Cadena de Márkov y el baseball

Martin Martin del Campo Benito 150079

Escrito

Cadena de Márkov y el baseball

El baseball es un deporte conformado por 9 jugadores en cada equipo y consta de completar circuitos para generar puntos. La forma de puntaje es, pasa un bateador y tiene 3 posibilidades, hacer un Home run, correr entre 3 diamantes (Triplete, doble o simple) o tener un out. Si el jugador hace un home run, implica que el y en su caso donde exista otro jugador del mismo equipo en la cancha podrán terminar el circuito y sumar un punto por jugador. El juego consiste de 9 entradas y cada entrada se termina cuando se consigue 3 out.

\Nosotros nos vamos a basar en eventos básicos para simular cuantos puntos puede conseguir un equipo y eso será mediante una matriz de transición de 24 estados por entrada. Estos 24 estados serán de la siguiente forma:

• En realidad son 8 estados por out, ya sea 0,1 o 2 out, ya que el tercer out termina la entrada. • Vamos a necesitar un estado absorbente, que representa el tercer out, siendo un total de 25.

! Figura 1

En la imagen podemos observar los primeros 24 estados, fila 1 es cero out y así sucesivamente. La forma de (i,j) donde i representa las posiciones de los jugadores en la cancha y j la cantidad de out que han pasado en la entrada.

Pongamos un ejemplo. Si queremos pasar de (123,1) a (0,0) no es posible ya que los outs no desaparecen, pero si podríamos pasar de (123,1) a (0,1) si hacemos un Home run (No sería la única posibilidad). Al ser 9 entradas al final vamos a tener 9*24 = 216 estados, más el estado absorbente, nos da un total de 217 estados por juego lo que hace una matriz de 217x217. Y si sabemos que cada equipo tiene 9 jugadores, estos nos haría una matriz o un tensor de 9x217x217.

Vamos a necesitar otra matriz de las carreras y vamos a suponer que ningún jugador se vuela la base, entonces pueden avanzar cada vez que alguien batea y no hay outs, de igual manera solo se puede sumar puntos cuando no hay out. La matriz de jugadores la vamos hacer de la siguiente forma. En nuestro código usamos la función creada “crea_Matriz” y hace lo siguiente: • Hace la suma total de las estadísticas del jugador • Con ese total saca la probabilidad (caso favorable entre caso total) • Creamos una pequeña matriz de 8x8 que será replicada en toda la diagonal de la matriz de 217x217 • La matriz de 8x8 son las probabilidades de cambiar de estado sin que ocurra un out Ejemplo: Matriz8[1,] = c(h, b+s, d, t, 0, 0, 0, 0) empezando en (0,0) 1. Quedar en (0,0) es la probabilidad de un home run 2. Quedar en (1,0) es la probabilidad de hacer un simple o que te regalen la primer, por eso se suma. 3. Quedar en (0,2) es la probabilidad de hacer un doble 4. Quedar en (0,3) es la probabilidad de hacer un triplete 5. El resto no es posible ya que no hay más jugadores en la cancha.

! Figura 2

• Después Ajustamos la matriz según los out que han ocurrido y reemplazamos unos valores por la matriz de identidad • Al final para el tercer out hacemos otros reemplazos con los valores de 1 para así crear el estado absorbente y el por último valor de la matriz igual se le da un valor de 1 ya que es el final del juego. La matriz de Carrera se hace de una forma similar, excepto que la matriz de 8x8 usaremos la que venía con los datos. No hay necesidad de crearla. Después para cada bateador, volvemos a calcular nuestro vector de probabilidad de situación basad en la matriz de probabilidad de transición de estados de jugadores . Si S es el vector de situación y P es la matriz, la fórmula es solo S: = S * P También necesitamos hacer un seguimiento de cuántas carreras se anotaron en esos transiciones Si R es la matriz de carrea para cada transición, entonces R * P es el número esperado de carreras que el jugador tendrá para cada transición y, S * (R * P) es el vector del número esperado de carreras que el jugador tendrá si batea en cada estado, ponderado por la

probabilidad de cada estado. Entonces suma (S * (R * P)) da el número total de carreras esperadas cada que batea un jugador, si sumamos todas las sumas nos da el resultado del partido.

Los jugadores se eligen 9 de una base de forma aleatoria y en el orden que se toma es el orden en el que pasan los jugadores.

Simulaciones con la Base de jugadores

Leemos los 2 archivos, tanto la data como el la matriz 8x8 ya creada. Aparte hacemos un sample de 9, ya que solo juegan 9 jugadores en Baseball, no mas

```
setwd("C:/Users/marti/Downloads/Proyecto") # Hago la carpeta del trabajo la ruta principal

#Lectura de archivo
file<-"PlayerData.csv"
data<-read.csv(file, sep=",")#Base de jugadores
smallmat<-read.csv("smallmat.csv", sep=",", header = 0)#Matriz de 8x8 de carreras
smallmat[1][1]=1#Cuando abre la matriz la primer casilla trae strings, entonces le volvemos a asignar e
smallmat<-as.numeric(as.matrix(smallmat))#La hacemos numerica
runmatrix <- matrix(0, nrow = 217, ncol = 217) #Creamos una matriz de 217x271
data<-data[sample(nrow(data), 9), ]
```

Hacemos las 2 matrices, la de jugador y la de carrera

```
Jugadores = array(0, dim=c(9,217,217)) #Creamo nuestra matriz de 9x217X217 con puro 0


for (i in 1:(9*3)){
  runmatrix[((i-1)*8+1):((i-1)*8+8),((i-1)*8+1):((i-1)*8+8)] = smallmat #Se crear la matriz de carrera
}

crea_Matriz<- function(h,t,d,s,b,o){#Funcion para hacer la matriz de jugadores
  total = h + t + d + s + b + o #Suma de todos los valores
  h = h / total #Hace la probabilidad de que suceda h (casos favorables entre totales)
  t = t / total
  d = d / total
  s = s / total
  b = b / total
  o = o / total


  Matriz8 = array(0,c(8,8)) #Creamos nuestra matriz de 8x8 para la probabilidad de pasar de un estado a
  Matriz8[1,] = c(h, b+s, d, t, 0, 0, 0, 0) #Pasar de (0,0) a otro estado
  Matriz8[2,] = c(h, 0, d/2, t, b+s/2, s/2, d/2, 0)
  Matriz8[3,] = c(h, s/2, d, t, b, s/2, 0, 0)
  Matriz8[4,] = c(h, s, d, t, 0, b, 0, 0)
  Matriz8[5,] = c(h, 0, d/2, t, s/6, s/3, d/2, b+s/2)
  Matriz8[6,] = c(h, 0, d/2, t, s/2, s/2, d/2, b)
  Matriz8[7,] = c(h, s/2, d, t, 0, s/2, 0, b)
  Matriz8[8,] = c(h, 0, d/2, t, s/2, s/2, d/2, b)


  MatrizDeTransicion = array(0, dim=c(217 ,217 )) #Creamos la matriz de trancision

  for (i in 1:(27)){
```

```
    MatrizDeTransicion[((i-1)*8+1):((i-1)*8+8),((i-1)*8+1):((i-1)*8+8)] = Matriz8 #Hacemos la matriz8 l
  }

  for (i in 1:9){
    for (j in 1:2){
      MatrizDeTransicion[((i-1)*24+(j-1)*8+1):((i-1)*24+(j-1)*8+8),
               ((i-1)*24+(j-1)*8+9):((i-1)*24+(j-1)*8+16)] = o * diag(1,8) #Metemos la matriz identidad
    }
    MatrizDeTransicion[((i-1)*24+17) : ((i-1)*24+24) , (i*24+1)] = o*array(1,c(8,1)) #Se agrega el 3er
  }

  MatrizDeTransicion[217 , 217] <- 1 # Fin del juego, es un estado absorbente, ya que si termina el jue
  MatrizDeTransicion
}

for (i in 1:9){ # Se le da valores a la matriz de jugadores
  h<-data[i,1]
  t<-data[i,2]
  d<-data[i,3]
  s<-data[i,4]
  b<-data[i,5]
  o<-data[i,6]
  Jugadores[i,,]<-crea_Matriz(h,t,d,s,b,o)
}
```

Creamos la simulacion de juego segun los jugadores fueron selecionados

```
orden <- c(1,2,3,4,5,6,7,8,9)
situacion  <- as.vector(array(0,c(1,217))) #Creamos nuestra situacion de probabilidad
situacion [1] <- 1
carreras <- 0 # Numero de Carreras
bateador <- 1 #Bateador Bateando
while (situacion[217]<0.99){
  carreras<-carreras + sum(situacion %*% (runmatrix * Jugadores[orden[bateador],,])) #carreras + sum(S*
  situacion <- situacion %*% Jugadores[orden[bateador],,] #(S*P)
  bateador<- bateador +1
  if (bateador >9){ # Si llegamos al bateador 9, el siguiente debe de ser el primero, ya que no hay mas
    bateador <- 1
  }
}
```

```
paste(carreras, "es el numero de carreras esperadas en el orden que salieron del sample")
```

```
## [1] "2.54644879359699 es el numero de carreras esperadas en el orden que salieron del sample"
```

Creamos la simulacion de juego con los jugadores de atras hacia adelante

```
orden <- c(9,8,7,6,5,4,3,2,1)
situacion  <- as.vector(array(0,c(1,217)))
situacion [1] <- 1
carreras <- 0
bateador <- 1
while (situacion[217]<0.99){
  carreras<-carreras + sum(situacion %*% (runmatrix * Jugadores[orden[bateador],,]))
  situacion <- situacion %*% Jugadores[orden[bateador],,]
  bateador<- bateador +1
```

```
  if (bateador >9){
    bateador <- 1
  }
}
```

```
paste(carreras, "es el numero de carreras esperadas en el orden inverso de como salieron del sample")
```

```
## [1] "2.56408247409449 es el numero de carreras esperadas en el orden inverso de como salieron del sa
```

Creamos la simulacion de juego con los jugadores forma aleatoria

```
orden<- c(1,2,3,4,5,6,7,8,9)
orden<-sample(orden)
situacion  <- as.vector(array(0,c(1,217)))
situacion [1] <- 1
carreras <- 0
bateador <- 1
while (situacion[217]<0.99){
  carreras<-carreras + sum(situacion %*% (runmatrix * Jugadores[orden[bateador],,]))
  situacion <- situacion %*% Jugadores[orden[bateador],,]
  bateador<- bateador +1
  if (bateador >9){
    bateador <- 1
  }
}
orden
```

```
## [1] 4 1 2 7 8 9 5 3 6
```

```
carreras
```

```
## [1] 2.530182
```

```
paste(carreras, "es el numero de carreras esperadas con un orden aleatorio")
```

```
## [1] "2.53018176555782 es el numero de carreras esperadas con un orden aleatorio"
```

Como podemos observar de esta manera La variacion no nada grande, solo cambia por decimales, en cambio si corremos la simulcion con la otra informacion, donde esta mas acotada, ahi si es importante quien batea para sumar la mayor cantidad de puntos.

Simulaciones del Articulo

```
data9<-read.csv(file, sep=",")
Jugadores9 = array(0, dim=c(9,217,217))
for (i in 1:9){
  h<-data9[i,1]
  t<-data9[i,2]
  d<-data9[i,3]
  s<-data9[i,4]
  b<-data9[i,5]
  o<-data9[i,6]
  Jugadores9[i,,]<-crea_Matriz(h,t,d,s,b,o)
}
```

```
orden <- c(1,2,3,4,5,6,7,8,9)
situacion  <- as.vector(array(0,c(1,217)))
situacion [1] <- 1
carreras <- 0
bateador <- 1
while (situacion[217]<0.99){
  carreras<-carreras + sum(situacion %*% (runmatrix * Jugadores9[orden[bateador],,]))
  situacion <- situacion %*% Jugadores9[orden[bateador],,]
  bateador<- bateador +1
  if (bateador >9){
    bateador <- 1
  }
}
```

```
paste(carreras, "es el numero de carreras esperadas en orden")
```

```
## [1] "4.24495661901513 es el numero de carreras esperadas en orden"
```

```
orden<- c(1,1,1,1,1,1,1,1,1)
orden<-sample(orden)
situacion  <- as.vector(array(0,c(1,217)))
situacion [1] <- 1
carreras <- 0
bateador <- 1
while (situacion[217]<0.99){
  carreras<-carreras + sum(situacion %*% (runmatrix * Jugadores9[orden[bateador],,]))
  situacion <- situacion %*% Jugadores9[orden[bateador],,]
  bateador<- bateador +1
  if (bateador >9){
    bateador <- 1
  }
}
```

```
paste(carreras, "es el numero de carreras esperadas con solo el jugador 1")
```

```
## [1] "7.28744681765448 es el numero de carreras esperadas con solo el jugador 1"
```

La diferencia presentada es de 3 carreras, por lo que al equipo le gustaria tener a los 9 jugadores con estadisticas similares al primer jugador

Conclusiones:

Basado en el paper, las simulaciones por Márkov pueden llegar a tener un margen de error de hasta 7%. Comprobando con nuestras simulaciones también nos damos cuenta que con la base usada, el orden de los jugadores no afectan tanto, como puede ser que tomes a los 9 jugadores como el mismo, todos por igual es con una data limitada a solo 9 jugadores. Con la data del mismo jugador puede variar más de 4 carreras, mientras con una data "PlayerData.csv" de más de 41 mil jugadores la diferencia es de menor de una carrea, solo es por fracción. Los primeros 9 renglones del .csv son los 9 registros de la data del paper, y así enseñar las simulaciones con esos 9, misma que hacer el documento. Hay formas de mejorar la simulación, por ejemplo hacer la data por equipo, para ser más realistas, meter los eventos secundarios, como robar base, etc. Se podría hacer comparativa entre equipos y sería ideal poder meter a la defensa para simular un partido completo y no solo a la ofensiva.

Referencias

- Base de datos http://www.seanlahman.com/baseball-archive/statistics/
- An Intuitive Markov Chain Lesson From Baseball https://pubsonline.informs.org/doi/pdf/10.1287/ited.5.1.47
- FINDING BETTER BATTING ORDERS http://www.pankin.com/markov/btn1191.htm -The Markov Chain Model of Baseball http://statshacker.com/blog/2018/05/07/the-markov-chain-model-of-baseball/