Práctica 2: Limpieza y análisis de datos

UOC - Tipología y ciclo de vida de los datos

Miguel Santos Pérez y Alejandro Arzola García

17 de junio de 2020

Índice

```r
 # Cargamos las librerías necesarias para el proyecto
suppressMessages(library(dplyr))
suppressMessages(library(plyr))
```

1 Introducción y descripción de los datasets

En la Práctica 1 de la asginatura de *Tipología y ciclo de vida de los datos* se creó un proceso *web scraping* en el que se obtuvieron los datos relacionados a todos los partidos y las estadísticas individuales de todos los jugadores de la actual temporada (2019-2020) de la Euroliga, la máxima competición de clubes de baloncesto de Europa. El objetivo final era contribuir al aumento de explotación de los datos para los equipos europeos y tratar de igualar lo que se realiza en América con las poderosas franquicias de NBA.

Mediante el proceso de obtención de datos realizado en *Python* que comentamos anteriormente, conseguimos crear dos datasets: el primero contiene los datos generales de un partido (nombre de los equipos, puntuación de cada equipo, fecha y hora del partido) y el segundo contiene más de quince estadísticas individuales para cada jugador por cada partido (puntos, minutos, rebotes, asistencias, ...). Este trabajo está disponible en un repositorio de *GitHub* al que se puede acceder haciendo click aquí.

El primer dataset (`euroleague_scoreboards.csv`) contiene las siguientes variables:

- **Id:** identificador único de partido.
- **Date:** fecha del partido.
- **HomeTeam:** nombre del equipo local.
- **HomeScore:** puntos anotados por el equipo local.
- **VisitingTeam:** nombre del equipo visitante.
- **VisitingScore:** puntos anotados por el equipo visitante.
- **Link:** enlace a la página web con las estadísticas individuales del partido.

El segundo dataset (`euroleague_stats_per_game.csv`) contiene las siguientes variables:

- **MatchId:** identificador único de partido.
- **Team:** equipo al que pertenece el jugador.
- **PlayerNumber:** número del jugador.
- **PlayerName:** nombre del jugador.
- **Min:** minutos jugados.
- **Pts:** puntos anotados.
- **2FG:** tiros de dos (metidos-anotados).
- **3FG:** tiros de tres (metidos-anotados).
- **FT:** tiros libres (metidos-anotados).
- **O:** rebotes ofensivos.
- **D:** rebotes defensivos.
- **T:** rebotes totales.
- **As:** asistencias.
- **St:** recuperaciones.
- **To:** pérdidas.
- **Fv:** tapones a favor.
- **Ag:** tapones en contra.
- **Cm:** faltas cometidas.
- **Rv:** faltas recibidas.
- **PIR:** valoración del jugador.

En esta segunda práctica queremos seguir progresando para lograr el objetivo de igualarnos con la NBA y por ello vamos a utilizar los datasets que hemos mencionado para realizar un análisis estadístico. Este análisis se va a enfocar en analizar las estadísticas globales por partido de cada equipo para tratar de estimar y predecir las victorias de un equipo o definir qué equipos defienden mejor, cuales tienen un mejor ataque y aquellos que tienen una forma de jugar más equilibrada.

Los datasets y el código utilizado para generar el análisis estadístico que se desarrolla a continuación está disponible en el siguiente repositorio de *GitHub*:

- https://github.com/aarzola-uoc/practica2-tycvd

2 Integración y selección de los datos de interés a analizar

Los datos relevantes del primer dataset son exclusivamente los nombres de los equipos y el marcador. Sin embargo, estos datos son posibles calcularlos a partir del segundo dataset, incluso aumentar los datos de equipo ya que son la suma de las estadísticas de todos los jugadores para cada partido.

En base a esto, hemos decidido que se creará un nuevo dataset utilizando código R, a partir del dataset de estadísticas de los jugadores, que tendrá las estadísticas de cada equipo por partido (suma de cada jugador es el total del equipo). El primer dataset se utilizará únicamente como referencia para obtener el equipo oponente de cada partido.

Además vamos a calcular una serie de estadísticos muy interesantes y que permiten un análisis mucho más técnico y útli para los equipos. Estos estadísticos son el ritmo de juego, eficiencia ofensiva/defensiva, porcentaje efectivo de tiros de campo, tasa de rebote y algunos más. La explicación de cada uno de estos estadísticos y su correspondiente fórmula se explicará más adelante.

---

3 Limpieza de los datos

El primer paso que tenemos que realizar es cargar los ficheros csv `euroleague_stats_per_game.csv` y `euroleague_scoreboards.csv`. Para ello vamos a utilizar la función `read.csv2` ya que el fichero a cargar está en formato csv español (los separadores son el caracter `;`). Como resultado obtendremos dos objetos `data.frame` que contendrán todos los datos de nuestros dos datasets:

```
data_scoreboards <- read.csv2('../csv/euroleague_scoreboards.csv', header = TRUE)
data_euroleague <- read.csv2('../csv/euroleague_stats_per_game.csv', header = TRUE)
```

Mostramos los primeros registros de ambos datasets para verificar que los datos se han cargado correctamente:

```
# Primer dataset (euroleague_scoreboards.csv)
head(data_scoreboards)
```

```
##   Id                Date                  HomeTeam HomeScore
## 1  1 October 3 19:00 CET      Khimki Moscow Region        89
## 2  2 October 3 20:00 CET Panathinaikos OPAP Athens        87
## 3  3 October 3 20:30 CET          FC Bayern Munich        78
## 4  4 October 3 21:00 CET               Real Madrid        81
## 5  5 October 4 19:00 CET           Zalgiris Kaunas        58
## 6  6 October 4 19:30 CET     Anadolu Efes Istanbul        64
##                        VisitingTeam VisitingScore
## 1              Maccabi FOX Tel Aviv            83
## 2        Crvena Zvezda mts Belgrade            82
## 3          AX Armani Exchange Milan            64
## 4          Fenerbahce Beko Istanbul            77
## 5 KIROLBET Baskonia Vitoria-Gasteiz            70
## 6                      FC Barcelona            74
##                                                                                 Link
## 1 https://www.euroleague.net/main/results/showgame?gamecode=1&seasoncode=E2019
## 2 https://www.euroleague.net/main/results/showgame?gamecode=8&seasoncode=E2019
## 3 https://www.euroleague.net/main/results/showgame?gamecode=2&seasoncode=E2019
## 4 https://www.euroleague.net/main/results/showgame?gamecode=4&seasoncode=E2019
## 5 https://www.euroleague.net/main/results/showgame?gamecode=5&seasoncode=E2019
## 6 https://www.euroleague.net/main/results/showgame?gamecode=6&seasoncode=E2019
```

```r
# Segundo dataset (euroleague_stats_per_game.csv)
head(data_euroleague)
```

```
##   MatchId                 Team PlayerNumber          PlayerName   Min Pts X2FG
## 1       1 Khimki Moscow Region            1       SHVED, ALEXEY 30:19  22  4/7
## 2       1 Khimki Moscow Region            5       BOOKER, DEVIN 19:21   4  1/4
## 3       1 Khimki Moscow Region            6        TIMMA, JANIS 33:53  17  1/3
## 4       1 Khimki Moscow Region            8 ZAYTSEV, VYACHESLAV 10:23   0    0
## 5       1 Khimki Moscow Region            9      VIALTSEV, EGOR  4:58   0    0
## 6       1 Khimki Moscow Region           10 DESIATNIKOV, ANDREI   DNP   -    -
##   X3FG   FT O D T As St To Fv Ag Cm Rv PIR
## 1 2/10 8/10 0 2 2  6  0  3  1  0  1  5  19
## 2    0  2/2 3 3 6  0  2  1  0  0  2  1   7
## 3 5/12    0 1 3 4  1  4  1  0  0  5  2  13
## 4    0    0 1 1 2  2  1  1  0  0  2  1   3
## 5    0    0 0 0 0  1  0  0  0  0  2  0  -1
## 6    -    - - - -  -  -  -  -  -  -  -   -
```

La variable MatchId es una variable que necesitamos para cruzar ambos datasets. Sin embargo, en el primer dataset, el nombre de esta variable es Id por lo que vamos a homogenizar su nombre:

```r
names(data_scoreboards)[names(data_scoreboards) == "Id"] <- "MatchId"
```

En total disponemos de estadísticas de 252 partidos y de 306 jugadores diferentes.

3.1 Elementos vacíos

Los datos vacíos o no definidos pueden presentarse en distintos formatos, normalmente en forma de cadena de caracteres vacía (`""`) o `NA` (*Not Available en inglés*), pero en algunos contextos pueden incluso tomar valores numéricos como 0 o 999. Es fundamental identificar para cada variable los valores que indiquen que existe una pérdida de datos para poder aplicar una solución y que no afecte en la calidad de los análisis estadísticos que se realicen posteriormente.

Procedemos a buscar valores vacíos en nuestro primer dataset:

```r
colSums(is.na(data_scoreboards))
```

```
##       MatchId          Date      HomeTeam     HomeScore  VisitingTeam
##             0             0             0             0             0
## VisitingScore          Link
##             0             0
```

```r
colSums(data_scoreboards=="")
```

```
##       MatchId          Date      HomeTeam     HomeScore  VisitingTeam
##             0             0             0             0             0
## VisitingScore          Link
##             0             0
```

Como se puede observar no se ha detectado ningún elemento vacío del tipo `NA` o cadena de caracteres vacía. En el caso de que se haya utilizado valores numéricos fuera del dominio del atributo lo podremos detectar en el análisis de valores extremos.

Realizamos el mismo análisis para el segundo dataset. En este caso, como hemos creado nosotros el dataset, sabemos que se ha utilizado el caracter `"-"` para indicar las estadísticas de los jugadores que no han jugado el partido y `"DNP"` (*Did Not Play*) en el atributo de tiempo:

```
colSums(is.na(data_euroleague))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0          504            0            0            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0            0            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="-")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0            0          492
##         X2FG         X3FG           FT            0            D            T
##          492          492          492          492          492          492
##           As           St           To           Fv           Ag           Cm
##          492          492          492          492          492          492
##           Rv          PIR
##          492          492
```

```
colSums(data_euroleague=="DNP")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0           NA            0          492            0
##         X2FG         X3FG           FT            0            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

Para solucionar este problema y poder realizar un análisis estadístico en el que se pueda asegurar un nivel alto de calidad de datos, hemos optado por informar todos estos campos con el valor 0:

```
data_euroleague[is.na(data_euroleague)] <- 0
data_euroleague[data_euroleague=="-"] <- 0
data_euroleague[data_euroleague=="DNP"] <- 0

colSums(is.na(data_euroleague))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="-")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

```
colSums(data_euroleague=="DNP")
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##            0            0            0            0            0            0
##         X2FG         X3FG           FT            O            D            T
##            0            0            0            0            0            0
##           As           St           To           Fv           Ag           Cm
##            0            0            0            0            0            0
##           Rv          PIR
##            0            0
```

Hemos vuelto a buscar elementos vacíos para comprobar que se han solucionado todos los problemas que habíamos detectado y como se puede ver en los fragmentos anteriores se han resuelto todos correctamente.

3.2 Tratamiento de variables

El siguiente paso consisten en analizar el tipo de dato que ha asignado R para cada una de nuestras variables al cargar los datasets:

```
str(data_scoreboards)
```

```
## 'data.frame':    252 obs. of  7 variables:
##  $ MatchId      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Date         : Factor w/ 218 levels "December 12 18:00 CET",..: 203 204 205 206 214 215 216 217 2
##  $ HomeTeam     : Factor w/ 18 levels "ALBA Berlin",..: 9 14 7 15 17 2 1 11 16 5 ...
##  $ HomeScore    : int  89 87 78 81 58 64 85 82 71 79 ...
##  $ VisitingTeam : Factor w/ 18 levels "ALBA Berlin",..: 12 4 3 8 10 6 18 13 5 7 ...
##  $ VisitingScore: int  83 82 64 77 70 74 65 63 96 68 ...
##  $ Link         : Factor w/ 252 levels "https://www.euroleague.net/main/results/showgame?gamecode=1&s
```

Para el dataset `data_scoreboards` se han interpretado correctamenta las variables `MatchId`, `HomeScore` y `VisitingTeam` con el tipo de dato `int` ya que son variables cuantitativas discretas, mientras que el resto de variables (`Date`, `HomeTeam` y `VisitingTeam`) que son variables cualitativas nominales, se han interpretado como tipo factor.

Verificamos ahora el segundo dataset:

```
str(data_euroleague)
```

```
## 'data.frame':    6505 obs. of  20 variables:
##  $ MatchId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Team        : Factor w/ 18 levels "ALBA Berlin",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ PlayerNumber: num  1 5 6 8 9 10 11 12 13 24 ...
##  $ PlayerName  : Factor w/ 306 levels "ABALDE, ALBERTO",..: 250 28 276 303 288 67 127 184 97 132 ...
##  $ Min         : Factor w/ 1910 levels "0","0:01","0:02",..: 1345 626 1502 99 1645 1 976 617 924 486
##  $ Pts         : Factor w/ 41 levels "-","0","1","10",..: 17 33 11 2 2 2 8 38 12 2 ...
##  $ X2FG        : Factor w/ 92 levels "-","0","0/1",..: 49 14 13 2 2 2 32 2 31 2 ...
##  $ X3FG        : Factor w/ 74 levels "-","0","0/1",..: 22 2 58 2 2 2 28 25 38 4 ...
##  $ FT          : Factor w/ 66 levels "-","0","0/1",..: 58 22 2 2 2 2 38 2 42 2 ...
##  $ O           : Factor w/ 10 levels "-","0","1","2",..: 2 5 3 3 2 2 3 2 5 3 ...
##  $ D           : Factor w/ 14 levels "-","0","1","10",..: 7 8 8 3 2 2 11 7 8 2 ...
##  $ T           : Factor w/ 19 levels "-","0","1","10",..: 12 16 14 12 2 2 17 12 16 3 ...
##  $ As          : Factor w/ 19 levels "-","0","1","10",..: 16 2 3 12 3 2 2 2 14 14 ...
##  $ St          : Factor w/ 8 levels "-","0","1","2",..: 2 4 6 3 2 2 4 3 2 2 ...
##  $ To          : Factor w/ 11 levels "-","0","1","2",..: 5 3 3 3 2 2 3 2 2 3 ...
##  $ Fv          : Factor w/ 7 levels "-","0","1","2",..: 3 2 2 2 2 2 2 2 2 2 ...
##  $ Ag          : Factor w/ 5 levels "-","0","1","2",..: 2 2 2 2 2 2 4 2 3 2 ...
##  $ Cm          : Factor w/ 7 levels "-","0","1","2",..: 3 4 7 4 4 2 4 3 7 3 ...
##  $ Rv          : Factor w/ 13 levels "-","0","1","10",..: 9 3 6 3 2 2 7 2 9 6 ...
##  $ PIR         : Factor w/ 57 levels "-","-1","-10",..: 24 55 18 36 2 13 16 56 28 36 ...
```

En el segundo dataset vemos que la mayoría de variables se han interpretado como tipo `factor` debido a que muchas contenían elementos vacíos. Como ya hemos solucionado este problema, vamos a proceder a asignar el tipo de dato correspondiente a cada variable:

```
# Variables cuantitativas discretas
data_euroleague$PlayerNumber <- as.integer(as.character(data_euroleague$PlayerNumber))
data_euroleague$Pts <- as.integer(as.character(data_euroleague$Pts))
data_euroleague$O <- as.integer(as.character(data_euroleague$O))
data_euroleague$D <- as.integer(as.character(data_euroleague$D))
data_euroleague$T <- as.integer(as.character(data_euroleague$T))
data_euroleague$As <- as.integer(as.character(data_euroleague$As))
data_euroleague$St <- as.integer(as.character(data_euroleague$St))
data_euroleague$To <- as.integer(as.character(data_euroleague$To))
data_euroleague$Fv <- as.integer(as.character(data_euroleague$Fv))
data_euroleague$Ag <- as.integer(as.character(data_euroleague$Ag))
data_euroleague$Cm <- as.integer(as.character(data_euroleague$Cm))
data_euroleague$Rv <- as.integer(as.character(data_euroleague$Rv))
data_euroleague$PIR <- as.integer(as.character(data_euroleague$PIR))
```

Verificamos que se han realizado los cambios de tipos de variable:

```
str(data_euroleague)
```

```
## 'data.frame':    6505 obs. of  20 variables:
##  $ MatchId     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Team        : Factor w/ 18 levels "ALBA Berlin",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ PlayerNumber: int  1 5 6 8 9 10 11 12 13 24 ...
##  $ PlayerName  : Factor w/ 306 levels "ABALDE, ALBERTO",..: 250 28 276 303 288 67 127 184 97 132 ...
##  $ Min         : Factor w/ 1910 levels "0","0:01","0:02",..: 1345 626 1502 99 1645 1 976 617 924 486
##  $ Pts         : int  22 4 17 0 0 0 14 6 18 0 ...
##  $ X2FG        : Factor w/ 92 levels "-","0","0/1",..: 49 14 13 2 2 2 32 2 31 2 ...
##  $ X3FG        : Factor w/ 74 levels "-","0","0/1",..: 22 2 58 2 2 2 28 25 38 4 ...
##  $ FT          : Factor w/ 66 levels "-","0","0/1",..: 58 22 2 2 2 2 38 2 42 2 ...
##  $ O           : int  0 3 1 1 0 0 1 0 3 1 ...
##  $ D           : int  2 3 3 1 0 0 6 2 3 0 ...
##  $ T           : int  2 6 4 2 0 0 7 2 6 1 ...
##  $ As          : int  6 0 1 2 1 0 0 0 4 4 ...
##  $ St          : int  0 2 4 1 0 0 2 1 0 0 ...
##  $ To          : int  3 1 1 1 0 0 1 0 0 1 ...
##  $ Fv          : int  1 0 0 0 0 0 0 0 0 0 ...
##  $ Ag          : int  0 0 0 0 0 0 2 0 1 0 ...
##  $ Cm          : int  1 2 5 2 2 0 2 1 5 1 ...
##  $ Rv          : int  5 1 2 1 0 0 3 0 5 2 ...
##  $ PIR         : int  19 7 13 3 -1 0 11 8 22 3 ...
```

Hemos conseguido limpiar bastante los datos pero aún debemos realizar algunas modificaciones en algunas variables:

- **Min:** la variable minutos la pasaremos a formato decimal.
- **Tiros:** las variables tiros de dos (`X2FG`), tiros de tres (`X3FG`) y tiros libres (`FT`), se encuentran en formato "M/A" donde M es convertidos (*made*) y A intentados (*attempts*). Por cada una de ellas, crearemos por lo tanto dos variables, separando los aciertos de los intentos.

```
# Modificamos la variable tiempo
levels(data_euroleague$Min) <- c(levels(data_euroleague$Min), "0:00")
data_euroleague$Min[data_euroleague$Min=="0"] <- "0:00"

data_euroleague$Min <- sapply(strsplit(as.character(data_euroleague$Min),":"),
  function(x) {
    x <- as.numeric(x)
    x[1]+x[2]/60
  })
```

```
# Modificamos la variables tiros de 2
levels(data_euroleague$X2FG) <- c(levels(data_euroleague$X2FG), "0/0")
data_euroleague$X2FG[data_euroleague$X2FG=="0"] <- "0/0"

data_euroleague$x2M <- sapply(strsplit(as.character(data_euroleague$X2FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$x2A <- sapply(strsplit(as.character(data_euroleague$X2FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$X2FG <- NULL
```

```
# Modificamos la variables tiros de 3
levels(data_euroleague$X3FG) <- c(levels(data_euroleague$X3FG), "0/0")
data_euroleague$X3FG[data_euroleague$X3FG=="0"] <- "0/0"

data_euroleague$x3M <- sapply(strsplit(as.character(data_euroleague$X3FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$x3A <- sapply(strsplit(as.character(data_euroleague$X3FG),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$X3FG <- NULL
```

```r
# Modificamos la variables tiros libres
levels(data_euroleague$FT) <- c(levels(data_euroleague$FT), "0/0")
data_euroleague$FT[data_euroleague$FT=="0"] <- "0/0"

data_euroleague$FTM <- sapply(strsplit(as.character(data_euroleague$FT),"/"),
  function(x) {
    x <- as.integer(x)
    x[1]
  })

data_euroleague$FTA <- sapply(strsplit(as.character(data_euroleague$FT),"/"),
  function(x) {
    x <- as.integer(x)
    x[2]
  })

data_euroleague$FT <- NULL
```

Realizamos la última comprobación de los tipos de datos de las variables del dataset `data_euroleague`:

```r
sapply(data_euroleague, function(x) class(x))
```

```
##      MatchId         Team PlayerNumber   PlayerName          Min          Pts
##    "integer"     "factor"    "integer"     "factor"    "numeric"    "integer"
##            0            D            T           As           St           To
##    "integer"    "integer"    "integer"    "integer"    "integer"    "integer"
##           Fv           Ag           Cm           Rv          PIR          x2M
##    "integer"    "integer"    "integer"    "integer"    "integer"    "integer"
##          x2A          x3M          x3A          FTM          FTA
##    "integer"    "integer"    "integer"    "integer"    "integer"
```

Por tanto, el resumen es el siguiente:

- Variables cualitativas nominales: `Team`, `PlayerName` (tipo `factor`).
- Variables cuantitativas discretas: `MatchId`, `PlayerNumber`, `Pts`, `O`, `D`, `T`, `As`, `St`, `To`, `Fv`, `Ag`, `Cm`, `Rv`, `PIR`, `x2M`, `x2A`, `x3M`, `x3A`, `FTM`, `FTA` (tipo `integer`).
- Variables cuantitativas continuas: `Min` (tipo `numeric`).

3.3 Agregación de datos

Como previamente comentamos en el apartado de integración y selección de datos de interés, la idea principal es crear un nuevo dataset que agrupe las estadísticas para cada equipo y partido. Por tanto, el resultado será la suma de las estadísticas de todos los jugadores del equipo para cada partido. Este nuevo dataset se llamará `data_team`:

```
data_team <- as.data.frame(data_euroleague %>% group_by(Team, MatchId) %>%
  select_if(is.numeric) %>%
  summarise_each(list(sum)))

data_team[c("PlayerNumber")] <- NULL
```

Mostramos los primeros registros de este nuevo dataset:

```
head(data_team)
```

```
##          Team MatchId Min Pts  O  D  T As St To Fv Ag Cm Rv PIR x2M x2A x3M x3A
## 1 ALBA Berlin       7 200  85 11 31 42 23  5 10  4  4 19 17 106  19  40  13  29
## 2 ALBA Berlin      16 225 105 14 27 41 28  4  7  2  3 21 19 127  25  48  14  30
## 3 ALBA Berlin      26 200  84  6 18 24 21  9 17  2  4 27 24  75   8  24  14  35
## 4 ALBA Berlin      34 200  66 14 19 33 20 10 15  0  1 21 22  64  16  40   7  24
## 5 ALBA Berlin      39 200  78 13 21 34 15 10 13  2  2 20 21  81   9  28  14  37
## 6 ALBA Berlin      54 200  71 11 23 34 14  8 20  2  4 25 17  56  20  38   8  26
##   FTM FTA
## 1   8   8
## 2  13  15
## 3  26  30
## 4  13  22
## 5  18  20
## 6   7  12
```

A primera vista este dataset se ha creado correctamente. Como prueba de la calidad de los datos y del proceso de transformación podemos comprobar que los minutos de los partidos son correctos:

```
5 jugadores * 10 mintuos * 4 cuartos = 200 minutos
```

Para los partidos con prórroga se deben sumar 5 minutos por cada parte de tiempo extra y multiplicar por los 5 jugadores. Por eso vemos partidos con 225 minutos.

3.4 Identificación y tratamiento de valores extremos

Los valores extremos (*extreme scores* o *outliers*) son aquellos datos que se encuentran muy alejados de la distribución normal de una variable o población. Generalmente se considera que cuando un valor se encuentra alejado 3 desviaciones estándar con respecto a la media del conjunto es un *outlier*. Para encontrar estos valores extremos se utiliza los diagramas de cajas o *boxplot*, que permiten identificar de manera muy sencilla si existen valores extremos para una determinada variable.

Para detectar los valores extremos de nuestro dataset de estadísticas de equipo, vamos a crear un diagrama de caja por cada variable cuantitativa:

4 Enriquecimiento de los datos

4.1 Datos del oponente.

Durante un encuentro de baloncesto, es tan importante medir lo que tu equipo ha hecho como lo que tu equipo ha recibido. Por lo tanto, buscamos enriquecer la información por partido de cada equipo añadiendo la información del oponente. Para ello, utilizaremos la tabla de scores para completar así los registros.

```
names(data_scoreboards)[names(data_scoreboards) == "Id"] <- "MatchId"
names(data_scoreboards)[names(data_scoreboards) == "HomeTeam"] <- "Team"
names(data_scoreboards)[names(data_scoreboards) == "VisitingTeam"] <- "Opponent"


df_1 <- merge(data_team, data_scoreboards, by=c("MatchId","Team"))
df_1["Local"] <- 1


names(data_scoreboards)[names(data_scoreboards) == "Team"] <- "HomeTeam"
names(data_scoreboards)[names(data_scoreboards) == "Opponent"] <- "Team"
names(data_scoreboards)[names(data_scoreboards) == "HomeTeam"] <- "Opponent"
df_2 <- merge(data_team, data_scoreboards, by=c("MatchId","Team"))
df_2["Local"] <- 0

data_team <- bind_rows(df_1,df_2)

data_team[c("Date")] <- NULL
data_team[c("HomeScore")] <- NULL
data_team[c("VisitingScore")] <- NULL
data_team[c("Link")] <- NULL

data_team2 <- data_team [c("MatchId","Team","Min","Pts","O","D","T","As","St","To","Fv","Ag","Cm","Rv",

names(data_team2)[names(data_team2) == "Pts"] <- "Pts_Opp"
names(data_team2)[names(data_team2) == "O"] <- "O_Opp"
names(data_team2)[names(data_team2) == "D"] <- "D_Opp"
names(data_team2)[names(data_team2) == "T"] <- "T_Opp"
names(data_team2)[names(data_team2) == "As"] <- "As_Opp"
names(data_team2)[names(data_team2) == "St"] <- "St_Opp"
names(data_team2)[names(data_team2) == "To"] <- "To_Opp"
names(data_team2)[names(data_team2) == "Fv"] <- "Fv_Opp"
names(data_team2)[names(data_team2) == "Ag"] <- "Ag_Opp"
names(data_team2)[names(data_team2) == "Cm"] <- "Cm_Opp"
names(data_team2)[names(data_team2) == "Rv"] <- "Rv_Opp"
names(data_team2)[names(data_team2) == "PIR"] <- "PIR_Opp"
names(data_team2)[names(data_team2) == "x2M"] <- "x2M_Opp"
names(data_team2)[names(data_team2) == "x2A"] <- "x2A_Opp"
names(data_team2)[names(data_team2) == "x3M"] <- "x3M_Opp"
names(data_team2)[names(data_team2) == "x3A"] <- "x3A_Opp"
names(data_team2)[names(data_team2) == "FTM"] <- "FTM_Opp"
names(data_team2)[names(data_team2) == "FTA"] <- "FTA_Opp"

data_teamopp <- merge(data_team, data_team2, by.x=c("MatchId","Opponent"),  by.y=c("MatchId","Team"))
data_teamopp["y.min"]<-NULL
```

4.2 Creación de nuevas métricas

Durante los últimos años, se ha dado una vuelta al análisis de la estadísticas en baloncesto. Como los diferentes equipos, de acuerdo a su estilo, juegan a distintos ritmos, no podemos usar los promedios por partido para compararlos. Para poder lograr las comparaciones hay definir el concepto de las posesiones, que es la base de estos cálculos. El basket es un juego en el que ambos equipos se alternan la posesión de la pelota. El equipo que aproveche mejor sus posesiones será el equipo ganador. Se entiende que una posesión termina con un tiro al aro, una pérdida de balón o un tiro libre. Allí el balón pasa al rival y la posesión se termina. ¿Qué ocurre si el equipo falla el tiro de campo pero captura el rebote ofensivo? Hoy por hoy, la mayoría de los estadistas de baloncesto consideran que no se le debe anotar una nueva posesión al equipo, sino considerar que la misma posesión continúa.

Para tener en cuenta lo anteriormente mencionado, definimos las métricas a continuación

4.2.1 Pace o Ritmo

Nos da una idea del ritmo de juego del equipo, expresado en cantidad de posesiones por juego que utiliza. Hay equipos que corren más y equipos que prefieren el juego estático. Por eso las estadísticas por juego NO sirven para comparar equipos. Las posesiones se calculan con la siguiente fórmula: $Pos = FGA \check{} OR + TO + (FTA * 0.4)$ donde - Pos: posesiones - FGA (field goal attempts): lanzamientos de campo (tanto de 2 y de 3) - OR (ofensive rebounds): rebotes ofensivos - TO (turnovers): pelotas perdidas - FTA (free throw attempts): tiros libres lanzados

```
data_teamopp ["Poss"] <- +data_teamopp["x2A"] +data_teamopp["x3A"] -data_teamopp["O"] +data_teamopp["To
```

4.2.2 Rating ofensivo y defensivo

Evaluamos el aspecto ofensivo, defensivo y la diferencia entre ambos.

4.2.2.1 Eficiencia ofensiva (Rating ofensivo)

Habitualmente se evalúa la ofensiva en puntos convertidos por juego, lo cual es una manera un tanto absurda. Si pensamos el juego como una serie de posesiones, el equipo que más puntos convierta en sus posesiones, será el más efectivo. Se multiplica por 100 para expresar los puntos cada 100 posesiones, y no manejar números con decimales. Así:

$OffensiveRating = (puntos/posesiones) * 100$

4.2.2.2 Eficiencia defensiva (Rating defensivo)

Así como medimos la eficiencia ofensiva en base a puntos convertidos cada 100 posesiones, podemos medir la defensa en base a puntos recibidos (o puntos del oponente) cada 100 posesiones del equipo contrario.

$DefensiveRating = (puntos_{oponente}/posesiones) * 100$

```
data_teamopp ["Off_Rat"] <- +data_teamopp["Pts"]/data_teamopp["Poss"]

data_teamopp ["Def_Rat"] <- +data_teamopp["Pts_Opp"]/data_teamopp["Poss"]
```

4.2.3 Porcentaje efectivo de tiros de campo (eFG%)

Esta estadística ajusta los tiros de campo dandole el valor extra (un punto más) a los triples. Esto corrige el FG% común que subestima a los triples. Por ejemplo, si un jugador ha hecho 2/5 en T2 y 1/4 en T3, habrá convertido 3/9 en tiros de campo (33%), que es similar a si hubiera metido 3/5 de T2 y 0/4 en T3. Sin embargo, en el primer supuesto ha conseguido más puntos para el equipo. Así, el eFG% del primero será 44.4% que se ajusta más a la realidad. La fórmula, por tanto es:

$eFG\% = (FGM + 0.5 * 3PM)/FGA$

FGM (field goal made): tiros de campo convertidos 3PM (3 points made): triples convertidos FGA (field goal attempts): tiros de campo intentados

```
data_teamopp ["eFG"] <- (data_teamopp["x2M"]+1.5*data_teamopp["x3M"])/ (data_teamopp["x2A"]+data_teamopp

data_teamopp ["eFG_Opp"] <- (data_teamopp["x2M_Opp"]+1.5*data_teamopp["x3M_Opp"])/ (data_teamopp["x2A_Op
```

4.2.4 True Shooting (TS) o Lanzamientos reales

Esta métrica tiene en cuenta los dobles, triples y tiros libres, para dar una idea de cómo tira el jugador globalmente. Ejemplo: en la 2009-2010, Martin Leiva quedó #8 en FG% con 58,72. Pero si le agregamos los libres, cae al puesto #91 con 54.8%. La formula es:

$TS = puntos/(2 * (FGA + 0.44 * FTA)$

- FGA (field goal attempts): lanzamientos de cancha intentados
- FTA (free throw attempts): tiros libres intentados

```
data_teamopp ["TS"] <- data_teamopp["Pts"] /  (data_teamopp["x2A"]+data_teamopp["x3A"]+0.44*data_teamopp

data_teamopp ["TS_Opp"] <- data_teamopp["Pts_Opp"] /  (data_teamopp["x2A_Opp"]+data_teamopp["x3A_Opp"]+0
```

4.2.5 Rebotes

Los rebotes totales de un equipo son de poco valor. Capturar un rebote ofensivo requiere diferentes habilidades que capturar uno defensivo, por lo que deben analizarse por separado. Tener en cuenta el número absoluto de rebotes conseguidos, o el promedio de rebotes por partido, nos puede llevar a errores, ya que los rebotes ´disponibles´ dependen de la efectividad: si un equipo falla poco, hay pocos rebotes por tomar. Ejemplo: el equipo A tomó en un partido 20 rebotes defensivos. Si el equipo B falló 30 lanzamientos (o sea que hubo 30 rebotes en el aro defensivo de A) entonces A capturó 66% de los rebotes en su aro (20 de 30). Pero si B erró 25 tiros, A tomó 80% de los rebotes (20 de 25). Esto hace que, para evaluarlo correctamente, definamos: DR% como porcentaje de rebotes defensivos y OR% porcentaje de rebotes ofensivos.

4.2.5.1 Tasa de rebotes ofensivos

$\%derebotesofensivos = [OR/(OR + opDR)] * 100$

- OR (ofensive rebounds): rebotes ofensivos
- Op DR (oponent defensive rebounds): rebotes defensivos del rival

4.2.5.2 Tasa de rebotes defensivos

$\%derebotesdefensivos = [DR/(DR + opOR)] * 100$

- DR (defensive rebounds): rebotes defensivos
- Op OR (oponent ofensive rebounds): rebotes ofensivos del rival

```
data_teamopp ["Off_Reb"] <- data_teamopp["O"] /  (data_teamopp["O"]+data_teamopp["D_Opp"])

data_teamopp ["Def_Reb"] <- data_teamopp["D"] /  (data_teamopp["D"]+data_teamopp["O_Opp"])
```

4.2.6 Porcentaje de asistencias y pérdidas

Al igual que con los rebotes y otras estadísitcas, las asistencias por juego no son un buen parámetro, ya que dependen del ritmo de juego. Más preciso es calcular las asistencias expresadas en posesiones terminan con una pérdida de balón. Habitualmente expresado en porcentaje.

4.2.7 Porcentaje de asistencias

Se calculan mediante la siguiente fórmula:

$\%deasistencias = (asistencias/posesiones) * 100$

```
data_teamopp ["Pct_ass"] <- data_teamopp["As"] /  (data_teamopp["Poss"])

data_teamopp ["Pct_ass_opp"] <- data_teamopp["As_Opp"] /  (data_teamopp["Poss"])
```

4.2.7.1 Porcentaje de pérdidas

Lo mismo ocurre con las pérdidas. No es lo mismo perder 10 pelotas en un partido en que hubo 100 posesiones, que en uno que hubo 80. Por eso es mejor calcular las pérdidas cada 100 posesiones. El valor ideal depende del ritmo de juego, pero podríamos decir que el objetivo sería tener menos de 15% de TO y provocar en el oponente más de 15%. La fórmula es: $\%depérdidas = (pelotaspérdidas/posesiones) * 100$

```
data_teamopp ["Pct_To"] <- data_teamopp["To"] /  (data_teamopp["Poss"])

data_teamopp ["Pct_To_opp"] <- data_teamopp["To_Opp"] /  (data_teamopp["Poss"])
```

4.2.7.2 Tiros libres, respecto a tiros de campo (FTM/FGA)

Es simplemente una manera de expresar el número de veces que un equipo va a la línea y cuántas veces envía al oponente a la línea. Esta considerado (junto con el effective field goal percentage, la tasa de rebotes ofensivos y la tasa de pérdidas) uno de los cuatro factores con los que se miden los partidos. La fórmula es: $ Libres por lanzamientos de cancha = (libres convertidos/tiros de cancha intentados) * 100$

```
data_teamopp["FTR"] <- data_teamopp["FTM"] /  (data_teamopp["x2A"]+ data_teamopp["x3A"])

data_teamopp["FTR_opp"] <- data_teamopp["FTM_Opp"] /  (data_teamopp["x2A_Opp"]+ data_teamopp["x3A_Opp"]
```

4.2.8 Expected wins (triunfos esperados)

Para ganar, obviamente hay que meter más puntos que el rival. Existe un cálculo (comprobado en la NBA, baloncesto FIBA y universitario) que de acuerdo a los puntos convertidos y recibidos, se puede estimar la cantidad de partidos que habría que haber ganado. Suele correlacionar muy bien con la realidad.

$Triunfosesperados = eficienciaofensiva^14/(eficienciaofensiva^14 + eficienciadefensiva^14)$

```
data_teamopp ["Expected_Wins"] <- data_teamopp["Off_Rat"]^14 /  (data_teamopp["Off_Rat"]^14+data_teamopp
```

---

5 Análisis de los datos

FALTA POR HACER

5.1 Selección de los grupos de datos que se quieren analizar/comparar

5.2 Comprobación de la normalidad y homogeneidad de la varianza

5.3 Pruebas estadísticas para comparar los grupos de datos

5.3.1 Regresión lineal para estimar los ratings

```
#data_teamopp["Off_Reb_Opp"]<-1-data_teamopp["Def_Reb"]
```

```
#library(data.table)
#data_ataque <- data_teamopp[c("Off_Rat","eFG","Off_Reb","Pct_To","FTR")]

#corr_ataque <- round(cor(data_ataque),2)

#library(ggcorrplot)
#ggcorrplot(corr_ataque, type = "upper",
#   lab = TRUE)

#model_offesive_rating<- lm(Off_Rat~., data=data_ataque)
#summary(model_offesive_rating)


#data_defensa <- data_teamopp[c("Def_Rat","eFG_Opp","Off_Reb_Opp","Pct_To_opp","FTR_opp")]
#corr_defensa <- round(cor(data_defensa),2)

#ggcorrplot(corr_defensa, type = "upper",
#   lab = TRUE)

#model_defensive_rating<- lm(Def_Rat~., data=data_defensa)
#summary(model_defensive_rating)
```

5.3.2 Regresión logística para estimar victoria

Buscamos estimar la probabibilidad de victoria en función de los estadísticos creados.

```
#datos_logit<-data_teamopp[c("eFG","Off_Reb","Pct_To","FTR","eFG_Opp","Off_Reb_Opp","Pct_To_opp","FTR_o

#model_victoria <- glm(formula=Win ~ . , data=datos_logit, family=binomial)
#summary(model_victoria)
```

```
#library(pROC)
#rocobj <- roc( datos_logit$Win, predict(model_victoria, datos_logit), auc = TRUE, ci = TRUE )
#print(rocobj)

#plot.roc( rocobj, legacy.axes = TRUE, print.thres = "best", print.auc = TRUE, auc.polygon = FALSE, max
```

5.3.3 Clustering de jugadores

---

6 Exportación de datos finales

A continuación vamos a exportar nuestros dataframes finales a un archivo csv. Estos archivos se llamarán euroleague_scoreboards_clean.csv, euroleague_stats_per_game_clean.csv y euroleague_stats_per_team_clean.cs
Utilizamos la función write.csv2() para exportar el fichero en formato csv español:

```
write.csv2(data_scoreboards, row.names = FALSE,
           file = "../csv/euroleague_scoreboards_clean.csv")

write.csv2(data_euroleague, row.names = FALSE,
           file = "../csv/euroleague_stats_per_game_clean.csv")

write.csv2(data_team, row.names = FALSE,
```

```
            file = "../csv/euroleague_stats_per_team_clean.csv")
```

Estos nuevos datasets también estarán disponibles en el repositorio de GitHub mencionado en el primer apartado de este documento.

---

7 Representación de los resultados a partir de tablas y gráficas

FALTA POR HACER

---

8 Resolución del problema

FALTA POR HACER

---

9 Contribuciones al trabajo

Contribuciones	**Firma**
Selección del dataset	MSP, AAG
Creación del repositorio GitHub	MSP, AAG
Desarrollo código en R	MSP, AAG
Redacción de las respuestas	MSP, AAG