lo siguiente: independientemente de la moneda que se lance si se obtiene sello se tirará en el primer turno de la otra ronda con la moneda justa, pero si sale cara se lanzará el siguiente turno con la moneda trucada.

Las reglas siempre obligan a que si el primer jugador elige la moneda trucada para su primer lanzamiento, el otro debe iniciar con la moneda justa (O viceversa).

Generalmente los amigos apuestan con eventos como: El primero que obtenga en la misma ronda dos caras o dos sellos. POr ello después de tanto jugar deciden determinar si el juego está siendo justo en los eventos con los que usualmente apuestan.

Por ello se preguntan:

1. Si se apuesta sobre obtener dos caras seguidas en el primer lanzamiento de cada ronda ¿Existe alguna diferencia entre escoger lanzar de primero la moneda trucada o la moneda justa?
2. ¿Cuáles son las probabilidades de obtener la combinación $Cara, Cara$ en una ronda?

Finalmente y ya adentrados en el tema, los amigos se preguntan sobre cuál es la probabilidad de que independientemente del segundo lanzamiento de una ronda el primero salga cara o sello y lo mismo con el segundo lanzamiento (Distribuciones marginales)

Implementación del código

Note que por la definición del problema si establecemos a $1 = Sello$ y $0 = Cara$ tendremos que para cada ronda el resultado de la tirada 1 y la tirada 2 son variables aleatorias de tipo Bernoulli con parámetro desconocido. Además de esto también por la información dada en el problema tendremos definidas las siguientes probabilidades condicionales para $X =$ "Resultado de la primera tirada" y para $Y =$ "Resultado de la segunda tirada"

Lanzar la moneda justa en el segundo lanzamiento de la ronda

- $P(Y_n = 0 \,|X_n = 0) = 1/2$
- $P(Y_n = 1 \,|X_n = 0) = 1/2$

Lanzar la moneda trucada en el segundo lanzamiento de la ronda

- $P(Y_n = 0\,|X_n = 1) = 3/4$
- $P(Y_n = 1\,|X_n = 1) = 1/4$

Lanzar la moneda justa en el primer lanzamiento de la ronda

- $P(X_n = 0\,|Y_n = 1) = 1/2$
- $P(X_n = 1\,|Y_n = 1) = 1/2$

Lanzar la moneda trucada en el primer lanzamiento de la ronda

- $P(X_n = 0\,|Y_n = 0) = 3/4$
- $P(X_n = 1\,|Y_n = 0) = 1/4$

Como se tienen todas las condicionales el muestreador de Gibbs es una opción viable para obtener muestras de la distribución conjunta de las dos variables sin tener que conocerla pues aunque si bien es cierto que este ejercicio puede abordarse de forma analítica también es cierto que el cálculo de esto es un poco complejo. Así pues, a continuación se presenta el algoritmo de Gibbs para el caso de dos variables aleatorias (Puede extenderse fácilmente a n variables):

```
1. Verifique que tiene las probabilidades condicionales de todas las
variables respecto a todas las demás.

2. Si las tiene escoja un orden para sus variables aleatorias. En
este caso el orden del vector (X,Y).

3. Seleccione un número al azar "k" válido como valor de X para iniciar con la
generación de la muestra.

4. Con el valor de k genere un número aleatorio "y" para la variable Y
de acuerdo a las probabilidades condicionales definidas anteriormente.

5. Almacene en la última fila de una matriz de 2 columnas el vector (k,y).

6. Genere un valor "k" para la variable aleatoria X con el valor obtenido "y"
y de acuerdo a las probabilidades condicionales anteriormente definidas.

7. Repita los pasos del 4 al 6 hasta generar el número de muestras requerido.

8. Al terminar sus muestras quedarán almacenadas
en la matriz definida antes.
```

Nota: Si quisiera extenderse el algoritmo a n variables aleatorias tendríamos que generar inicialmente $n-1$ números para n-1 *variables* y poder empezar el algoritmo muestreando la variable a la que no le generamos ningún número. Además de que deberíamos conocer **TODAS** las probabilidades condicionales. Esto es: si queremos muestrear la probabilidad conjunta de X, Y, Z tendríamos que conocer las probabilidades condicionales $P(X_n\,|Y_n = a, Z_n = b)$, $P(Y_n\,|X_n = a, Z_n = b)$ y $P(Z_n\,|Y_n = a, X_n = b)$.

Ahora bien, el muestreador de Gibbs genera una secuencia que proviene de una cadena de Markov y lo que es aún mejor converge a la función de distribución conjunta de las variables y las realizaciones de este para cada una de las variables por separado (En este caso cada una de las columnas de la matriz), además de ser realizaciones de una cadena de Markov, por si mismas convergen a las funciones marginales de estas variables. La demostración y otros detalles se encuentran en Explaining the Gibbs Sampler by George Casella.

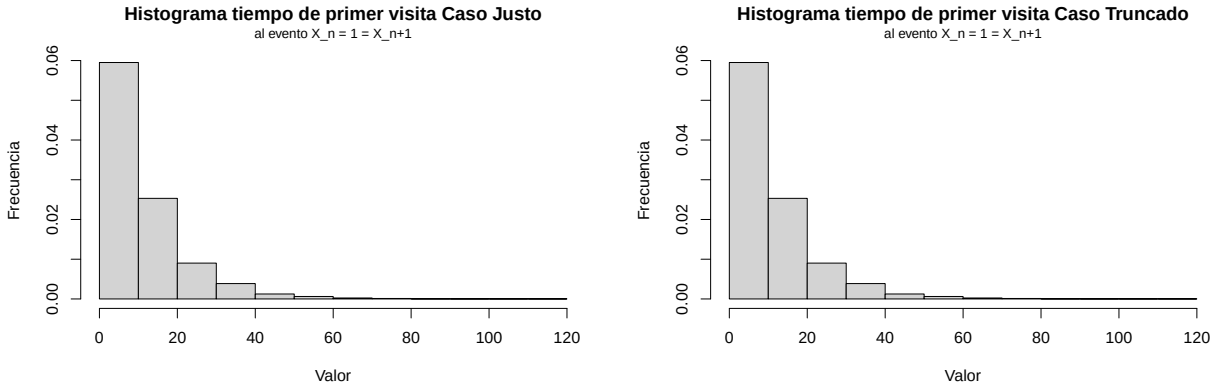
Por lo dicho anteriormente para responder la pregunta 1 lo que haremos será calcular 1000 muestras de Gibbs de tamaño 1000 pero para cada iteración del algoritmo del muestreador aumentamos un contador (Que nos dice cuántos elementos llevamos en la muestra con el fin de no tener que almacenar todas las realizaciones), verificaremos la condición $X_{n-1} = 1 = X_n$ y cuando la condición se cumpla detenemos el

algoritmo y guardamos el contador en el vector definido para ello. Esta operación se hará tanto para el caso en que se empieza con la moneda trucada como para el caso en que se empieza con la moneda justa.

```
# Orden del vector de probabilidades dado x = 0
# P(Y=0|X=0),P(Y=1,X=0)
x0 = c(0.5,0.5) # Vector de probabilidades para Y dado X=0
x1 = c(0.75,0.25) # Vector de probabilidades para Y dado X=1

y0 = c(0.75,0.25) # Vector de probabilidades para X dado Y=0
y1 = c(0.5,0.5) # Vector de probabilidades para X dado Y=1

#Vector para guardar cúanto tardó en llegar el muestreador
CasoTruncado <- c()

#Vector para guardar cúanto tardó en cumplirse la condición en el caso justo
CasoJusto <- c()

# Inicio ciclo 1000 muestras
for (i in 1:10000){
  # Iniciamos el caso truncado
  x = sample(c(0,1),size=1,prob=c(0.5,0.5))
  Counter = 1
  xm1 = 2
  while (!((x == 1) && (x == xm1))){
    if (Counter != 1){x=xm1}
    if (xm1 == 0){
      proba = x0
      } else {
        proba = x1
        }
    y = sample(c(0,1),size=1,prob=proba)
    if (y == 0){
      proba = y0
      } else {
        proba = y1
        }
    xm1 = sample(c(0,1),size=1,prob=proba)
    Counter = Counter+1
  }
  CasoJusto<- c(CasoJusto,Counter)

  # Iniciamos el caso truncado
  x = sample(c(0,1),size=1,prob=c(0.75,0.25))
  Counter = 1
  xm1 = 2
  while (!((x == 1) && (x == xm1))){
    if (Counter != 1){x=xm1}
    if (xm1 == 0){
      proba = x0
      } else {
        proba = x1
        }
    y = sample(c(0,1),size=1,prob=proba)
    if (y == 0){
```

```
      proba = y0
      } else {
        proba = y1
        }
    xm1 = sample(c(0,1),size=1,prob=proba)
    Counter = Counter+1
  }
  CasoTruncado <- c(CasoTruncado,Counter)
}
```

Una vez generadas las muestras podemos ver la distribución de los tiempos que tardaron




```
## [1] "La media de los tiempos para el caso de la moneda justa fue: 11.4863"
```

```
## [1] "La media de los tiempos para el caso de la moneda truncada fue: 12.121"
```

Y como las distribuciones de los tiempos son muy parecidas y las medias están muy cercanas no parece haber una diferencia significativa en escoger el primer lanzamiento con la moneda justa o la moneda trucada.

Finalmente y por temas de espacio intentaremos responder la pregunta 2 y hallar las distribuciones marginales generando una sola muestra de tamaño 100000 de Gibbs contando en cada caso cuántas muestras cumplen con la propiedad de ser $Cara, Cara$ y calculando la proporción de caras obtenidas por columnas de la matriz de muestras.

```
muestraG = c()
x = sample(c(0,1),size=1,prob=c(0.75,0.25))
# Es posible que experimente lentitud al correr el siguiente algoritmo.
for (i in 1:100000){
  if (x == 0){
    proba = x0
  } else {
    proba = x1
  }
  y = sample(c(0,1),size=1,prob=proba)
  muestraG = rbind(muestraG,c(x,y))
  if (y == 0){
    proba = y0
  } else {
    proba = y1
  }
  x = sample(c(0,1),size=1,prob=proba)
}
```

```
CaraCara = length(which(muestraG[, 1] == 0 & muestraG[, 2] == 0))/100000
print('La proporcion de casos en los que en una sola ')
```

```
## [1] "La proporcion de casos en los que en una sola "
```

```
print(paste('ronda salieron dos caras es de ', CaraCara))
```

```
## [1] "ronda salieron dos caras es de  0.32277"
```

Por otro lado, para el primer lanzamiento de cada ronda se tendrá apróximadamente la siguiente distribución:

Cara	0.3527
Sello	0.6473

Y para el segundo lanzamiento se tendrá:

Cara	0.41276
Sello	0.58724

Conclusiones

Si bien el método de Gibbs representa una gran ventaja en cuanto a permitirnos extraer muestras de distribuciones complejas también es cierto que tiene algunas cosas que no juegan muy a su favor como se vió en la sección anterior pues, por ejemplo, la velocidad de procesamiento en muchos casos es más lenta que la velocidad de otros métodos. Además en este caso estuvimos exentos de tener que pensar en otros problemas como por ejemplo la convergencia del modelo. Además se debe contar con todas las probabilidades condicionales por lo cual en algunas ocasiones no factible usar este método.

Dicho esto, cabe destacar que esta es una primera aproximación a este muestreador debido a que su principal uso se encuentra en el muestreo de algunas distribuciones mayormente utilizadas en estadística bayesiana y cabe destacar que este muestreador es un caso especial del algoritmo de Hasting-Metropolis (Visto en clase).

Para futuras ocasiones, ya teniendo más conocimiento sobre otras áreas sería interesante ver cómo se comporta el algoritmo para distribuciones más complejas y evidenciar las razones de lo que comunmente se conoce como quemado de muestras. Respecto a las preguntas como tal sería interesante revisar qué distribución tienen los tiempos de primer arribo al evento $Cara, Cara$ pues parece distribuirse exponencialmente así como también revisar las distribuciones del tiempo de primer arribo a otros estados de la distribución conjunta, además de esto podemos decir que la no afectación de la elección de la moneda justa o la moneda truncada era apenas de esperarse.

Como estudiantes en proceso de formación, se cree importante revisar la demostración formal de que en este caso el muestreador genera cadenas de Markov tanto para la distribución conjunta de las variables como para la distribución marginal para entender un poco más el algoritmo y sobre qué se basa para cuando se quiera utilizar en casos más complejos.

Finalmente, se hace la anotación de que no se hizo la exploración de la distribución conjunta debido a falta de espacio, pero si se revisa la respuesta a la pregunta 3 se puede más o menos formar un criterio sobre cómo sacar la aproximación de esta distribución de las muestras.