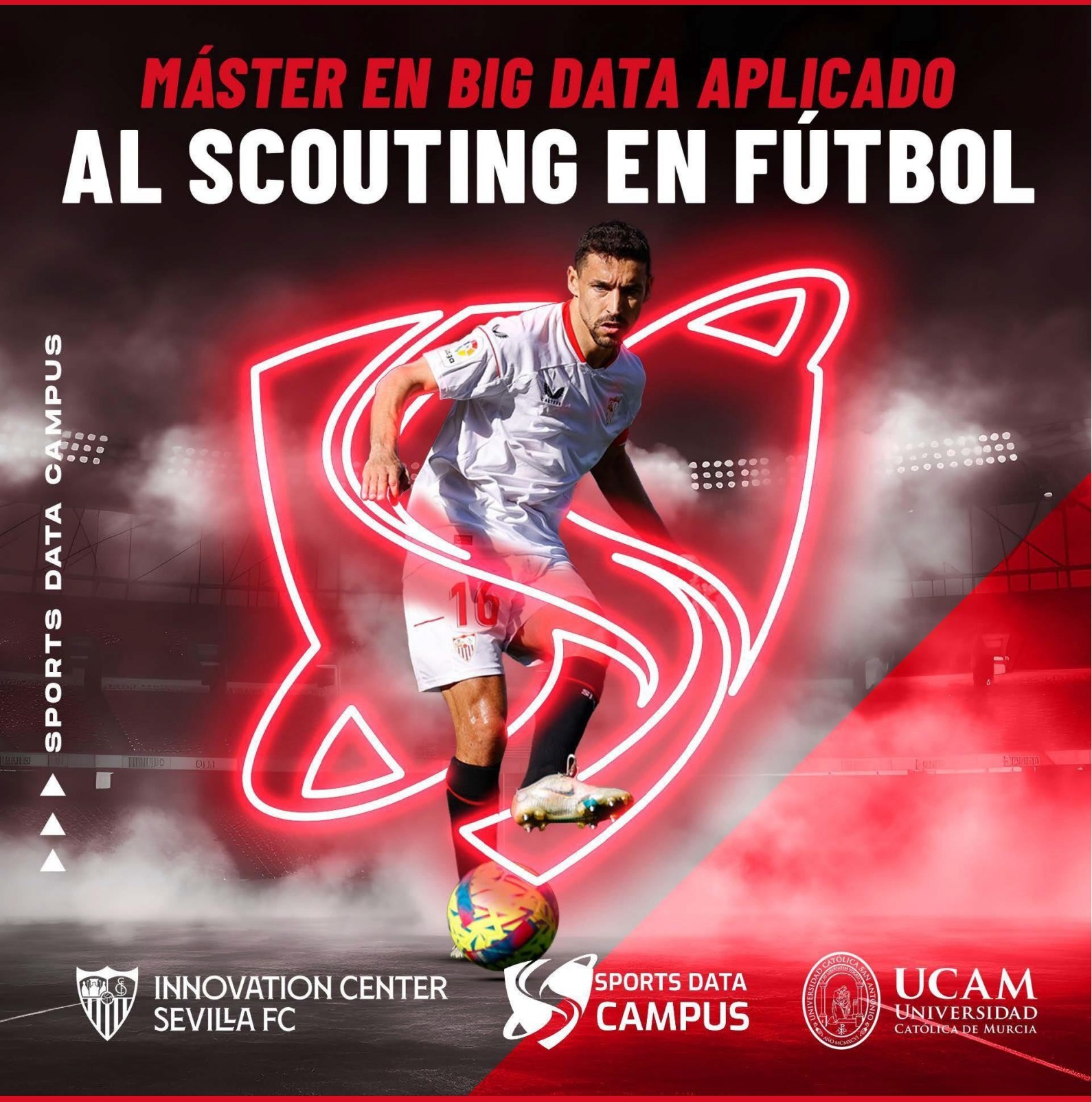
SPORTS DATA CAMPUS

**MÁSTER EN BIG DATA APLICADO AL SCOUTING EN FÚTBOL**
**Predicción del resultado más probable en el partido**
**Manchester City 1-2 Chelsea**
**en un simulación de 1000 partidos**




SPORTS DATA CAMPUS
UCAM UNIVERSIDAD CATÓLICA DE MURCIA
ENIIT INNOVA IT BUSINESS SCHOOL
BIG DATA International Campus

Índice

1. Análisis del código .....3

# 1. Análisis del código

```r

# 1. Lectura del fichero

setwd("C:/Users/pepec/Documents/Master/PM-Estadística/")

# Listado de archivos en mi directorio actual
list.files()

# Lectura del csv
shots_df <- read.csv(
  file = "Data/Understat_City_Chelsea.csv",
  sep = ";",
  header = TRUE,
  encoding = "UTF-8"
)
```

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.

```r
# 2. Descripción y resumen estadístico del fichero

# Nombre de las columnas
colnames(shots_df)

# Número de columnas del dataframe
cat("Numero de columnas: ", ncol(shots_df))

# Número de disparos
cat("Numero de disparos (filas): ", nrow(shots_df))

# Características del dataframe
str(shots_df)

# Goles esperados del Manchester City y del Chelsea
cat("xG del Manchester City: ", sum(shots_df[shots_df$team == 'Manchester City', 'xG']))
cat("xG del Chelsea: ", sum(shots_df[shots_df$team == 'Chelsea', 'xG']))

# Goles que hubo en el partido
nrow(shots_df[shots_df$result == 'Goal', ])

```

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.

```r
# a. Proceso de simulación de partidos.

N <- 1000 # Número de partidos a analizar
results <- data.frame() # Creamos el dataframe results

for (j in 1:N)
{

  # Semilla de aletoriedad
  set.seed(j)

  # Almacenaremos aquí los resultados de los partidos, estando al principio vacío
  results_match <- c(NA, NA)

  # Iterador de posición para el vector de resultados
  k <- 1

  # Bucle que recorre los distintos equipos (2)
  for (team in unique(shots_df$team))
  {

    # Almacenamos los tiros de cada equipo
    shots_team <- shots_df[shots_df$team == team, ]

    # Iniciamos la variable a 0. Dependiendo de la binomial, se convertirá en 0 o 1
    goals_team <- 0

    # Bucle que corre los tiros de cada equipo
    for (i in 1:nrow(shots_team))
    {
      # Según el xG, goals_team será 0 o 1.
      goals_team <- goals_team + rbinom(n=1, size=1, prob=shots_df$xG[i])
    }

    # Almacenamos en la posición del equipo sus goles
    results_match[k] <- goals_team
    k <- k + 1
  }

  # Concatenamos los resultados
  results <- rbind(results, results_match)
  # Renombramos las columnas
  colnames(results) <- unique(shots_df$team)

}
```

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 aletoriedad 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, recorremos 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.

```r
# b. Gráficas que ayudan a entender el resultado.

# Añadimos una columna al dataframe con el resultado en forma de string
results$results_match <- paste(as.character(results$`Manchester City`), "-", as.character(results$Chelsea))

# Añadimos una columna al dataframe con el equipo que ha ganado el partido.
results$win <- ifelse(results$`Manchester City` > results$Chelsea, 'Manchester City',
                      ifelse(results$Chelsea > results$`Manchester City`, 'Chelsea', 'Empate'))

# Creamos una tabla con los resultados. Irán ordenados de mayor a menor según se repitan
results_table <- sort(table(results$results_match), decreasing = TRUE)

# Resultado que más se repite: 2-1
results_table

# Obtenemos la posición del resultado
pos_real_result <- which(names(results_table) == "2 - 1")

# Colores para los resultados
colors <- rep("gray", length(results_table))

# El color del resultado más repetido será diferente
colors[pos_real_result] <- "skyblue1"

# Creamos el barplot con los resultados, los colores establecidos, y los nombres de X, Y y el título
barplot(results_table,
        col = colors,
        main = 'Resultado más frecuente: Manchester City 2-1 Chelsea',
        xlab = 'Resutado',
        ylab = 'Partidos',
        las=2
        )

# Creamos una tabla con las proporciones de los resultados
win_proportions <- prop.table(table(results$win))

# Vemos los resultados y el orden
win_proportions

# Establecemos los colores del gráfico
colors <- c("blue", "gray", "skyblue1")

# Establecemos los campos del gráfico
labels <- paste(teams, round(100*win_proportions, 2), "%")

# Creamos el gráfico de sectores con las proporciones, colores y campos establecidos
pie(win_proportions,
    col = colors,
    labels = labels,
    main = "Proporción de victoria tras 1000 simulaciones")

# Creamos una leyenda abajo a la izquierda
legend("bottomleft", names(win_proportions), fill = colors, pt.cex = 1, bty = 'n', cex = 0.7)

```

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.

```
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.