Informe Final

Oscar

2023-09-14

CARGA Y LIMPIEZA DE DATOS

Se realiza la carga de los conjuntos de datos necesarios para el análisis así como su unión en un mismo Data Frame. Se llevan a cabo los siguientes pasos: *Carga de las bases de datos por cada liga, correspondientes a la última temporada.* Unión de estos Data Frame en uno solo. *Separar la columna Posición en varias columnas y eliminar a partir de la tercera, puesto que no interesa.* Crear nuevas variables para datos por 90 minutos.

```
df = rbind(players_bundesliga_22_23,players_epl_22_23,players_la_liga_22_23,players_ligue_1_22_23,playe
#View(df)
summary(df)
```

```
##        id        player_name            games            time
##  Min.   :    3   Length:2806        Min.   : 1.00   Min.   :   1.0
##  1st Qu.: 3224   Class :character   1st Qu.: 9.00   1st Qu.: 340.2
##  Median : 7196   Mode  :character   Median :22.00   Median :1164.0
##  Mean   : 6543                      Mean   :19.99   Mean   :1285.9
##  3rd Qu.: 9809                      3rd Qu.:31.00   3rd Qu.:2098.8
##  Max.   :11627                      Max.   :38.00   Max.   :3420.0
##      goals             xG              assists            xA
##  Min.   : 0.00   Min.   : 0.00000   Min.   : 0.00   Min.   : 0.00000
##  1st Qu.: 0.00   1st Qu.: 0.07424   1st Qu.: 0.00   1st Qu.: 0.06675
##  Median : 0.00   Median : 0.73405   Median : 0.00   Median : 0.63351
##  Mean   : 1.74   Mean   : 1.89416   Mean   : 1.21   Mean   : 1.33929
##  3rd Qu.: 2.00   3rd Qu.: 2.25419   3rd Qu.: 2.00   3rd Qu.: 1.83024
##  Max.   :36.00   Max.   :32.76140   Max.   :16.00   Max.   :17.26671
##      shots          key_passes      yellow_cards      red_cards
##  Min.   :  0.00   Min.   :  0.00   Min.   : 0.000   Min.   :0.000
##  1st Qu.:  2.00   1st Qu.:  1.00   1st Qu.: 0.000   1st Qu.:0.000
##  Median :  9.00   Median :  7.00   Median : 2.000   Median :0.000
##  Mean   : 16.28   Mean   : 12.08   Mean   : 2.578   Mean   :0.129
##  3rd Qu.: 23.00   3rd Qu.: 18.00   3rd Qu.: 4.000   3rd Qu.:0.000
##  Max.   :150.00   Max.   :118.00   Max.   :14.000   Max.   :3.000
##    position          team_title             npg              npxG
##  Length:2806        Length:2806        Min.   : 0.000   Min.   : 0.00000
##  Class :character   Class :character   1st Qu.: 0.000   1st Qu.: 0.07424
##  Mode  :character   Mode  :character   Median : 0.000   Median : 0.71628
##                                        Mean   : 1.583   Mean   : 1.73964
##                                        3rd Qu.: 2.000   3rd Qu.: 2.12457
##                                        Max.   :29.000   Max.   :27.43321
##     xGChain          xGBuildup
##  Min.   : 0.000   Min.   : 0.0000
```

```
##  1st Qu.: 1.034   1st Qu.: 0.5527
##  Median : 3.686   Median : 2.1301
##  Mean   : 5.287   Mean   : 3.1783
##  3rd Qu.: 7.791   3rd Qu.: 4.5664
##  Max.   :39.003   Max.   :24.7041
```

```
library(dplyr)
library(tidyr)
library(stringr)

df1 = df %>%
  separate(position, c('position1', 'position2','position3')) %>%
  cbind(goals90=df$goals/df$time*90) %>%
  cbind(xG90=df$xG/df$time*90) %>%
  cbind(assists90=df$assists/df$time*90) %>%
  cbind(xA90=df$xA/df$time*90) %>%
  cbind(shots90=df$shots/df$time*90) %>%
  cbind(key_passes90=df$key_passes/df$time*90) %>%
  cbind(npg90=df$npg/df$time*90) %>%
  cbind(npxG90=df$npxG/df$time*90) %>%
  cbind(xGChain90=df$xGChain/df$time*90) %>%
  cbind(xGBuildup90=df$xGBuildup/df$time*90)
```

Respecto al Data Frame de los tiros realizados en estas ligas durante la temporada 2022-2023, se realizará la limpieza de datos para luego utilizarlo luego.

Para la limpieza de los tiros, se eliminan las columnas erróneas que han debido crearse al realizar el web scrapping y los menos recientes, quedándonos con los tiros de las últimas 3 temporadas.

```
shots_dataset <- read_csv("shots_dataset.csv")
```

```
## Rows: 435443 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (9): result, player, h_a, situation, shotType, h_team, a_team, player_...
## dbl  (10): id, minute, X, Y, xG, player_id, season, match_id, h_goals, a_goals
## dttm  (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
shots_dataset = shots_dataset[shots_dataset$h_a !='h_a', ]
shots_dataset = shots_dataset[shots_dataset$season >=2022, ]
```

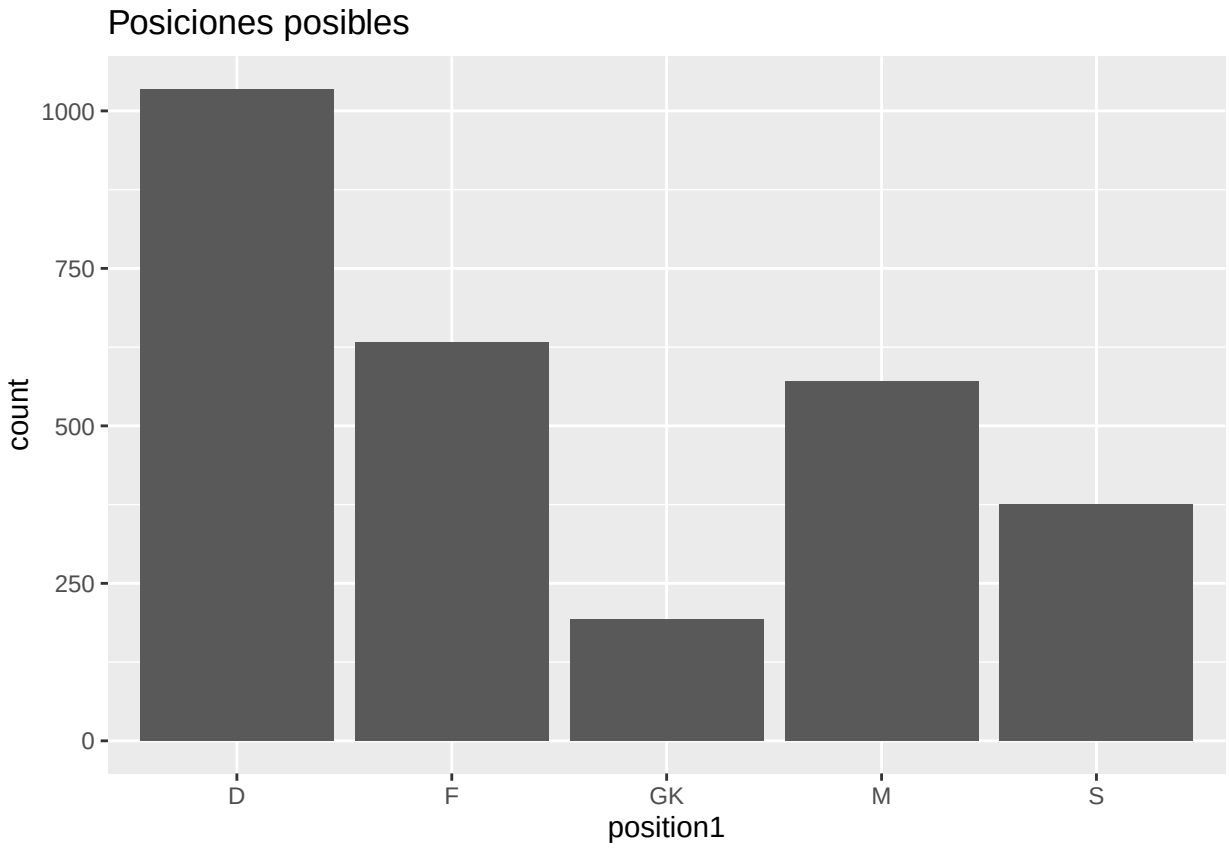
Con esto, ha finalizado la limpieza de los datos.

ANÁLISIS DE DATOS

El objetivo de este estudio es proporcionar varios jugadores como posibles sustitutos de los jugadores del Atlético de Madrid, y se realizará por posición. Por tanto, se estudiarán las métricas de los jugadores titulares del equipo y se buscarán jugadores similares de otros equipos.

El estudio se realizará por posiciones de juego. Vamos a ver cuántas posiciones hay y a qué equivalen esas siglas.

```
ggplot(df1, aes(position1)) +
  geom_bar() +
  labs(title="Posiciones posibles")
```



Se ve de forma clara a qué corresponde cada sigla: * D = Defender - Defensa * F = Forward - Atacante * GK = Goalkeeper - Portero * M = Midfielder - Centrocampista * s = Striker - Delantero

DELANTEROS

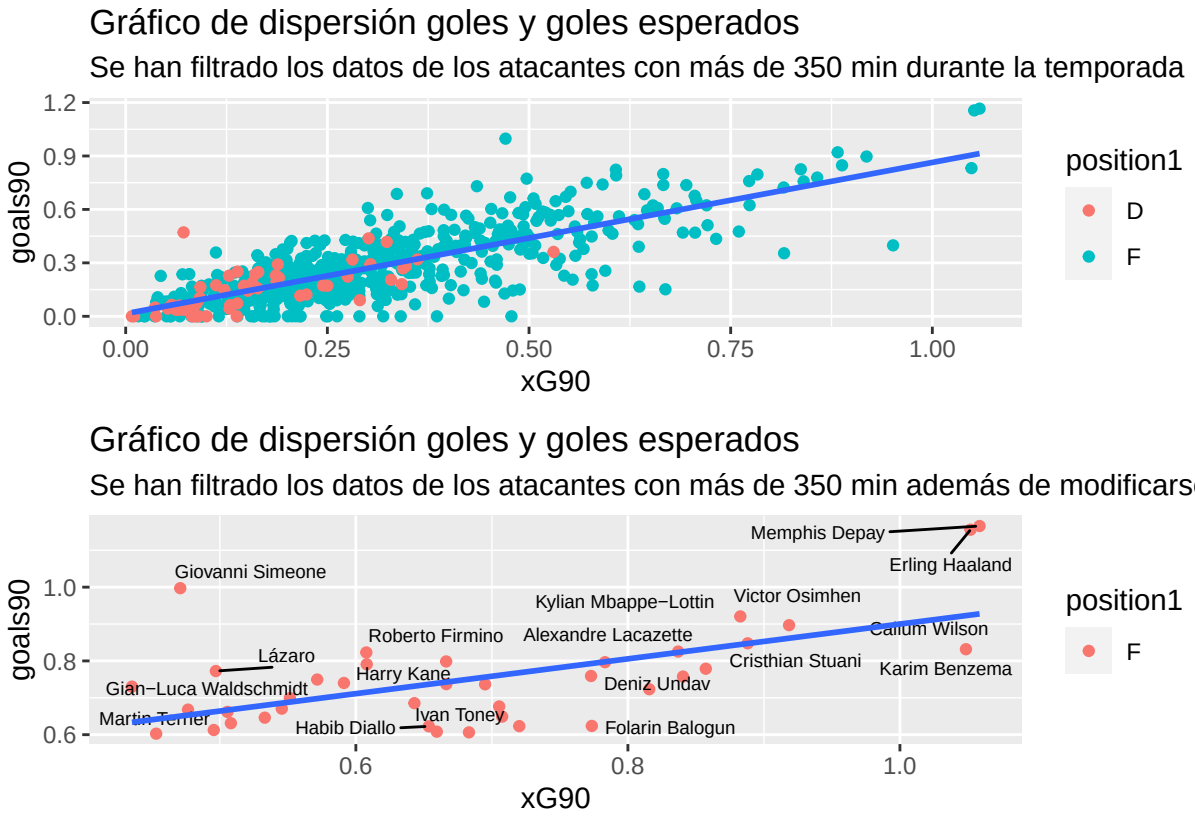
Para la posición de delantero, lo principal es que marquen más goles de lo que "deben", es decir, que la diferencia entre goals - xG sea positiva. Además, que sean capaces de marcar más goles/90 min es interesante. Por tanto, se crearán gráficos donde observar esta variable y se aprovechará para crear nuevas variables para datos por 90 minutos.

```
forward <- rbind(df1[df1$position1 == "F",], df1[df1$position2 == "F",], df1[df1$position1 == "S",])
forward <- forward %>% filter(!is.na(forward$player_name))
forward <- distinct(forward)
```

```
library("ggplot2")
library('patchwork')
library("ggrepel")

forw <- forward %>%
  filter(goals90 > 0.6 & xG90 >0.4 & time > 350)
```

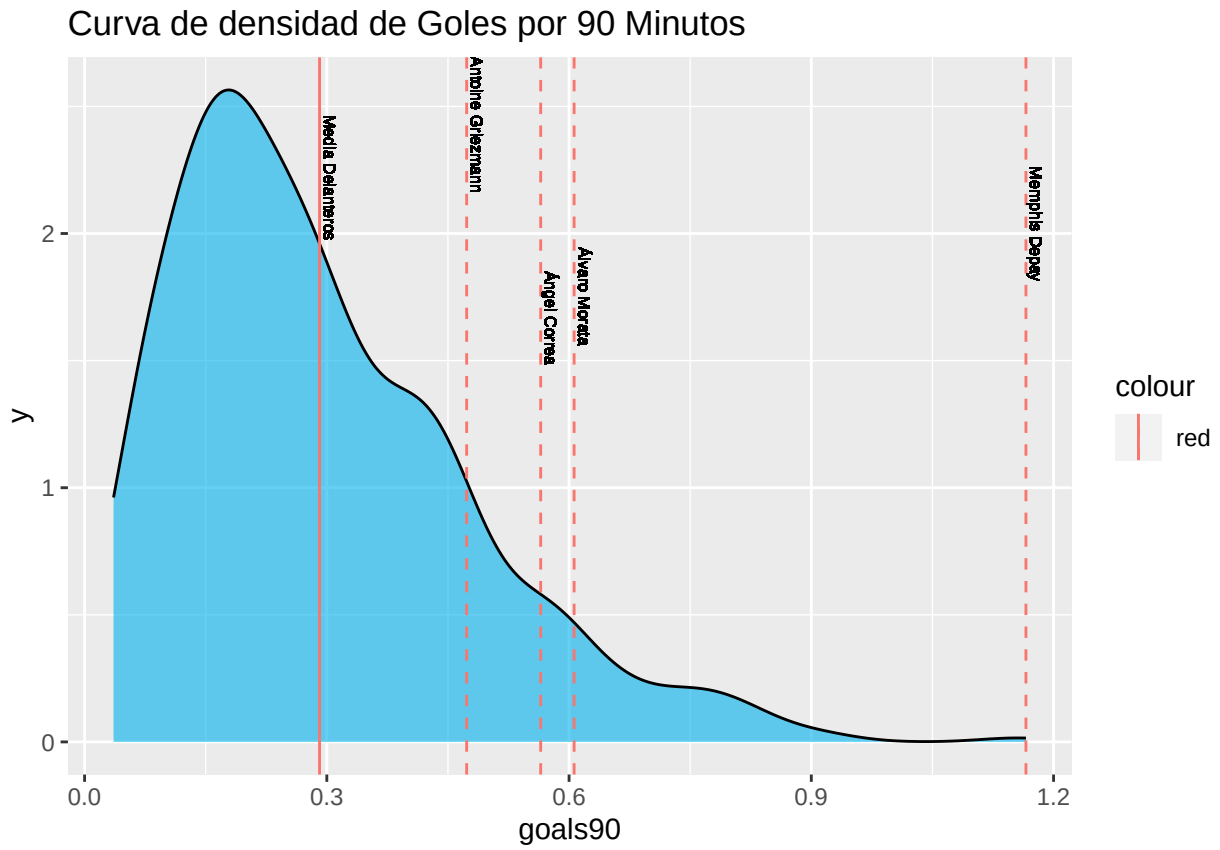
```r
#Crea una nueva variable de diferencia entre goles anotados y esperados y ordena por orden decreciente.
delantero <- forward %>%
  cbind(dif_xG = forward$goals90 - forward$xG90)
delantero <- delantero[order(delantero$dif_xG, decreasing = TRUE), ]

p1 <- ggplot(data = forward[forward$time > 350,], aes(x = xG90, y = goals90)) +
  geom_point(aes(colour = position1)) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title ="Gráfico de dispersión goles y goles esperados", subtitle = "Se han filtrado los datos de

p2 <- ggplot(forw, aes(x = xG90, y = goals90,label =player_name )) +
  geom_point(aes(colour = position1)) +
  geom_text_repel(aes(label = player_name), size=2.3) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title ="Gráfico de dispersión goles y goles esperados", subtitle = "Se han filtrado los datos de

p1 / p2
```

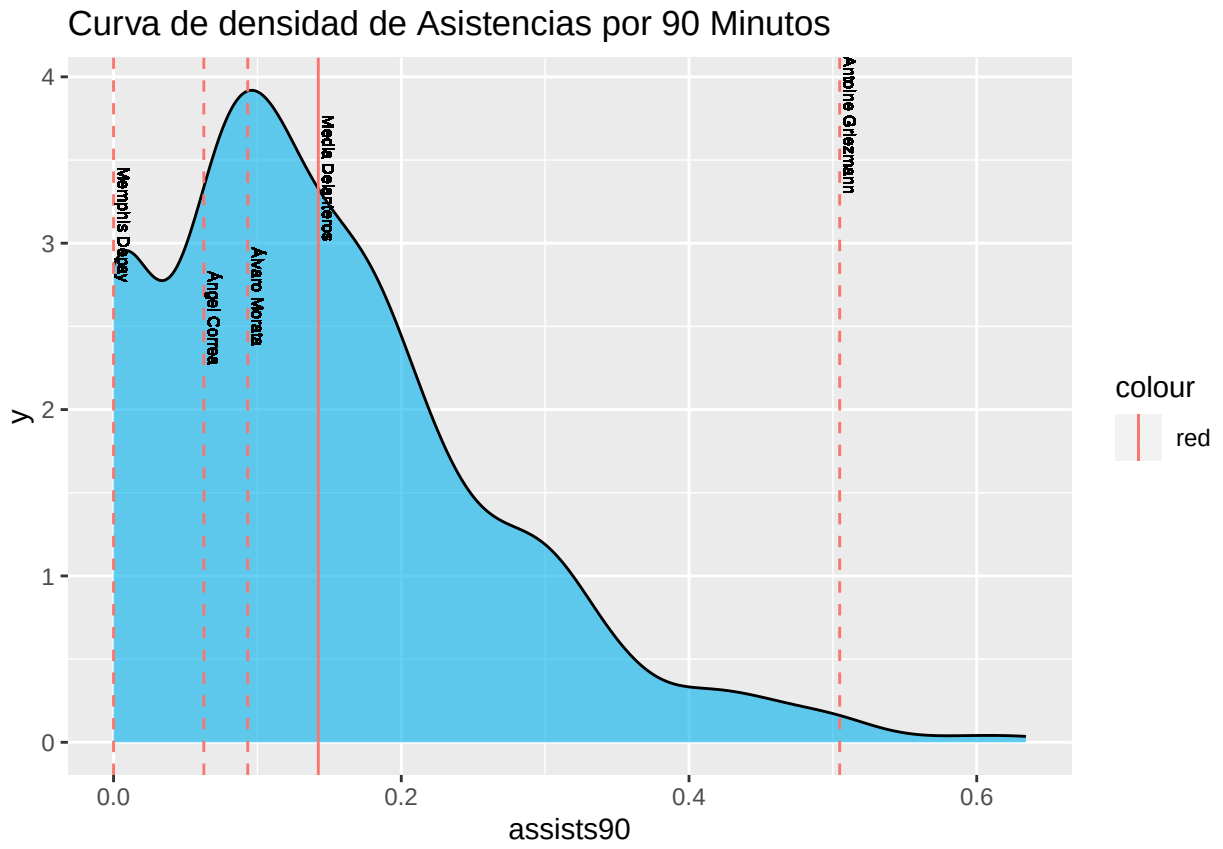
```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



Se representan las variables principales de un delantero en forma de función de densidad. El código será el mismo para todas los gráficos, solo se mostrará el primero. Gracias a la representación en rojo de los 4 delanteros del equipo, luego se filtrará la totalidad de jugadores con valores similares a estos.

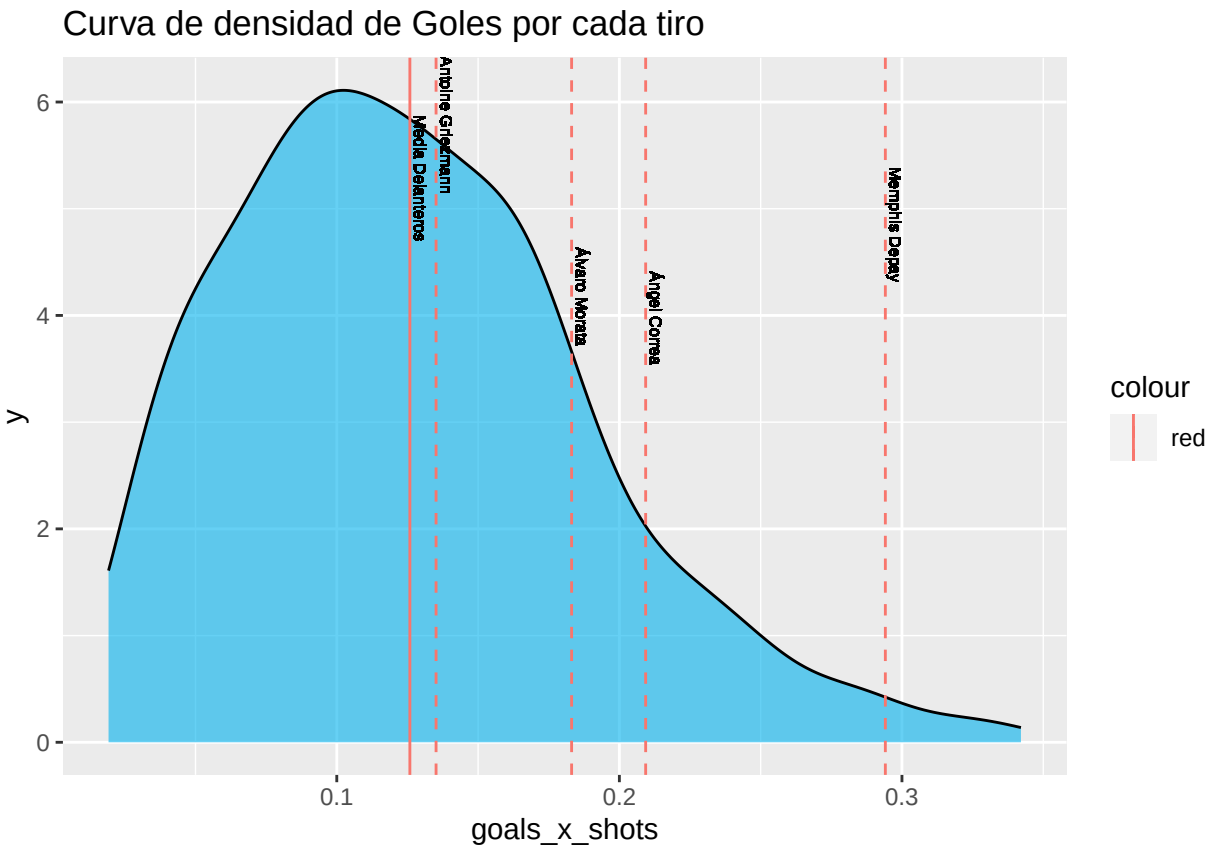
```
delantero %>%
  filter(goals > 0 & time > 500) %>%
  ggplot(aes(x = goals90)) +
  geom_density(fill="deepskyblue2", alpha=0.6) +
  geom_vline(aes(xintercept = delantero[delantero$player_name == "Memphis Depay",21 ], color="red"), li
    geom_vline(aes(xintercept = delantero[delantero$player_name == "Antoine Griezmann",21 ], color="red
    geom_vline(aes(xintercept = delantero[delantero$player_name == "Álvaro Morata",21 ], color="red"),
    geom_vline(aes(xintercept = delantero[delantero$player_name == "Ángel Correa",21 ], color="red"), li
  geom_vline(aes(xintercept = mean(goals90), color="red")) +
  geom_text(mapping=aes(x=mean(goals90), y = 0, label = "Media Delanteros"),
            size=2, angle=270, vjust=-0.4, hjust=5) +
     geom_text(mapping=aes(x=delantero[delantero$player_name == "Antoine Griezmann",21 ], y = 0, label =
    geom_text(mapping=aes(x=delantero[delantero$player_name == "Ángel Correa",21 ], y = 0, label = "Ángel
            size=2, angle=270, vjust=-0.4, hjust=5) +
    geom_text(mapping=aes(x=delantero[delantero$player_name == "Memphis Depay",21 ], y = 0, label = "Mem
            size=2, angle=270, vjust=-0.4, hjust=5) +
    geom_text(mapping=aes(x=delantero[delantero$player_name == "Álvaro Morata",21 ], y = 0, label = "Álv
            size=2, angle=270, vjust=-0.4, hjust=5) +
  labs(title="Curva de densidad de Goles por 90 Minutos")
```

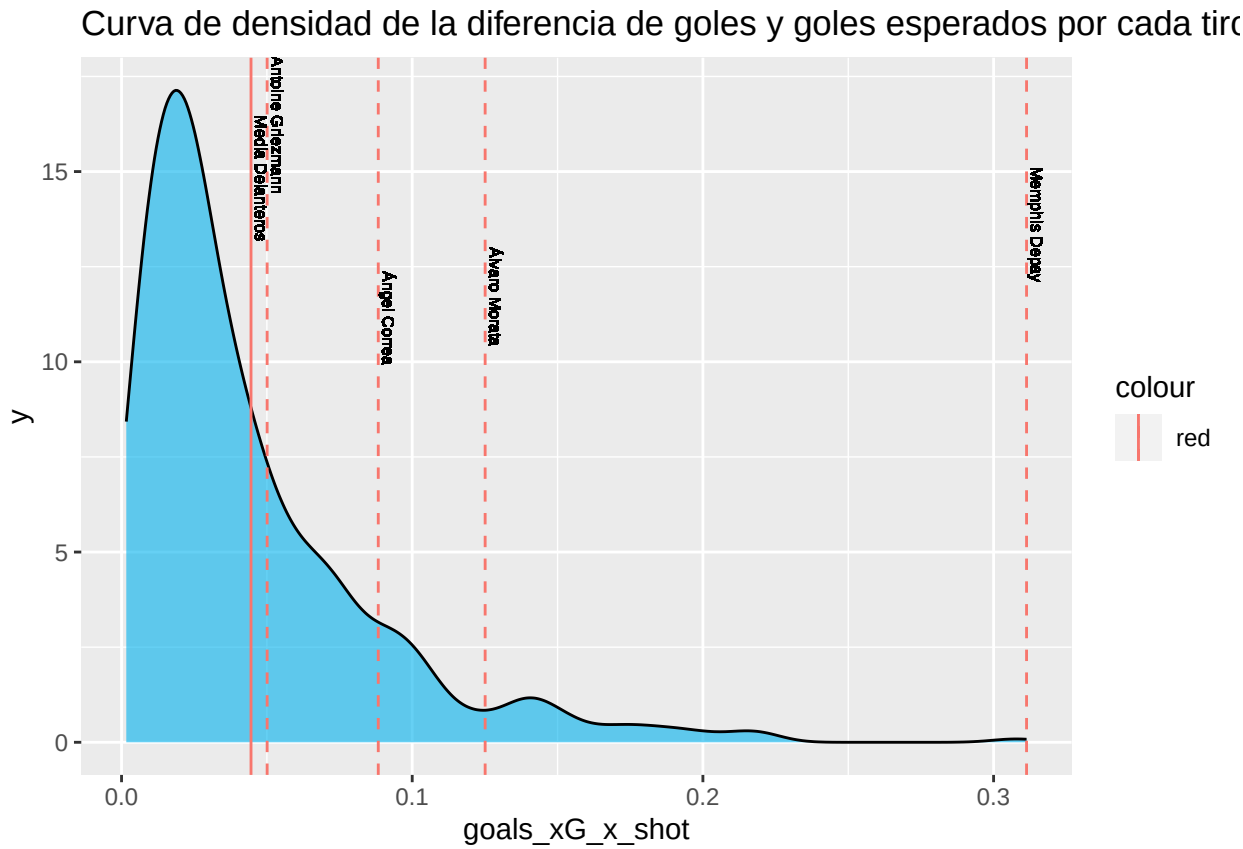


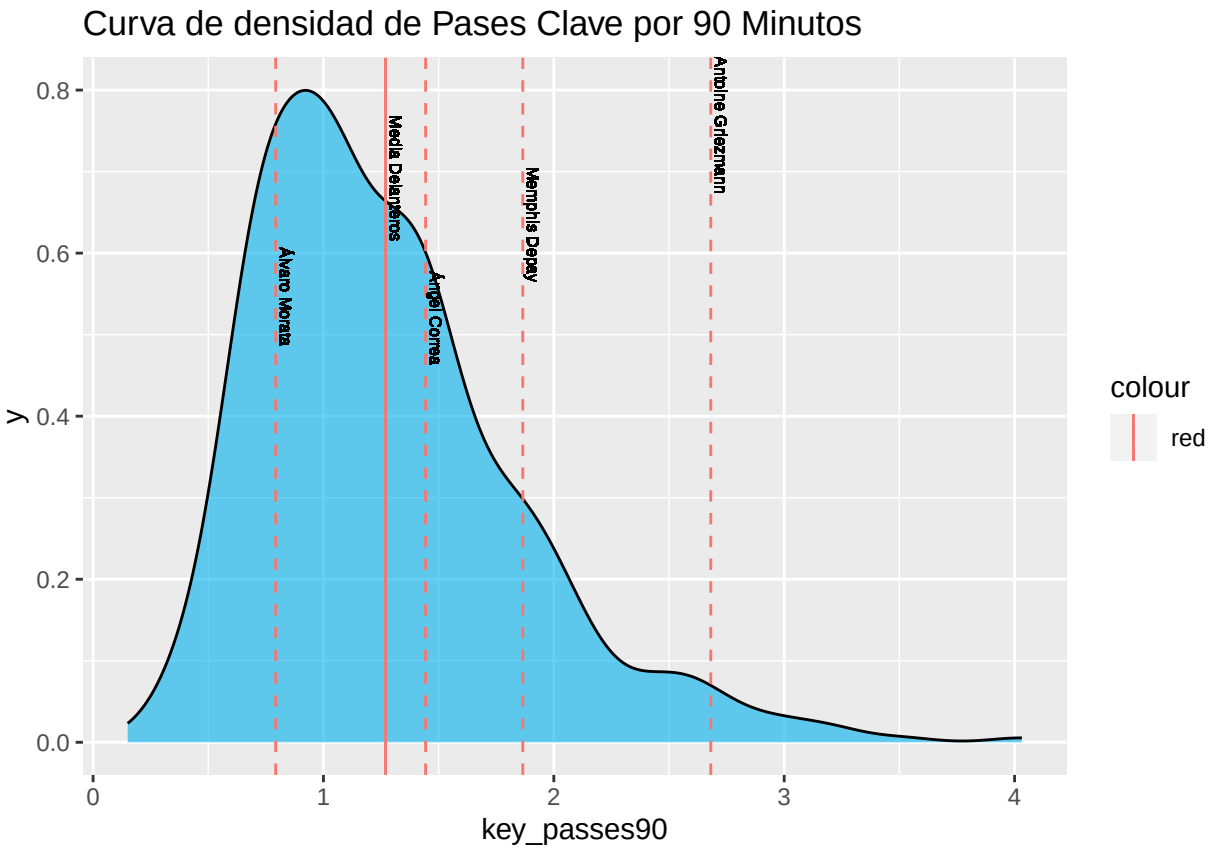


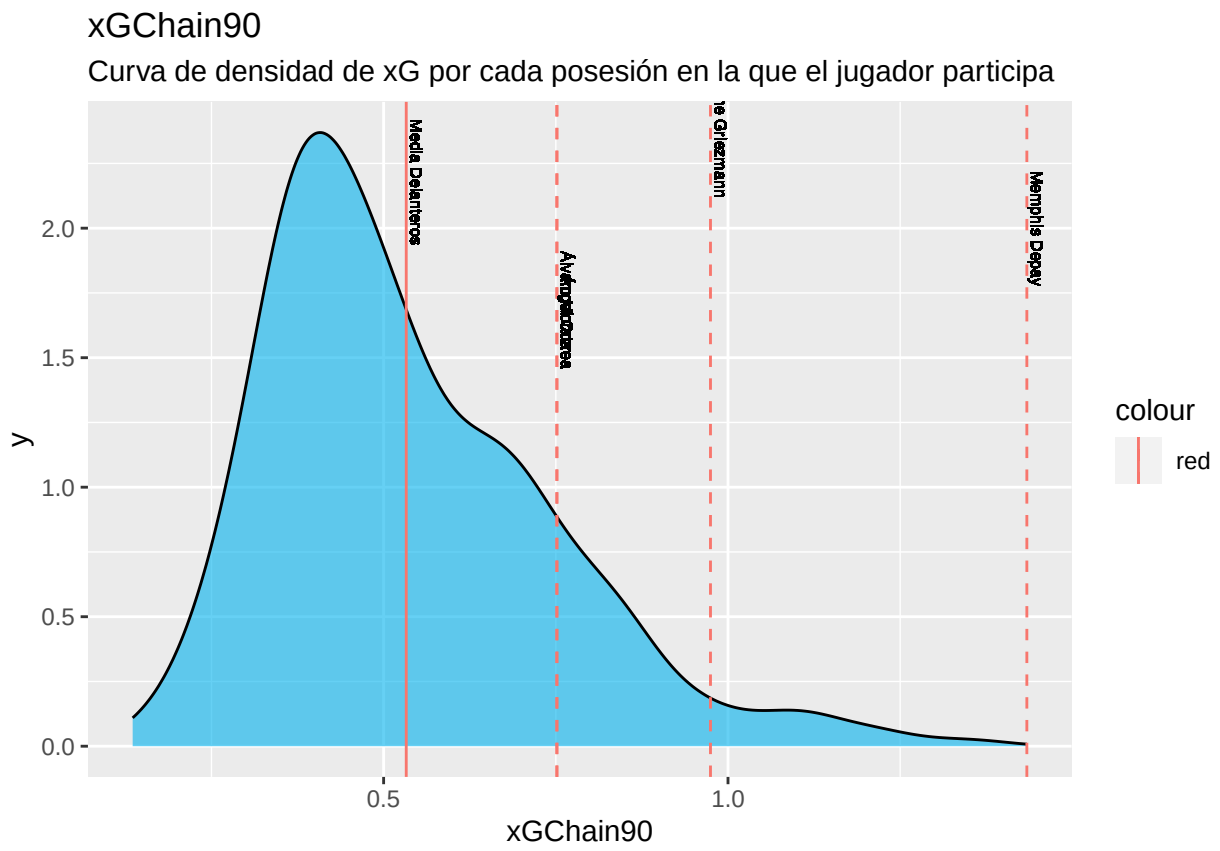
Creamos otras dos variables interesantes para el análisis y representación gráfica

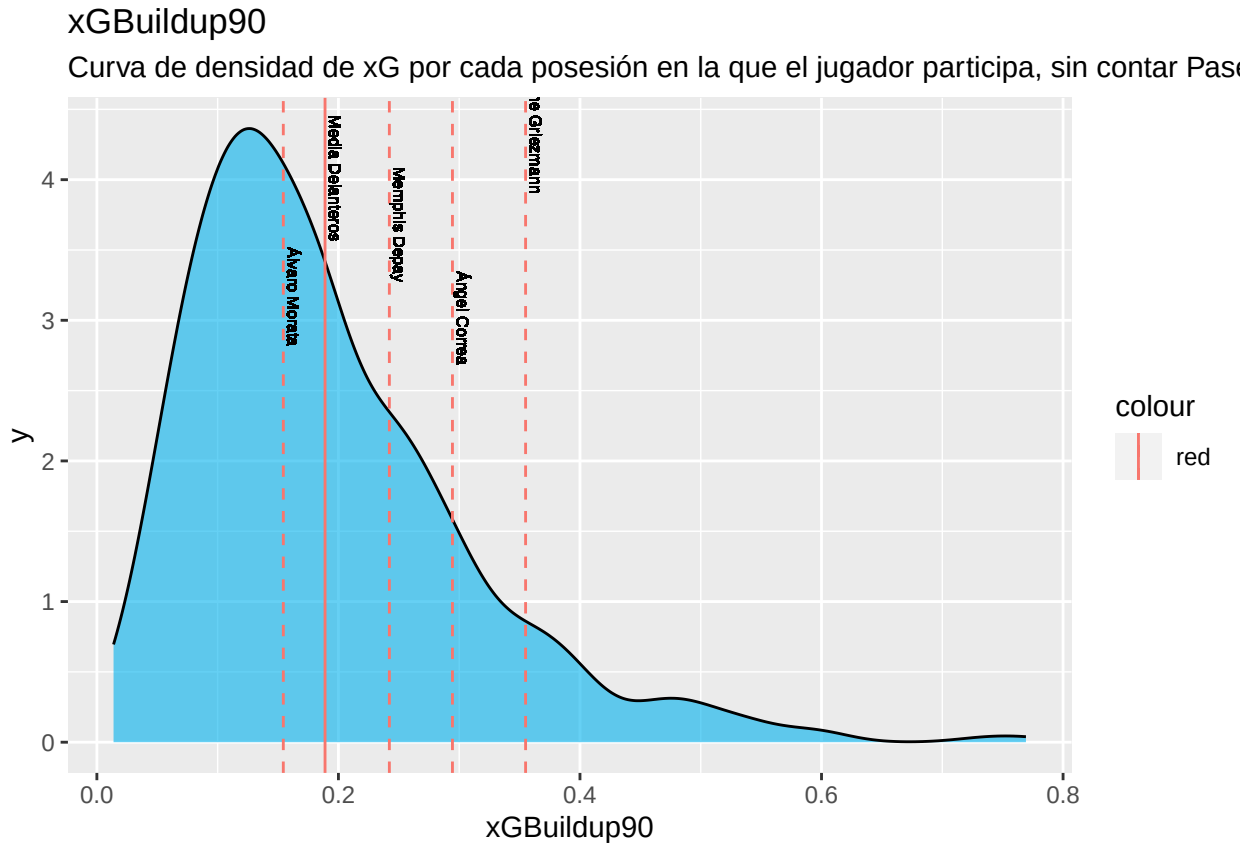
```
delantero <- delantero %>%
  cbind(goals_x_shots = delantero$goals90 / delantero$shots90) %>%
  cbind(goals_xG_x_shot = delantero$goals90 * delantero$xG90 / delantero$shots90)
```


Curva de densidad de Goles por cada tiro
Media Delanteros
Antoine Griezmann
Álvaro Morata
Ángel Correa
Memphis Depay
y
0
2
4
6
0.1
0.2
0.3
goals_x_shots
colour
red


Curva de densidad de la diferencia de goles y goles esperados por cada tirc
Antoine Griezmann
Media Delanteros
Ángel Correa
Álvaro Morata
Memphis Depay
15
10
5
0
y
0.0
0.1
0.2
0.3
goals_xG_x_shot
colour
red


Curva de densidad de Pases Clave por 90 Minutos
Álvaro Morata
Media Delanteros
Ángel Correa
Memphis Depay
Antoine Griezmann
0.8
0.6
0.4
0.2
0.0
y
0
1
2
3
4
key_passes90
colour
red


xGChain90
Curva de densidad de xG por cada posesión en la que el jugador participa
Media Delanteros
Memphis Depay
colour
red
y
2.0
1.5
1.0
0.5
0.0
0.5
1.0
xGChain90



Se va a realizar un filtro con los valores mínimos que se ha interpretado como necesarios de las gráficas anteriores, y se mostrará el nombre de los jugadores adecuados.

```
delantero[, c(2,4,21:26, 29:33)] %>%
   filter(goals90 > 0.45 & time > 500 & assists90 > 0.15 & goals_x_shots > 0.15 & key_passes90 > 1.4 & x
```

```
##               player_name time   goals90      xG90 assists90      xA90  shots90
## 51            Munas Dabbur  897 0.6020067 0.3792386 0.2006689 0.1766452 2.207358
## 520       Roberto Firmino 1203 0.8229426 0.6077025 0.2992519 0.2285417 2.917706
## 1658              Neymar 1561 0.7495195 0.5715906 0.6342088 0.4659599 2.190903
## 3          Vincenzo Grifo 2461 0.5485575 0.4248969 0.1828525 0.2366444 2.413653
## 5            Serge Gnabry 1950 0.6461538 0.5330467 0.2307692 0.2607164 3.784615
## 1645 Kylian Mbappe-Lottin 2835 0.9206349 0.8826614 0.1587302 0.2911049 4.761905
## 2243        Romelu Lukaku 1675 0.5373134 0.5269276 0.3223881 0.2285849 2.740299
## 1      Christopher Nkunku 1897 0.7590933 0.7728945 0.1897733 0.2326136 3.510807
## 510         Mohamed Salah 3307 0.5170850 0.6351998 0.3265800 0.2373215 3.401875
##      key_passes90 xGChain90 xGBuildup90      dif_xG goals_x_shots
## 51       1.605351 0.7826012   0.2918275  0.22276814     0.2727273
## 520      1.571072 1.0408461   0.5005115  0.21524017     0.2820513
## 1658     2.825112 1.3207866   0.7352456  0.17792898     0.3421053
## 3        2.011377 0.4476287   0.2413099  0.12366061     0.2272727
## 5        1.661538 1.1203708   0.4770068  0.11310715     0.1707317
## 1645     1.777778 1.1810819   0.3101497  0.03797356     0.1933333
## 2243     1.719403 0.8265280   0.2636352  0.01038580     0.1960784
## 1        1.802847 0.9167246   0.2276966 -0.01380122     0.2162162
## 510      1.768975 0.8856887   0.2413095 -0.11811483     0.1520000
```

```
##      goals_xG_x_shot
## 51        0.10342870
## 520       0.17140326
## 1658      0.19554414
## 3         0.09656748
## 5         0.09100797
## 1645      0.17064786
## 2243      0.10331914
## 1         0.16711233
## 510       0.09655037
```

Se ha obtenido una lista de 9 jugadores para una posible sustitución de los 4 que tiene el equipo. El coste del fichaje de estos 10 jugadores varía mucho y dependería de la situación económica del club en ese momento.

CENTROCAMPISTAS

LO primero que se hará será crear el dataframe correspondiente a los mediocentros y eliminar los que no hayan jugado el tiempo suficiente para que sus datos sean representativos.

```
mc <- rbind(df1[df1$position1 == "M",])
mc <- mc %>% filter(!is.na(mc$player_name)) %>%  filter(time > 350)
mc <- distinct(mc) %>%
  cbind(goals_xG_x_shot = mc$goals90 * mc$xG90 / mc$shots90) %>%
  cbind(goals_xG = mc$goals90 - mc$xG90)
```

Veamos cuántos centrocampistas están por encima de la media en las características típicas de su posición.

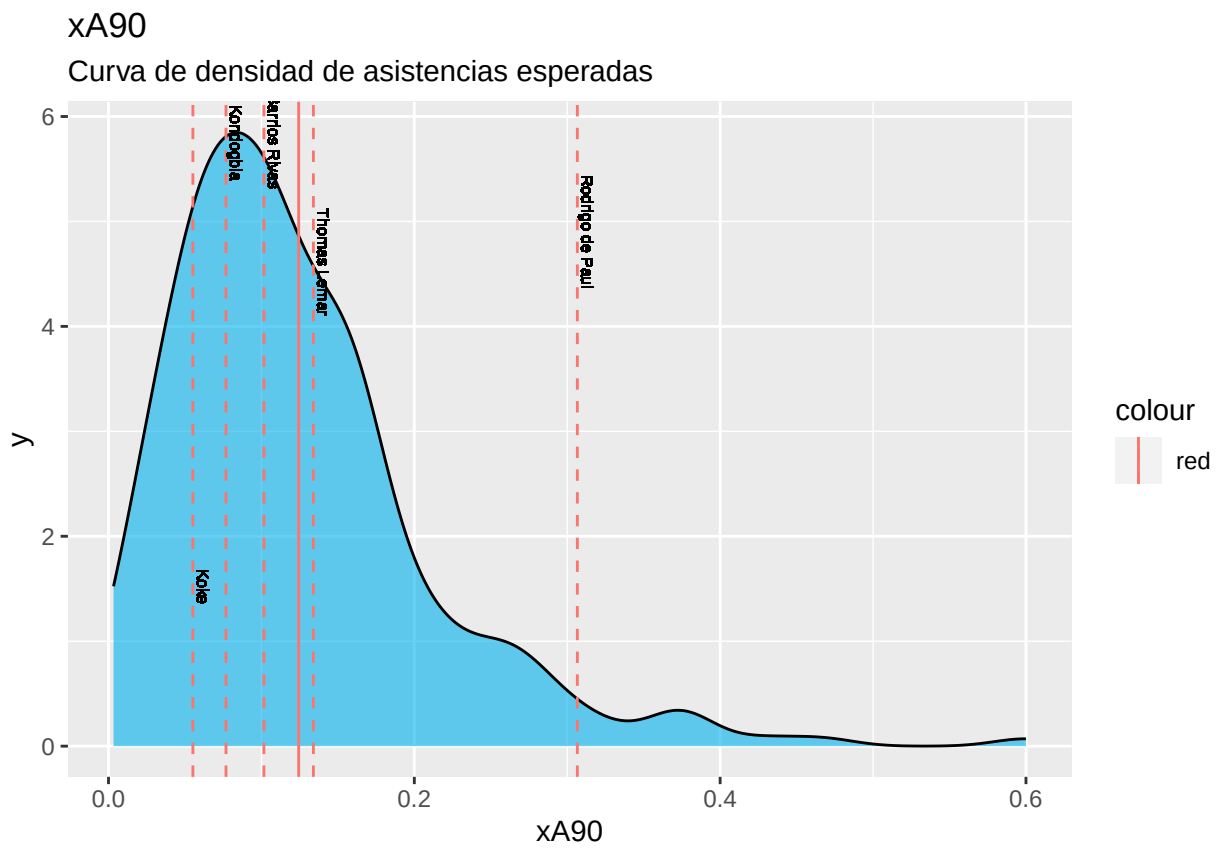
```
mc %>%
   filter( xA90 > mean(xA90) & key_passes90 > mean(key_passes90) & xGBuildup90 > mean(xGBuildup90)) %>%
dim()
```

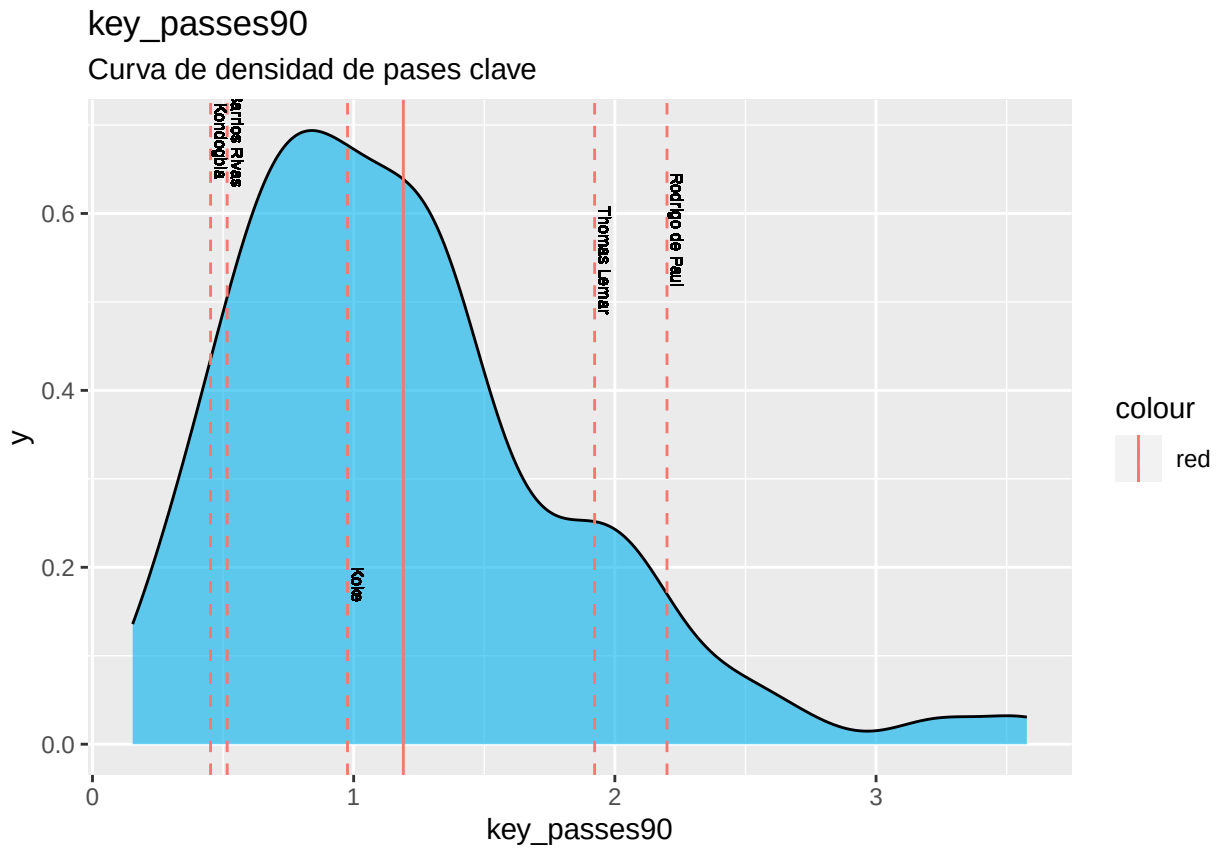
```
## [1] 85 32
```

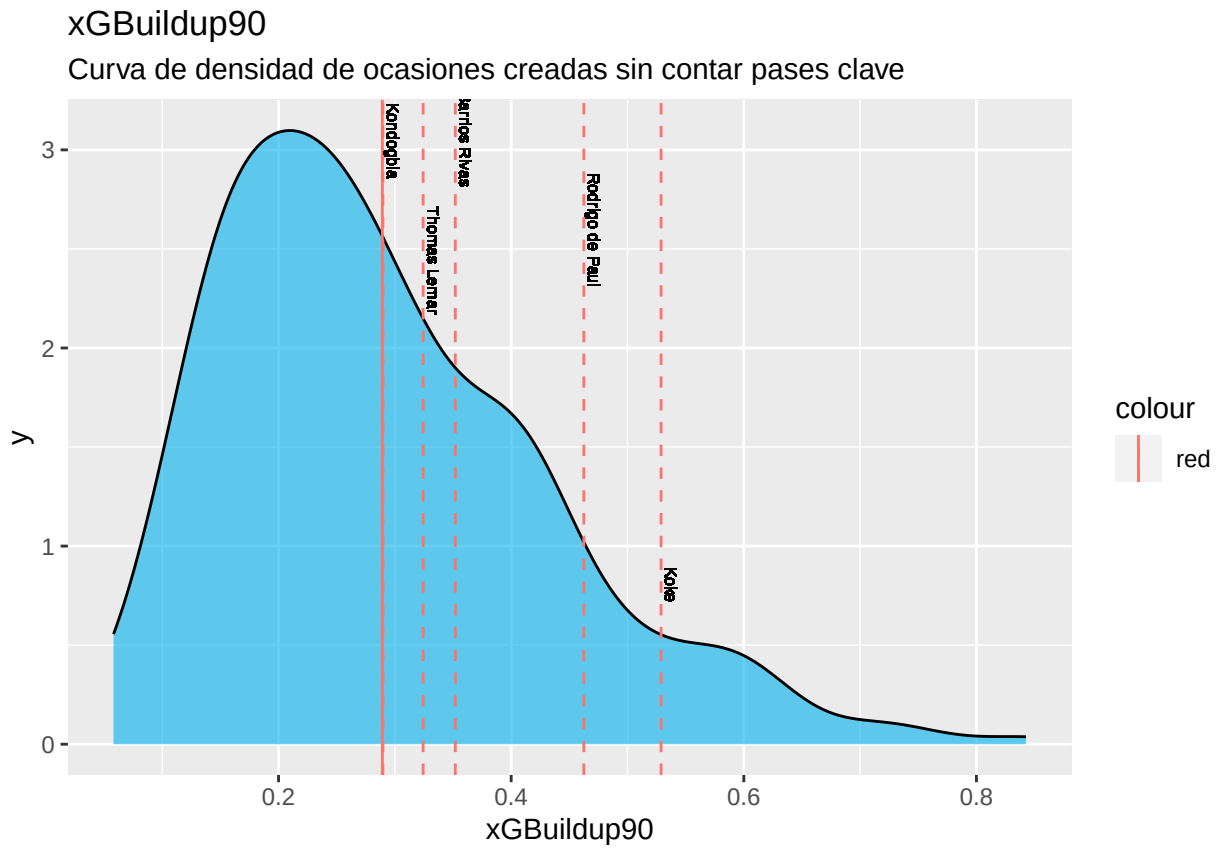
Hay 85 jugadores que cumplen con los filtros realizados. Son demasiados, por lo que se realizarán gráficos similares a los utilizados en la posición de delantero para filtrar más exhaustivamente los datos.

```
mc %>%
   filter(goals > 0 & time > 500) %>%
   ggplot(aes(x = xA90)) +
   geom_density(fill="deepskyblue2", alpha=0.6) +
   geom_vline(aes(xintercept = mc[mc$player_name == "Rodrigo de Paul",24 ], color="red"), linetype="dash
   geom_vline(aes(xintercept = mc[mc$player_name == "Thomas Lemar",24 ], color="red"), linetype="dashed
   geom_vline(aes(xintercept = mc[mc$player_name == "Geoffrey Kondogbia",24 ], color="red"), linetype="
   geom_vline(aes(xintercept = mc[mc$player_name == "Koke",24 ], color="red"),
              linetype="dashed") +
   geom_vline(aes(xintercept = mc[mc$player_name == "Pablo Barrios Rivas",24 ], color="red"),
              linetype="dashed") +
    geom_vline(aes(xintercept = mean(xA90), color="red")) +
    geom_text(mapping=aes(x=mean(xA90), y = 0, label = "Media Centrocampistas"),
             size=2, angle=270, vjust=-0.4, hjust=5) +
   geom_text(mapping=aes(x=mc[mc$player_name == "Rodrigo de Paul",24 ], y = 0, label = "Rodrigo de Paul
    geom_text(mapping=aes(x=mc[mc$player_name == "Thomas Lemar",24 ], y = 0, label = "Thomas Lemar"),
```

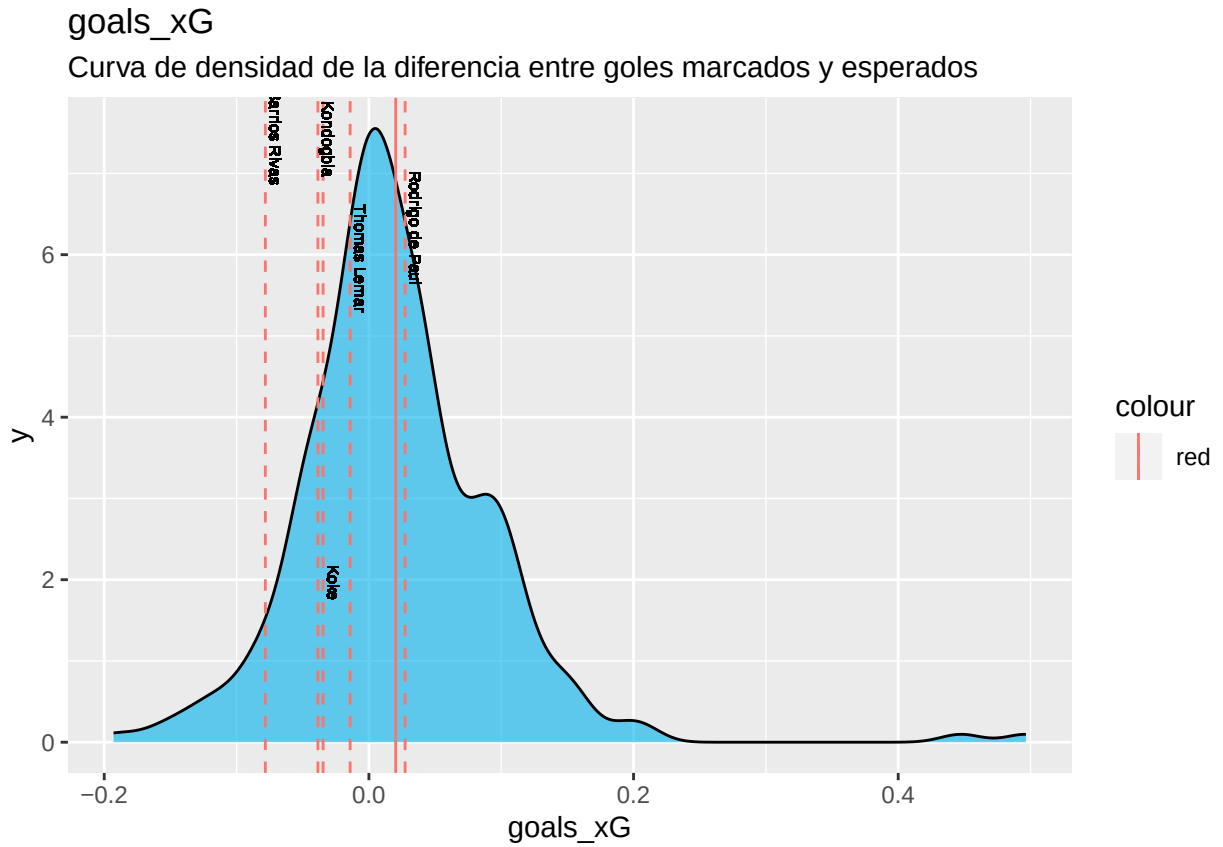
```
          size=2, angle=270, vjust=-0.4, hjust=5) +
  geom_text(mapping=aes(x=mc[mc$player_name == "Geoffrey Kondogbia",24 ], y = 0, label = "Geoffrey Kon
          size=2, angle=270, vjust=-0.4, hjust=5) +
  geom_text(mapping=aes(x=mc[mc$player_name == "Koke",24 ], y = 0, label = "Koke"),
          size=2, angle=270, vjust=-0.4, hjust=5) +
  geom_text(mapping=aes(x=mc[mc$player_name == "Pablo Barrios Rivas",24 ], y = 0, label = "Pablo Barri
          size=2, angle=270, vjust=-0.4, hjust=5) +
labs(title = "xA90", subtitle="Curva de densidad de asistencias esperadas")
```




key_passes90
Curva de densidad de pases clave
Jarrios Rivas
Kondogbia
Koke
Thomas Lemar
Rodrigo de Paul
0.6
0.4
0.2
0.0
y
0
1
2
3
key_passes90
colour
red



Las tres variables representadas son las más importantes para esta posición y son las que se tendrán en cuenta, pero como para determinar quién gana un partido se cuentan los goles, también se hará un último gráfico de diferencia entre goles y goles esperados.



Con toda la información de elos gráficos, ya podemos filtrar de forma más precisa para encontrar jugadores con al menos las mismas características que los 5 representados.

```
mc[, c(2,4,21:26, 29:32)] %>%
  filter( assists90 > 0.15 & goals_xG >= -0.5 & key_passes90 > 1.75 & xGBuildup90 > 0.45)
```

```
##          player_name time    goals90       xG90 assists90      xA90   shots90
## 1      Jamal Musiala 2234 0.48343778 0.37651540 0.4028648 0.2888762 2.6589078
## 2     Kingsley Coman 1422 0.50632911 0.38042854 0.3164557 0.3665046 3.1012658
## 3    Bruno Fernandes 3326 0.21647625 0.25396606 0.2164762 0.4672291 2.4894768
## 4    Kevin De Bruyne 2448 0.25735294 0.17493656 0.5882353 0.5999869 2.3897059
## 5       Fábio Vieira  468 0.19230769 0.14189953 0.3846154 0.4305446 2.8846154
## 6  N&#039;Golo Kanté  526 0.00000000 0.05274950 0.1711027 0.3557797 1.3688213
## 7        Luka Modric 1756 0.20501139 0.17561156 0.1537585 0.2628123 1.5375854
## 8    Rodrigo de Paul 1964 0.09164969 0.06427410 0.3207739 0.3066079 0.9623218
## 9         Toni Kroos 2181 0.08253095 0.04955442 0.1650619 0.2038234 1.2792297
## 10           Joaquín  437 0.00000000 0.24858717 0.4118993 0.3551709 1.6475973
## 11  Maxence Caqueret 2786 0.12921752 0.10733591 0.2261307 0.1781490 0.8399139
##    key_passes90 xGChain90 xGBuildup90 goals_xG_x_shot    goals_xG
## 1      2.094897 1.1129250   0.6187020     0.068457345  0.10692238
## 2      2.594937 1.1846752   0.5635866     0.062110782  0.12590057
## 3      3.193025 0.9583971   0.4796289     0.022084005 -0.03748981
## 4      3.566176 1.0709012   0.6115280     0.018839322  0.08241638
## 5      2.307692 1.0452089   0.6231808     0.009459968  0.05040817
## 6      2.224335 0.7700178   0.5111295     0.000000000 -0.05274950
## 7      2.152620 0.8175875   0.5915770     0.023414875  0.02939983
```

```
## 8      2.199593 0.7104027   0.4624547     0.006121343  0.02737560
## 9      2.228336 0.7629336   0.6973264     0.003197059  0.03297653
## 10     4.118993 0.9479848   0.5404653     0.000000000 -0.24858717
## 11     1.905958 0.6387383   0.4511230     0.016513217  0.02188161
```

Se han obtenido 11 jugadores que igualan o mejoran las estadísticas de los 5 jugadores del Atlético de Madrid.

En este curso se ha recomendado utilizar la mentalidad analítica para el proceso. Los filtros creados para obtener esta pequeña lista de jugadores se basan en la correcta interpretación de los gráficos realizados, no es la intuición sin base analítica.

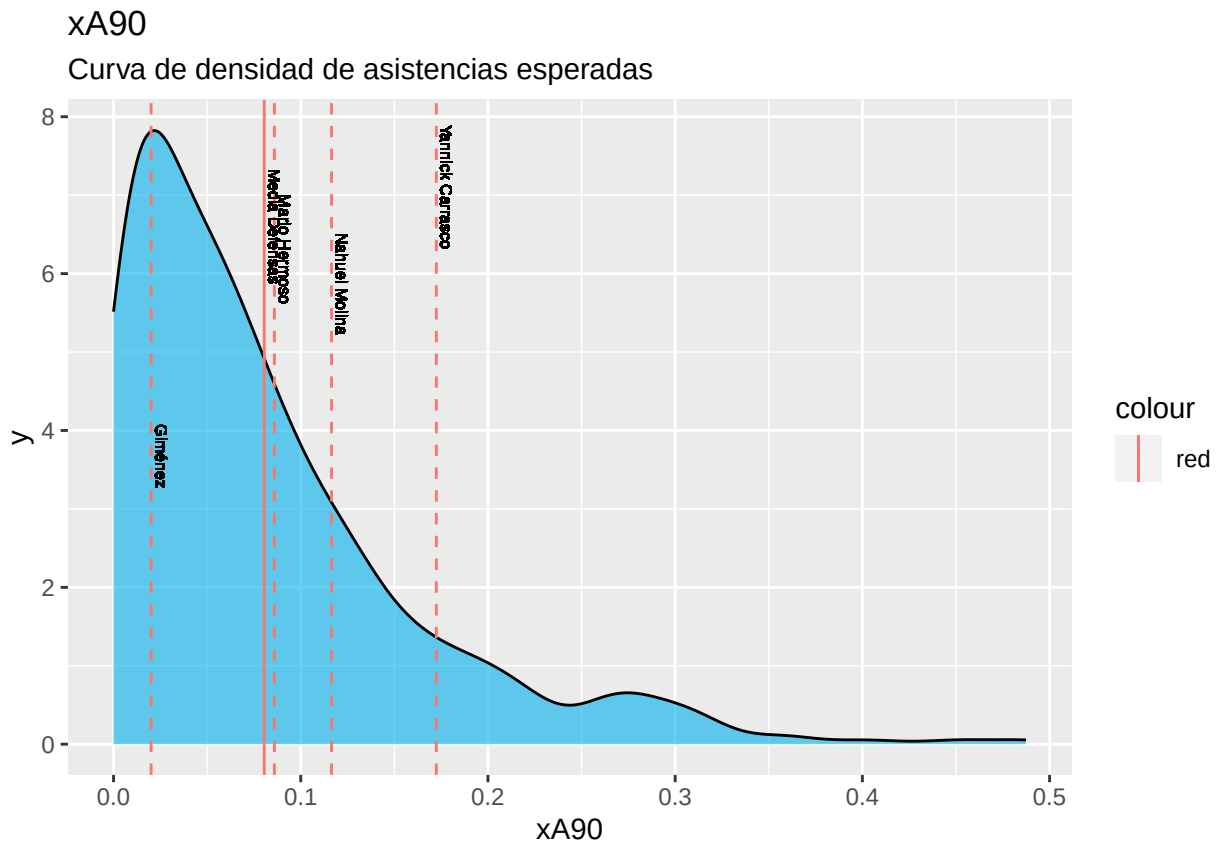
DEFENSAS

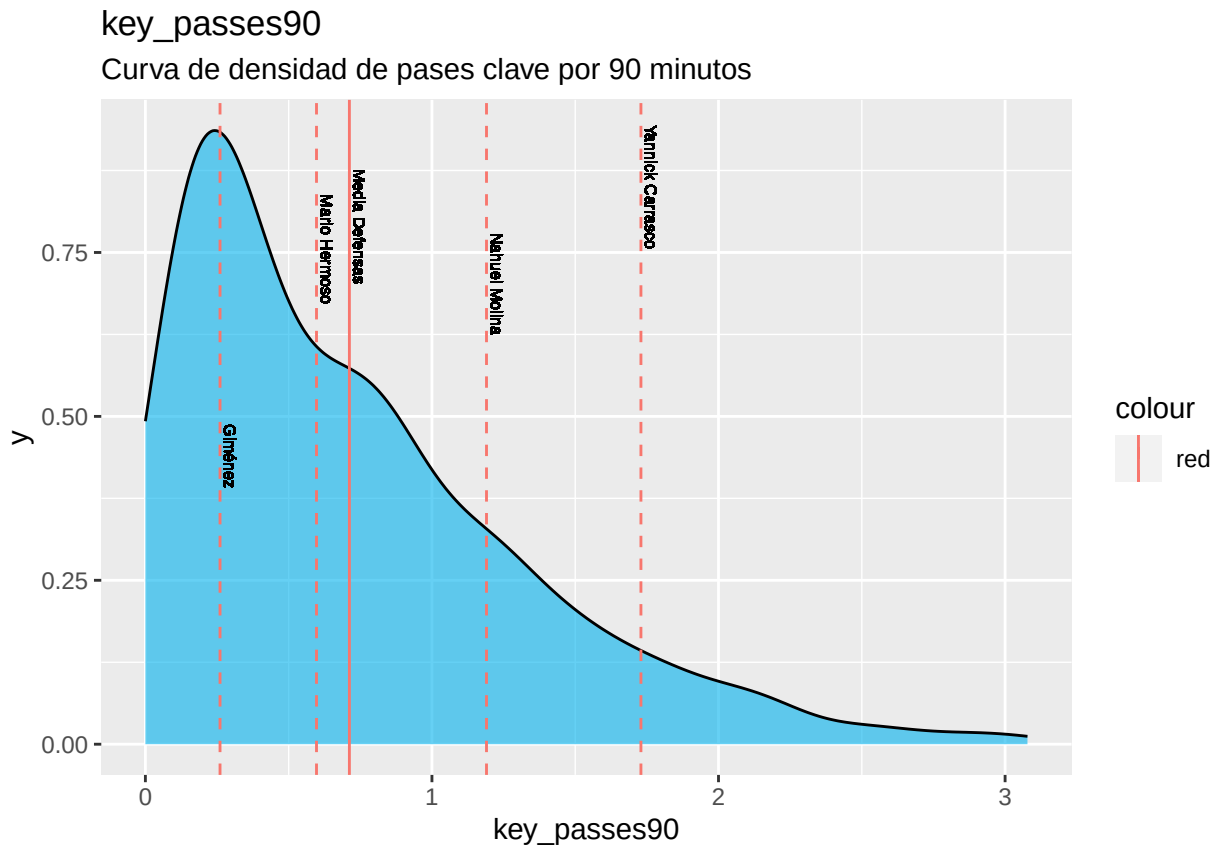
Este es el grupo de posiciones en el que menos valiosos son los datos que se tienen. Aún así, el análisis se realizará buscando defensas con vocación ofensivos debido a la naturaleza de los datos.

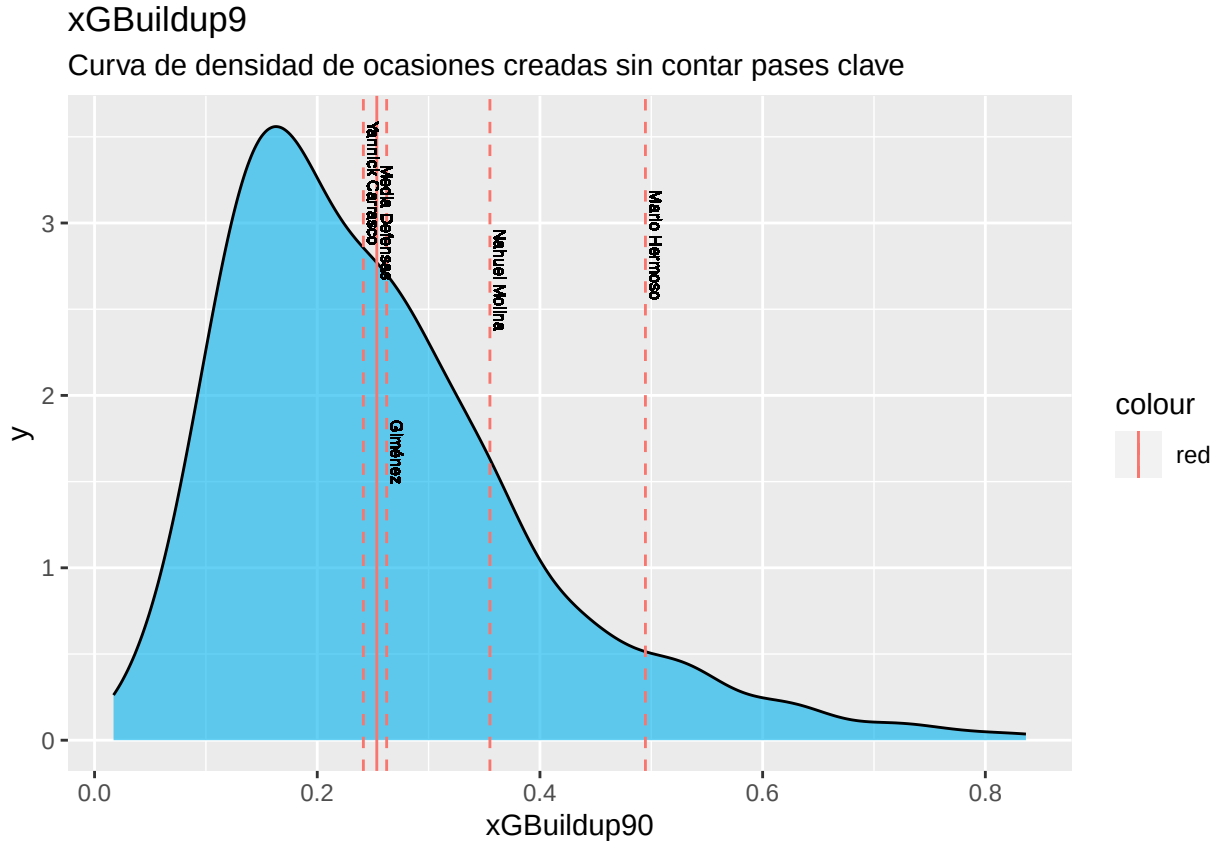
```
defend <- rbind(df1[df1$position1 == "D",])
defend <- defend %>% filter(!is.na(defend$player_name)) %>%  filter(time > 350)
defend <- distinct(defend) %>%
  cbind(goals_xG_x_shot = defend$goals90 * defend$xG90 / defend$shots90) %>%
  cbind(goals_xG = defend$goals90 - defend$xG90)

defend %>%
   filter( xA90 > mean(xA90) & key_passes90 > mean(key_passes90) & xGBuildup90 > mean(xGBuildup90)) %>%
  dim()
```

```
## [1] 151  32
```

Hay 151 jugadores que cumplen con los filtros realizados. Son demasiados, por lo que se realizarán gráficos similares a los utilizados en la posición de delantero para filtrar más exhaustivamente los datos.


xA90
Curva de densidad de asistencias esperadas
8
6
4
2
0
y
0.0
0.1
0.2
0.3
0.4
0.5
xA90
Giménez
Mario Hermoso
Nahuel Molina
Yannick Carrasco
colour
red


key_passes90
Curva de densidad de pases clave por 90 minutos
Yannick Carrasco
Nahuel Molina
Media Defensas
Mario Hermoso
Giménez
0.75
0.50
0.25
0.00
y
0
1
2
3
key_passes90
colour
red




```
defend[, c(2,4,21:26, 29:32)] %>%
  filter( assists90 > 0.15 & key_passes90 > 1.25 & xGBuildup90 > 0.35)
```

```
##               player_name time    goals90       xG90 assists90      xA90
## 1           Joshua Kimmich 2823 0.15940489 0.06286819 0.1912859 0.2265924
## 2        Raphael Guerreiro 2309 0.15591165 0.11411013 0.4677350 0.3045047
## 3          Alphonso Davies 2054 0.04381694 0.09703135 0.1752678 0.1521009
## 4        Noussair Mazraoui 1073 0.08387698 0.07231675 0.3355079 0.3206248
## 5             Pascal Groß 3261 0.24839006 0.16433492 0.2207912 0.3103780
## 6            Kaoru Mitoma 2306 0.27320035 0.36085027 0.1951431 0.2745965
## 7  Trent Alexander-Arnold 2954 0.06093433 0.05724442 0.2742045 0.4028503
## 8          Kieran Trippier 3368 0.02672209 0.02330158 0.1870546 0.3597518
## 9                Estupiñán 2707 0.03324714 0.07729735 0.1662357 0.1875950
## 10       Andrew Robertson 2609 0.00000000 0.03759803 0.2759678 0.2352921
## 11             Jordi Alba 1414 0.12729844 0.04932470 0.1909477 0.4508009
## 12          Edon Zhegrova 1051 0.25689819 0.34802040 0.3425309 0.4873687
## 13            Nuno Mendes 1565 0.05750799 0.10038332 0.3450479 0.2741505
## 14              Mário Rui 1781 0.00000000 0.03441271 0.3032004 0.1577746
##      shots90 key_passes90 xGChain90 xGBuildup90 goals_xG_x_shot     goals_xG
## 1  0.8926674     2.709883 0.9279154   0.7875906     0.011226462  0.096536702
## 2  1.2862711     2.260719 0.6801544   0.5008172     0.013831531  0.041801519
## 3  0.8325219     1.884129 0.8968362   0.6978629     0.005106913 -0.053214408
## 4  0.6710158     1.593663 1.1323001   0.8364744     0.009039594  0.011560230
## 5  1.1039558     2.207912 0.6969386   0.4142144     0.036975356  0.084055148
## 6  2.0294883     1.561145 0.8212886   0.3551828     0.048575998 -0.087649921
```

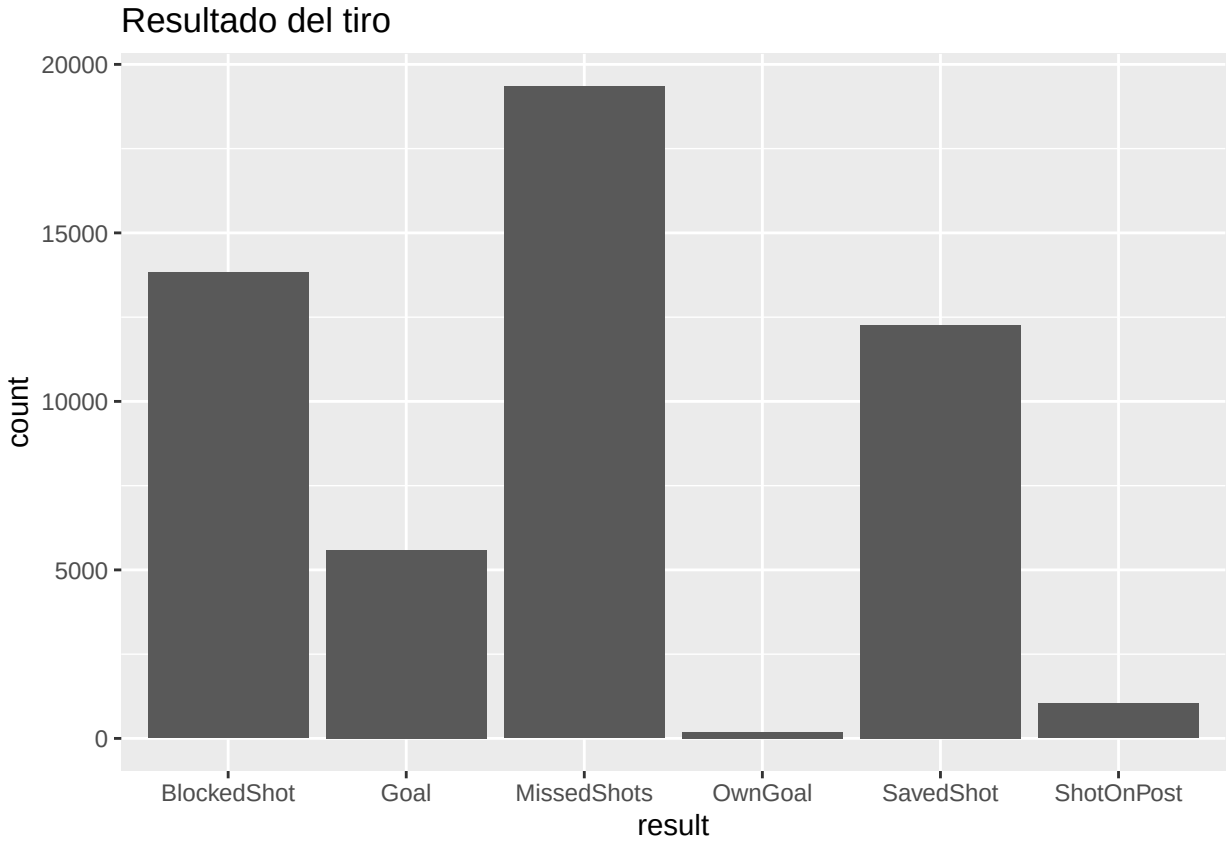
```
## 7  1.2186865     2.163169 0.6662740   0.4932559     0.002862221  0.003689906
## 8  0.3741093     2.939430 0.4668567   0.3902864     0.001664399  0.003420508
## 9  0.9974141     1.662357 0.6013318   0.4212923     0.002576578 -0.044050213
## 10 0.4484477     1.793791 0.4534521   0.3568093     0.000000000 -0.037598033
## 11 0.8274399     2.418670 0.9054322   0.6436719     0.007588416  0.077973739
## 12 3.5965747     2.568982 0.9491734   0.4492621     0.024858600 -0.091122207
## 13 0.5750799     1.437700 0.8011301   0.5331679     0.010038332 -0.042875328
## 14 0.5558675     2.324537 0.6283959   0.5695270     0.000000000 -0.034412706
```

Hay 14 jugadores posibles para sustituir las bajas de los actuales que cumplen con las métricas mínimas para al menos igualar lo que realizan estos jugadores actuales.

PORTEROS

Para este análisis, debido a la falta de datos, se realizará de forma distinta. Se va a utilizar el data frame de tiros realizados, donde se buscarán mediante el procesamiento de datos los equipos con mayor diferencia positiva entre los goles esperados y los recibidos. Es decir, se buscarán los equipos donde su/sus porteros hayan encajado menos goles de los esperados. Cuando se obtegan los datos, se hará un trabajo de investigación para comprobar si todos los partidos los ha jugado el mismo portero o no.

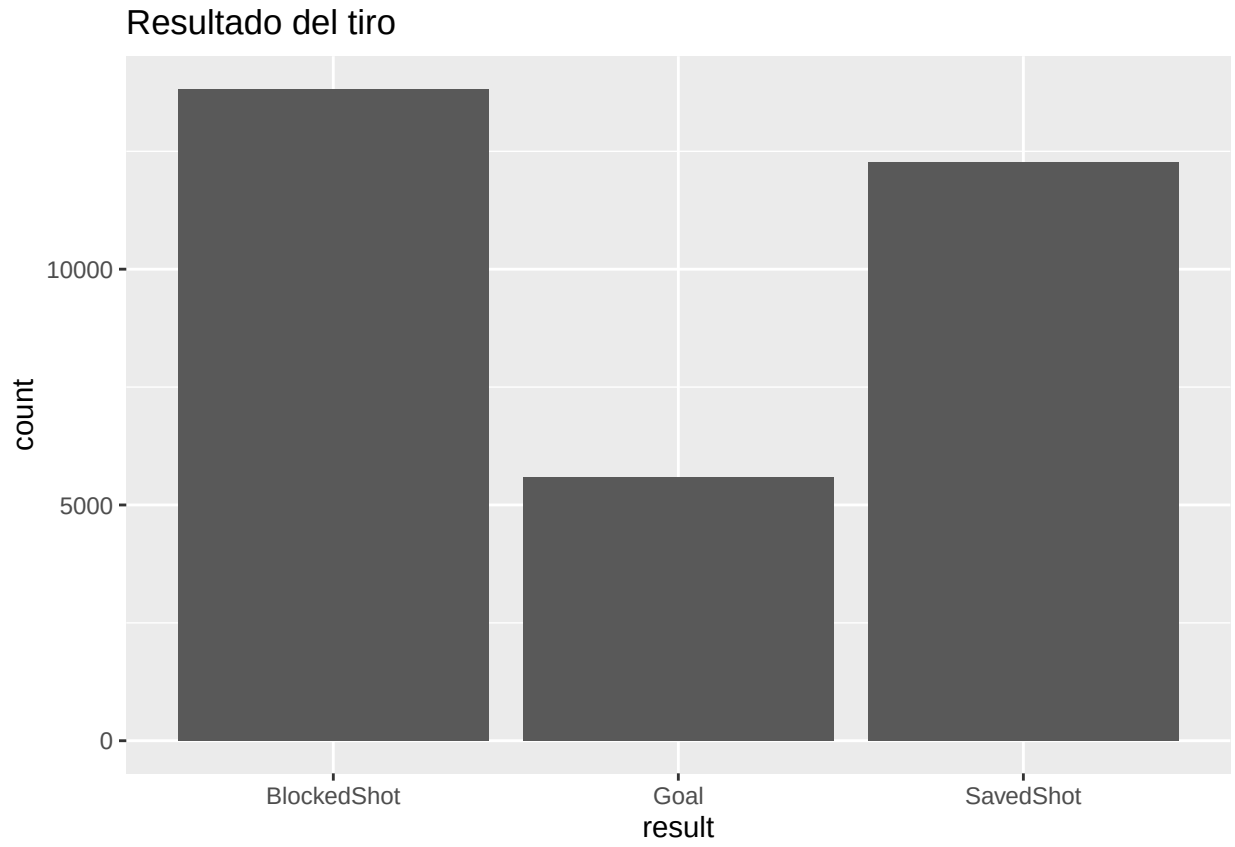
```
shots_dataset %>%
  ggplot(aes(result)) +
  geom_bar() +
  labs(title="Resultado del tiro")
```



Se van a eliminar MissedShots, OwnGoal y ShotOnPost. En estos tiros no interviene el portero ni acaban

en gol por lo que no interesan para el análisis. Además, para un análisis posterior se creará una variable nueva llamada "goal_no" que tendrá dos posibles respuestas: Sí y No. A "No" pertenecerán BlockedShot y SavedShot. La diferencia entre estas dos es que un tiro bloqueado el portero consigue quedarse con el balón y en un tiro salvado (parada) no lo consigue.

```
shots_dataset <- rbind(shots_dataset[shots_dataset$result == "BlockedShot", ], shots_dataset[shots_datas

shots_dataset %>%
  ggplot(aes(result)) +
  geom_bar() +
  labs(title="Resultado del tiro")
```




```
# Mediante filtros se crea un dataframe con la suma de xG de los tiros RECIBIDOS jugando de LOCAL y por
shots_home <- shots_dataset[, c("result", "xG", "h_a", "h_team", "a_team")] %>%
  filter(h_a == "a") %>%
  group_by(h_team, h_a, result) %>%
  summarise(xG_accumulate = sum(xG))
```

```
## `summarise()` has grouped output by 'h_team', 'h_a'. You can override using the
## `.groups` argument.
```

```
# Mediante filtros se crea un dataframe con la suma de xG de los tiros RECIBIDOS jugando de VISITANTE y
shots_away <- shots_dataset[, c("result", "xG", "h_a", "h_team", "a_team")] %>%
  filter(h_a == "h") %>%
  group_by(a_team, h_a, result) %>%
  summarise(xG_accumulate = sum(xG))
```

```
## `summarise()` has grouped output by 'a_team', 'h_a'. You can override using the
## `.groups` argument.
```

```
#Ejemplo de las 3 filas por equipo que se obtienen
shots_home[1:3,]
```

```
## # A tibble: 3 x 4
## # Groups:   h_team, h_a [1]
##   h_team   h_a   result      xG_accumulate
##   <chr>    <chr> <chr>               <dbl>
## 1 AC Milan a     BlockedShot          2.35
## 2 AC Milan a     Goal                 6.45
## 3 AC Milan a     SavedShot            3.66
```

```
# Se incorporan los dos dataframe creados y se agrupan de nuevo por equipo y resultado obteniendo la su
colnames(shots_home) <- c("team", "h_a","result", "xG_accumulate")
colnames(shots_away) <- c("team", "h_a","result", "xG_accumulate")
shots_dif <- rbind(shots_home, shots_away)
```

```
shots_dif[ , c("team", "result", "xG_accumulate")] %>%
  group_by(team, result) %>%
  summarise(xG_accumulate = sum(xG_accumulate))
```

```
## `summarise()` has grouped output by 'team'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 342 x 3
## # Groups:   team [114]
##    team     result      xG_accumulate
##    <chr>    <chr>               <dbl>
##  1 AC Milan BlockedShot          7.04
##  2 AC Milan Goal                14.9
##  3 AC Milan SavedShot            7.44
##  4 Ajaccio  BlockedShot          5.77
##  5 Ajaccio  Goal                25.7
##  6 Ajaccio  SavedShot            9.71
##  7 Almeria  BlockedShot          6.96
##  8 Almeria  Goal                20.6
##  9 Almeria  SavedShot           19.3
## 10 Angers   BlockedShot          7.22
## # i 332 more rows
```

```
#Modificaré el DF para obtener una fila por equipo
blocked <- shots_dif %>%
  filter(result == "BlockedShot") %>%
  mutate(Blocked_shot = xG_accumulate)
goal <- shots_dif %>%
  filter(result == "Goal") %>%
  mutate(goal = xG_accumulate)
saved <- shots_dif %>%
  filter(result == "SavedShot") %>%
  mutate(saved = xG_accumulate)
```

```
shots_dif <- rbind(blocked, goal, saved)
shots_dif <- shots_dif[order(shots_dif$team), ]
shots_dif <- shots_dif[ , c("team", "result", "Blocked_shot", "goal", "saved")] %>%
  group_by(team, result) %>%
  summarise(Blocked_shot = sum(Blocked_shot), goal = sum(goal), saved = sum(saved))
```

```
## `summarise()` has grouped output by 'team'. You can override using the
## `.groups` argument.
```

```
team_1 <- unique(shots_dif$team)

shots_dif <- shots_dif[ , c("team", "Blocked_shot", "goal", "saved")] %>%
  group_by(team = (row_number() - 1) %/% 3) %>%
  summarize(
    Blocked_shot = paste(na.omit(Blocked_shot), collapse = " "),
    goal = paste(na.omit(goal), collapse = " "),
    saved = paste(na.omit(saved), collapse = " ")
  ) %>%
  ungroup() %>%
  cbind(team_1 = team_1)

shots_dif <- shots_dif[, 2:5]
```

Ahora se crearán variables a partir del Data Frame limpio. Estas variables son: * Porcentaje de blocadas por parada * Diferencia entre paradas y goles esperados * Paradas/gol

```
shots_dif$Blocked_shot <- as.numeric(shots_dif$Blocked_shot)
shots_dif$goal <- as.numeric(shots_dif$goal)
shots_dif$saved <- as.numeric(shots_dif$saved)
shots_dif$porc_block = shots_dif$Blocked_shot/(shots_dif$Blocked_shot+shots_dif$saved)*100
shots_dif$dif_paradas = shots_dif$Blocked_shot+shots_dif$saved-shots_dif$goal
shots_dif$paradas_goal = (shots_dif$Blocked_shot+shots_dif$saved)/shots_dif$goal
```

Ahora se buscan los valores del Atlético de Madrid y se filtrará para buscar equipos con mejores valores.

```
shots_dif[shots_dif$team_1 == "Atletico Madrid", ]
```

```
##   Blocked_shot     goal    saved          team_1 porc_block dif_paradas
## 9     5.640751 11.67811 8.564547 Atletico Madrid   39.70878    2.527185
##   paradas_goal
## 9     1.216404
```

```
shots_dif %>%
  filter(porc_block > 40 & dif_paradas > 2.5 & paradas_goal > 1.2) %>% dim()
```

```
## [1] 20  7
```

```
#Con estos filtros se obtienen 20 equipos, si se quiere filtrar más, se podrían aumentar el filtro al c

shots_dif %>%
  filter(porc_block > quantile(porc_block, probs = 0.75) & dif_paradas > quantile(dif_paradas, probs = 
```

```
##   Blocked_shot     goal    saved                  team_1 porc_block dif_paradas
## 1    10.768777 13.13361 13.42421               Brentford   44.51197   11.059381
## 2    11.595980 15.15195 14.37391                  Empoli   44.65163   10.817944
## 3     9.486481 13.74288 11.51371       Manchester United   45.17332    7.257304
## 4     9.736773 15.13990 11.25855                  Verona   46.37591    5.855428
## 5    11.633311 17.95830 13.38567 Wolverhampton Wanderers   46.49794    7.060679
##   paradas_goal
## 1     1.842067
## 2     1.713964
## 3     1.528077
## 4     1.386755
## 5     1.393171
```

Se obtienen 5 equipos con muy buenas métricas. Ahora se comprobará el número de minutos que han tenido los porteros de estos equipos para ver si las métricas se le pueden atribuir a un único portero.

```
gk <- rbind(df1[df1$position1 == "GK",])
# Eliminamos los que hayan jugado menos de 350 min puesto que no son representativos
gk <- gk %>% filter(!is.na(gk$player_name)) %>%  filter(time > 350)
gk[, c("player_name", "time", "team_title")] %>% filter (team_title == "Wolverhampton Wanderers" | team
```

```
##         player_name time              team_title
## 1      David de Gea 3420       Manchester United
## 2        David Raya 3420               Brentford
## 3           José Sá 3240 Wolverhampton Wanderers
## 4   Samuele Perisan  630                  Empoli
## 5 Guglielmo Vicario 2784                  Empoli
## 6   Lorenzo Montipò 3330                  Verona
```

Podemos observar que los dos porteros del Empoli sí se han repartido los minutos de juego, por lo que las métricas en este caso no son fiables. Sin embargo, en el caso De Gea (M. United), Raya (Brentford), VJosé Sá (Wolves) y Montipò (Verona) serían jugadores muy interesantes si hiciesen falta.