



**Predicción del resultado más probable en el partido Manchester City 1-2 Chelsea en un simulación de 1000 partidos**

# Índice

Análisis del código.....	3
1. Lectura del fichero .....	3
2. Descripción y resumen estadístico del fichero .....	3
3. Cálculo de resultado más frecuente del partido analizado.....	4
a) Proceso de simulación de partidos. ....	4
b) Gráficas que ayudan a entender el resultado.....	5
4. Calcular los xPoints asociados a cada equipo.....	6

# Análisis del código

## 1. Lectura del fichero

```
1
2 # 1. Lectura del fichero
3
4 setwd("C:/Users/pepec/Documents/Master/PM-Estadística/")
5
6 # Listado de archivos en mi directorio actual
7 list.files()
8
9 # Lectura del csv
10 shots_df <- read.csv(
11   file = "Data/Understat_City_Chelsea.csv",
12   sep = ";",
13   header = TRUE,
14   encoding = "UTF-8"
15 )
16
```

En primer lugar, seteamos ese path como nuestro directorio actual. Ahí, contenemos todos los archivos correspondientes al Premáster Curso en Estadística y Matemáticas aplicadas al deporte con R

A continuación, leemos los archivos del directorio previamente establecido y leemos el archivo .csv que contiene las estadísticas del partido Manchester City vs Chelsea. El dataframe se llamará shots\_df.

## 2. Descripción y resumen estadístico del fichero

```
18 # 2. Descripción y resumen estadístico del fichero
19
20 # Nombre de las columnas
21 colnames(shots_df)
22
23 # Número de columnas del dataframe
24 cat("Numero de columnas: ", ncol(shots_df))
25
26 # Número de disparos
27 cat("Numero de disparos (filas): ", nrow(shots_df))
28
29 # Características del dataframe
30 str(shots_df)
31
32 # Goles esperados del Manchester City y del Chelsea
33 cat("xG del Manchester City: ", sum(shots_df[shots_df$team == 'Manchester City', 'xG']))
34 cat("xG del Chelsea: ", sum(shots_df[shots_df$team == 'Chelsea', 'xG']))
35
36 # Goles que hubo en el partido
37 nrow(shots_df[shots_df$result == 'Goal', ])
38
```

En esta parte visualizamos el nombre de las columnas, el número de columnas y filas (disparos) del dataframe, el xG de cada equipo y los goles que hubo en el partido. Como toda la información referente al partido está almacenada en el dataframe shots\_df, hacemos un filtrado según la columna team dependiendo del equipo y obtenemos solo su xG. A ese mismo código se le añade la función sum que logra la suma de todos los goles esperados.

Para saber el número de goles, hacemos un filtrado de la columna result. Esta puede adquirir varios valores (tiro fuera, parada del portero, tiro bloqueado o gol). En este caso, mostraremos las filas cuando la columna result de nuestro dataframe sea igual a Goal.

### 3. Cálculo de resultado más frecuente del partido analizado

#### a) Proceso de simulación de partidos.

```
41 # a. Proceso de simulación de partidos.
42
43 N <- 1000 # Número de partidos a analizar
44 results <- data.frame() # Creamos el dataframe results
45
46 for (j in 1:N)
47 {
48
49   # Semilla de aleatoriedad
50   set.seed(j)
51
52   # Almacenaremos aquí los resultados de los partidos, estando al principio vacío
53   results_match <- c(NA, NA)
54   |
55   # Iterador de posición para el vector de resultados
56   k <- 1
57
58   # Bucle que recorre los distintos equipos (2)
59   for (team in unique(shots_df$team))
60   {
61
62     # Almacenamos los tiros de cada equipo
63     shots_team <- shots_df[shots_df$team == team, ]
64
65     # Iniciamos la variable a 0. Dependiendo de la binomial, se convertirá en 0 o 1
66     goals_team <- 0
67
68     # Bucle que corre los tiros de cada equipo
69     for (i in 1:nrow(shots_team))
70     {
71       # Según el xG, goals_team será 0 o 1.
72       goals_team <- goals_team + rbinom(n=1, size=1, prob=shots_df$xG[i])
73     }
74
75     # Almacenamos en la posición del equipo sus goles
76     results_match[k] <- goals_team
77     k <- k + 1
78   }
79
80   # Concatenamos los resultados
81   results <- rbind(results, results_match)
82   # Renombramos las columnas
83   colnames(results) <- unique(shots_df$team)
84
85 }
```

En este paso, simularemos 1000 partidos y, según el gol esperado de cada disparo, cogerá un valor de 0 (si no es gol) o 1 (si es gol).

Para ello, la variable N será el número de partidos a recorrer y creamos el dataframe results para almacenar los resultados. Creamos un bucle en el que j es igual a uno y en cada iteración será su valor más uno, hasta llegar a 1000. En primer lugar, establecemos la semilla de aleatoriedad para que no coja valores aleatorios. Creamos un vector vacío que almacenará los resultados de cada simulación y k igual a 1 para la posición de los equipos.

Después, recorreremos para cada equipo sus disparos, almacenados en la variable shots\_team. Según el gol esperado de cada tiro, se le asocia 0 o 1 según la función rbinom,

que generará un valor ( $n=1$ ) por cada ensayo ( $size=1$ ) con la probabilidad del gol esperado ( $prob=shots\_df\$xG[i]$ ).

El valor que salga será el valor del vector de `results_match` según su equipo. El vector tiene dos posiciones y se inicializa la primera posición con 1. Para acceder al primer valor del vector se usa `results_match[1]`. Así, gracias a la variable `k` creada anteriormente podremos almacenar los goles de cada equipo en la posición correcta del vector.

Por último, concatenamos los valores de este vector con el dataframe `result` para que se almacenen en forma de dataframe, y nombramos a las columnas del dataframe con los nombres de los equipos.

#### b) Gráficas que ayudan a entender el resultado.

```
87 # b. Gráficas que ayudan a entender el resultado.
88
89 # Añadimos una columna al dataframe con el resultado en forma de string
90 results$results_match <- paste(as.character(results$Manchester City), "-", as.character(results$Chelsea))
91
92 # Añadimos una columna al dataframe con el equipo que ha ganado el partido.
93 results$win <- ifelse(results$Manchester City > results$Chelsea, 'Manchester City',
94                     ifelse(results$Chelsea > results$Manchester City, 'Chelsea', 'Empate'))
95
96 # Creamos una tabla con los resultados. Irán ordenados de mayor a menor según se repitan
97 results_table <- sort(table(results$results_match), decreasing = TRUE)
98
99 # Resultado que más se repite: 2-1
100 results_table
101
102 # Obtenemos la posición del resultado
103 pos_real_result <- which(names(results_table) == "2 - 1")
104
105 # Colores para los resultados
106 colors <- rep("gray", length(results_table))
107
108 # El color del resultado más repetido será diferente
109 colors[pos_real_result] <- "skyblue"
110
111 # Creamos el barplot con los resultados, los colores establecidos, y los nombres de x, y y el título
112 barplot(results_table,
113         col = colors,
114         main = 'Resultado más frecuente: Manchester City 2-1 Chelsea',
115         xlab = 'Resultado',
116         ylab = 'Partidos',
117         las=2
118         )
119
120 # Creamos una tabla con las proporciones de los resultados
121 win_proportions <- prop.table(table(results$win))
122
123 # Vemos los resultados y el orden
124 win_proportions
125
126 # Establecemos los colores del gráfico
127 colors <- c("blue", "gray", "skyblue")
128
129 # Establecemos los campos del gráfico
130 labels <- paste(teams, round(100*win_proportions, 2), "%")
131
132 # Creamos el gráfico de sectores con las proporciones, colores y campos establecidos
133 pie(win_proportions,
134     col = colors,
135     labels = labels,
136     main = "Proporción de victoria tras 1000 simulaciones")
137
138 # Creamos una leyenda abajo a la izquierda
139 legend("bottomleft", names(win_proportions), fill = colors, pt.cex = 1, bty = 'n', cex = 0.7)
140
```

Crearemos dos gráficos: gráfico de barras y gráfico de sectores.

El gráfico de barras representará los resultados más probables del partido. Para ello, crearemos la columna `results_match` en nuestro dataframe que tendrá el resultado en forma de string (ejemplo: "1 - 1"). Después, crearemos otra columna llamada `win` que tendrá el nombre del equipo ganador según el resultado.

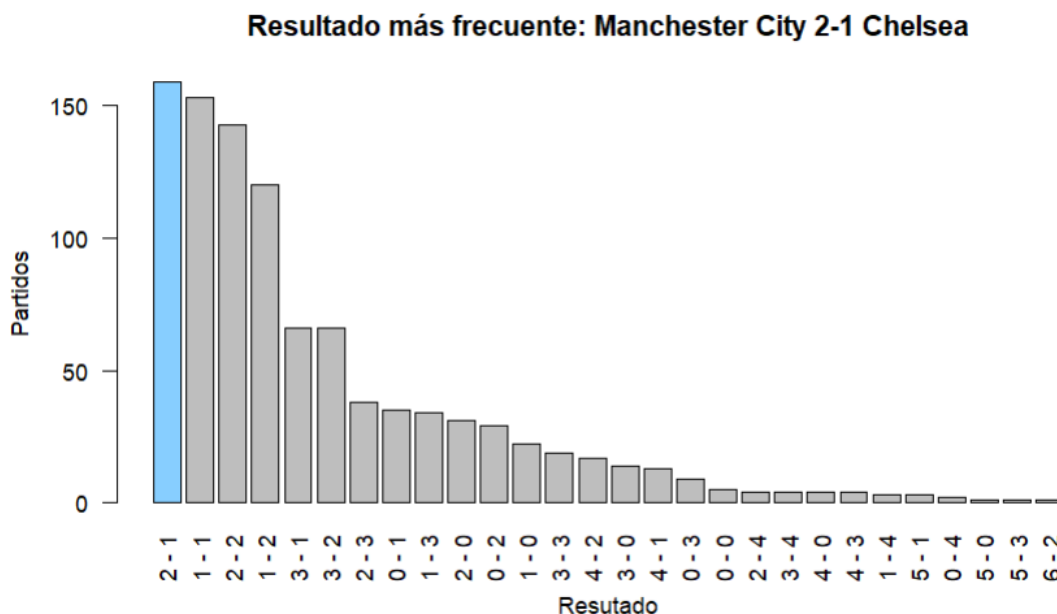
A continuación, crearemos una tabla con los resultados ordenados de mayor repetición a menor, obtenemos la posición del resultado más repetido (siempre será la posición nº1) y establecemos un color diferente para el resultado.

Para el gráfico de sectores, necesitamos las proporciones de resultado (victoria local, empate, victoria visitante). Esto se hace creando una tabla de proporciones, que nos servirá para crear el vector de colores para cada resultado. Para su representación, establecemos los valores de las proporciones en forma de porcentaje. Por último, crearemos una leyenda para mejorar el entendimiento del gráfico.

#### 4. Calcular los xPoints asociados a cada equipo.

```
142 # 4. Calcular los xPoints asociados a cada equipo.
143
144 # Teniendo en cuenta el cálculo de los puntos esperados:
145 MC_xPTS <- as.character(win_proportions["Manchester City"]*3 + win_proportions["Empate"])
146 Ch_xPTS <- as.character(win_proportions["Chelsea"]*3 + win_proportions["Empate"])
147
148 cat(paste("Expected Points: Manchester City", MC_xPTS, "-", Ch_xPTS, "Chelsea"))
```

Los puntos esperados para cada equipo se calculan multiplicando su proporción de victorias por 3 (puntos que consigues en una victoria) y se suma la proporción del empate.



### Proporción de victoria tras 1000 simulaciones

