Codigo_reducido

ManuelRuizBotella

December 25, 2020

1. Descripción del dataset.

(AÑADIR DESCRIPCION INICIAL)

```
# Tenemos que tener el fichero en el mismo directorio que el codigo.
fichero <- paste(getwd(),'Lol_ProfessionalGames.csv', sep='/')
fichero
```

```
## [1] "C:/Users/48784566-W/Desktop/Master/TCV/Practica2_TCV/Lol_ProfessionalGames.csv"
```

```
lol <- read.csv2(fichero,sep=',')
lol_base<- lol

# Comprobar que los datos se han cargado en el dataframe correctamente
str(lol)
```

```
## 'data.frame':    614 obs. of  517 variables:
##  $ torneo                           : Factor w/ 3 levels "LCS","LEC","WORLDS": 3 3 3 3 3 3 3 3 3 3
##  $ parte                            : Factor w/ 10 levels "LCS Spring 2020",..: 10 10 10 10 10 10 10
##  $ fecha                            : Factor w/ 122 levels "1/24/2020","1/25/2020",..: 120 120 120 1
##  $ semana                           : Factor w/ 27 levels "DAY1","DAY2",..: 11 11 11 11 11 11 9 9 9
##  $ tiempo                           : Factor w/ 534 levels "19:03","20:01",..: 220 263 182 114 407 1
##  $ parche                           : Factor w/ 13 levels "v10.1","v10.11",..: 8 8 8 8 8 8 8 8 8 8
##  $ nombre_azul                      : Factor w/ 35 levels "100 Thieves",..: 34 24 34 18 18 18 24 21
##  $ nombre_rojo                      : Factor w/ 35 levels "100 Thieves",..: 24 34 24 19 19 19 21 24
##  $ gana_azul                        : int  1 0 1 0 0 0 1 1 1 1 ...
##  $ gana_rojo                        : int  0 1 0 1 1 1 0 0 0 0 ...
##  $ num_asesinatos_azul              : int  21 11 17 3 18 4 22 14 21 13 ...
##  $ num_asesinatos_rojo              : int  12 24 10 9 17 14 14 9 8 7 ...
##  $ primera_sangre                   : Factor w/ 35 levels "100 Thieves",..: 24 34 34 19 19 18 24 21
##  $ num_torres_azul                  : int  11 0 9 1 7 3 7 10 8 8 ...
##  $ num_torres_rojo                  : int  1 11 2 11 7 10 4 1 2 3 ...
##  $ primera_torre                    : Factor w/ 36 levels "","100 Thieves",..: 35 35 35 20 20 20 25
##  $ num_dragones_azul                : int  3 2 4 1 3 1 3 3 3 1 ...
##  $ num_dragones_viento_azul         : int  1 0 1 0 0 1 1 0 1 0 ...
##  $ num_dragones_infierno_azul       : int  2 0 1 1 1 0 1 0 0 1 ...
##  $ num_dragones_oceano_azul         : int  0 1 0 0 1 0 1 0 1 0 ...
##  $ num_dragones_montaÃ.a_azul       : int  0 1 2 0 1 0 0 3 1 0 ...
##  $ num_dragones_rojo                : int  1 3 0 4 2 2 3 2 1 3 ...
##  $ num_dragones_viento_rojo         : int  0 0 0 3 0 0 0 1 0 1 ...
```

```
##  $ num_dragones_infierno_rojo       : int  0 3 0 0 0 0 0 1 0 1 ...
##  $ num_dragones_oceano_rojo         : int  1 0 0 0 0 1 3 0 0 1 ...
##  $ num_dragones_montaÃ.a_rojo       : int  0 0 0 1 2 1 0 0 1 0 ...
##  $ num_nashors_azul                 : int  2 0 2 0 2 0 0 1 1 1 ...
##  $ num_nashors_rojo                 : int  0 1 0 1 0 1 0 0 0 0 ...
##  $ num_oro_azul                     : Factor w/ 321 levels "30.7k","31.9k",..: 193 104 172 42 231 20
##  $ num_oro_rojo                     : Factor w/ 331 levels "28.6k","29.3k",..: 99 217 92 151 256 93
##  $ ban1_azul                        : Factor w/ 63 levels "Aatrox","Akali",..: 5 9 40 39 39 39 2 37
##  $ ban2_azul                        : Factor w/ 70 levels "Aatrox","Akali",..: 18 41 18 58 58 58 41
##  $ ban3_azul                        : Factor w/ 74 levels "Aatrox","Akali",..: 45 66 5 30 49 49 66 5
##  $ ban4_azul                        : Factor w/ 92 levels "Aatrox","Ahri",..: 43 34 52 62 62 30 46 4
##  $ ban5_azul                        : Factor w/ 99 levels "Aatrox","Akali",..: 73 76 49 19 82 65 24
##  $ ban1_rojo                        : Factor w/ 50 levels "Aatrox","Akali",..: 31 25 31 36 36 36 25
##  $ ban2_rojo                        : Factor w/ 64 levels "Akali","Aphelios",..: 37 38 37 38 38 38 4
##  $ ban3_rojo                        : Factor w/ 74 levels "Aatrox","Akali",..: 10 22 10 33 18 40 43
##  $ ban4_rojo                        : Factor w/ 92 levels "Aatrox","Akali",..: 29 41 29 8 8 26 38 24
##  $ ban5_rojo                        : Factor w/ 101 levels "Aatrox","Akali",..: 78 73 3 55 34 10 55
##  $ pick1_azul                       : Factor w/ 42 levels "Aatrox","Akali",..: 40 40 40 7 26 4 31 24
##  $ pick2_azul                       : Factor w/ 32 levels "Diana","Ekko",..: 17 16 17 17 16 16 8 16
##  $ pick3_azul                       : Factor w/ 46 levels "Akali","Azir",..: 46 2 38 8 37 8 8 17 2 3
##  $ pick4_azul                       : Factor w/ 35 levels "Aatrox","Aphelios",..: 29 18 9 12 6 29 7
##  $ pick5_azul                       : Factor w/ 24 levels "Alistar","Bard",..: 1 8 4 21 4 21 1 8 8 1
##  $ pick1_rojo                       : Factor w/ 49 levels "Aatrox","Akali",..: 23 34 41 34 3 47 25 3
##  $ pick2_rojo                       : Factor w/ 31 levels "Ekko","Elise",..: 14 6 14 6 15 6 23 6 6 1
##  $ pick3_rojo                       : Factor w/ 49 levels "Ahri","Akali",..: 11 16 35 40 16 40 37 49
##  $ pick4_rojo                       : Factor w/ 30 levels "Aphelios","Ashe",..: 13 29 3 13 6 6 27 6
##  $ pick5_rojo                       : Factor w/ 28 levels "Alistar","Annie",..: 16 1 9 9 9 1 17 13 1
##  $ top_azul                         : Factor w/ 43 levels "369","Acce","allorim",..: 7 5 7 39 39 39
##  $ jng_azul                         : Factor w/ 44 levels "AHaHaCiK","Akaadian",..: 1 20 1 3 3 3 20
##  $ mid_azul                         : Factor w/ 46 levels "Abbedagge","Ace",..: 30 7 30 40 40 40 7 1
##  $ adc_azul                         : Factor w/ 50 levels "Altec","Apollo",..: 17 48 17 37 37 37 48
##  $ sup_azul                         : Factor w/ 51 levels "Aphromoo","Bang",..: 39 42 39 19 19 19 42
##  $ top_rojo                         : Factor w/ 42 levels "369","Acce","allorim",..: 5 7 5 23 23 23
##  $ jng_rojo                         : Factor w/ 43 levels "AHaHaCiK","Akaadian",..: 19 1 19 26 26 26
##  $ mid_rojo                         : Factor w/ 45 levels "Abbedagge","Ace",..: 7 30 7 43 43 43 18 1
##  $ adc_rojo                         : Factor w/ 49 levels "Altec","Apollo",..: 48 18 48 28 28 28 26
##  $ sup_rojo                         : Factor w/ 49 levels "Aphromoo","BeryL",..: 37 34 37 28 28 28 4
##  $ kills_top_azul                   : int  2 5 4 0 5 2 5 2 1 1 ...
##  $ deaths_top_azul                  : int  3 5 2 3 4 2 3 2 2 2 ...
##  $ assists_top_azul                 : int  9 3 6 2 9 2 15 8 14 9 ...
##  $ kills_jng_azul                   : int  8 2 4 1 4 2 5 2 7 5 ...
##  $ deaths_jng_azul                  : int  1 5 3 1 2 3 3 3 1 0 ...
##  $ assists_jng_azul                 : int  8 6 10 2 13 1 7 7 10 5 ...
##  $ kills_mid_azul                   : int  3 1 7 0 2 0 2 7 4 3 ...
##  $ deaths_mid_azul                  : int  1 6 3 1 2 2 3 2 1 2 ...
##  $ assists_mid_azul                 : int  13 5 5 2 14 2 11 2 7 8 ...
##  $ kills_adc_azul                   : int  8 3 2 2 7 0 6 2 9 4 ...
##  $ deaths_adc_azul                  : int  3 4 2 2 4 5 1 2 2 1 ...
##  $ assists_adc_azul                 : int  7 5 11 1 8 0 10 10 5 5 ...
##  $ kills_sup_azul                   : int  0 0 0 0 0 0 4 1 0 0 ...
##  $ deaths_sup_azul                  : int  4 4 0 2 5 2 4 0 2 2 ...
##  $ assists_sup_azul                 : int  13 5 13 3 16 2 14 12 9 13 ...
##  $ kills_top_rojo                   : int  1 3 0 3 0 3 7 2 4 2 ...
##  $ deaths_top_rojo                  : int  4 3 2 0 5 2 4 4 5 2 ...
```

```
##  $ assists_top_rojo                : int  6 5 5 5 14 3 7 4 2 3 ...
##  $ kills_jng_rojo                  : int  5 6 2 3 1 4 0 2 3 2 ...
##  $ deaths_jng_rojo                 : int  2 2 5 0 3 1 5 2 3 1 ...
##  $ assists_jng_rojo                : int  5 14 6 2 10 6 6 5 2 5 ...
##  $ kills_mid_rojo                  : int  3 11 3 2 7 2 4 4 1 1 ...
##  $ deaths_mid_rojo                 : int  4 2 3 1 4 0 3 1 4 4 ...
##  $ assists_mid_rojo                : int  5 3 5 4 5 9 6 2 5 3 ...
##  $ kills_adc_rojo                  : int  3 4 5 1 7 4 2 1 0 2 ...
##  $ deaths_adc_rojo                 : int  7 2 3 1 3 0 6 1 4 3 ...
##  $ assists_adc_rojo                : int  8 11 3 6 8 4 6 2 5 2 ...
##  $ kills_sup_rojo                  : int  0 0 0 0 2 1 1 0 0 0 ...
##  $ deaths_sup_rojo                 : int  4 2 4 1 3 1 4 6 5 3 ...
##  $ assists_sup_rojo                : int  10 14 6 5 10 8 9 4 3 6 ...
##  $ summoner_1_top_azul             : Factor w/ 6 levels "","Boost","Dot",..: 4 4 4 6 6 6 4 4 4 4 .
##  $ summoner_2_top_azul             : Factor w/ 4 levels "","Dot","Flash",..: 4 4 4 3 3 3 4 4 4 4 .
##  $ summoner_1_jng_azul             : Factor w/ 5 levels "","Dot","Flash",..: 5 3 5 3 3 3 5 5 5 5 .
##  $ summoner_2_jng_azul             : Factor w/ 5 levels "","Dot","Flash",..: 3 5 3 5 5 5 3 3 3 3 .
##  $ summoner_1_mid_azul             : Factor w/ 8 levels "","Boost","Dot",..: 8 5 8 8 8 8 5 7 5 8 .
##  $ summoner_2_mid_azul             : Factor w/ 9 levels "","Barrier","Boost",..: 6 9 6 6 6 6 3 6 9
##  $ summoner_1_adc_azul             : Factor w/ 8 levels "","Barrier","Boost",..: 6 6 6 7 8 3 6 6 6
##  $ summoner_2_adc_azul             : Factor w/ 8 levels "","Barrier","Boost",..: 7 8 7 6 6 6 8 3 8
##  $ summoner_1_sup_azul             : Factor w/ 5 levels "","Dot","Exhaust",..: 2 4 3 4 4 4 4 3 4 2
##   [list output truncated]
```

```r
# Comprobar que las dimensiones del dataframe son correctas, 614 filasx517columnas
dim(lol)
```

```
## [1] 614 517
```

```r
# Comprobar los tipos de las variables
sapply(lol,class)
```

```
##                          torneo                           parte
##                        "factor"                         "factor"
##                           fecha                          semana
##                        "factor"                         "factor"
##                          tiempo                          parche
##                        "factor"                         "factor"
##                     nombre_azul                      nombre_rojo
##                        "factor"                         "factor"
##                      gana_azul                        gana_rojo
##                       "integer"                        "integer"
##             num_asesinatos_azul              num_asesinatos_rojo
##                       "integer"                        "integer"
##                 primera_sangre                 num_torres_azul
##                       "factor"                        "integer"
##                 num_torres_rojo                  primera_torre
##                       "integer"                        "factor"
##               num_dragones_azul          num_dragones_viento_azul
##                       "integer"                        "integer"
##      num_dragones_infierno_azul          num_dragones_oceano_azul
##                       "integer"                        "integer"
##      num_dragones_montaÃ.a_azul                num_dragones_rojo
```

```
##                      "integer"                      "integer"
##          num_dragones_viento_rojo       num_dragones_infierno_rojo
##                      "integer"                      "integer"
##          num_dragones_oceano_rojo       num_dragones_montaÃ.a_rojo
##                      "integer"                      "integer"
##                num_nashors_azul               num_nashors_rojo
##                      "integer"                      "integer"
##                   num_oro_azul                   num_oro_rojo
##                       "factor"                       "factor"
##                      ban1_azul                      ban2_azul
##                       "factor"                       "factor"
##                      ban3_azul                      ban4_azul
##                       "factor"                       "factor"
##                      ban5_azul                      ban1_rojo
##                       "factor"                       "factor"
##                      ban2_rojo                      ban3_rojo
##                       "factor"                       "factor"
##                      ban4_rojo                      ban5_rojo
##                       "factor"                       "factor"
##                     pick1_azul                     pick2_azul
##                       "factor"                       "factor"
##                     pick3_azul                     pick4_azul
##                       "factor"                       "factor"
##                     pick5_azul                     pick1_rojo
##                       "factor"                       "factor"
##                     pick2_rojo                     pick3_rojo
##                       "factor"                       "factor"
##                     pick4_rojo                     pick5_rojo
##                       "factor"                       "factor"
##                       top_azul                       jng_azul
##                       "factor"                       "factor"
##                       mid_azul                       adc_azul
##                       "factor"                       "factor"
##                       sup_azul                       top_rojo
##                       "factor"                       "factor"
##                       jng_rojo                       mid_rojo
##                       "factor"                       "factor"
##                       adc_rojo                       sup_rojo
##                       "factor"                       "factor"
##                  kills_top_azul                deaths_top_azul
##                       "integer"                      "integer"
##                assists_top_azul                 kills_jng_azul
##                       "integer"                      "integer"
##                 deaths_jng_azul                assists_jng_azul
##                       "integer"                      "integer"
##                  kills_mid_azul                deaths_mid_azul
##                       "integer"                      "integer"
##                assists_mid_azul                 kills_adc_azul
##                       "integer"                      "integer"
##                 deaths_adc_azul                assists_adc_azul
##                       "integer"                      "integer"
##                  kills_sup_azul                deaths_sup_azul
##                       "integer"                      "integer"
##                assists_sup_azul                  kills_top_rojo
```

```
##                    "integer"                    "integer"
##          deaths_top_rojo              assists_top_rojo
##                    "integer"                    "integer"
##           kills_jng_rojo               deaths_jng_rojo
##                    "integer"                    "integer"
##         assists_jng_rojo                kills_mid_rojo
##                    "integer"                    "integer"
##          deaths_mid_rojo              assists_mid_rojo
##                    "integer"                    "integer"
##           kills_adc_rojo               deaths_adc_rojo
##                    "integer"                    "integer"
##         assists_adc_rojo                kills_sup_rojo
##                    "integer"                    "integer"
##          deaths_sup_rojo              assists_sup_rojo
##                    "integer"                    "integer"
##       summoner_1_top_azul          summoner_2_top_azul
##                   "factor"                     "factor"
##       summoner_1_jng_azul          summoner_2_jng_azul
##                   "factor"                     "factor"
##       summoner_1_mid_azul          summoner_2_mid_azul
##                   "factor"                     "factor"
##       summoner_1_adc_azul          summoner_2_adc_azul
##                   "factor"                     "factor"
##       summoner_1_sup_azul          summoner_2_sup_azul
##                   "factor"                     "factor"
##       summoner_1_top_rojo          summoner_2_top_rojo
##                   "factor"                     "factor"
##       summoner_1_jng_rojo          summoner_2_jng_rojo
##                   "factor"                     "factor"
##       summoner_1_mid_rojo          summoner_2_mid_rojo
##                   "factor"                     "factor"
##       summoner_1_adc_rojo          summoner_2_adc_rojo
##                   "factor"                     "factor"
##       summoner_1_sup_rojo          summoner_2_sup_rojo
##                   "factor"                     "factor"
##              css_top_azul                 css_jng_azul
##                 "integer"                    "integer"
##              css_mid_azul                 css_adc_azul
##                 "integer"                    "integer"
##              css_sup_azul                 css_top_rojo
##                 "integer"                    "integer"
##              css_jng_rojo                 css_mid_rojo
##                 "integer"                    "integer"
##              css_adc_rojo                 css_sup_rojo
##                 "integer"                    "integer"
##       wards_destroyed_azul         wards_destroyed_rojo
##                   "integer"                    "integer"
##          wards_placed_azul            wards_placed_rojo
##                   "integer"                    "integer"
##          jng_share_15_azul            jng_share_15_rojo
##                   "factor"                     "factor"
##             jng_share_azul               jng_share_rojo
##                   "factor"                     "factor"
##        diferencia_oro_azul          diferencia_oro_rojo
```

```
##                         "factor"                         "factor"
##                         oro_azul                         oro_rojo
##                         "factor"                         "factor"
##                     gold_top_azul                    gold_jng_azul
##                         "factor"                         "factor"
##                     gold_mid_azul                    gold_adc_azul
##                         "factor"                         "factor"
##                     gold_sup_azul                    gold_top_rojo
##                         "factor"                         "factor"
##                     gold_jng_rojo                    gold_mid_rojo
##                         "factor"                         "factor"
##                     gold_adc_rojo                    gold_sup_rojo
##                         "factor"                         "factor"
##                     heraldos_azul                    heraldos_rojo
##                        "integer"                        "integer"
##                       inhibs_azul                      inhibs_rojo
##                        "integer"                        "integer"
##                      First_Blood      Total_Damage_Dealt_azul_top
##                         "factor"                         "integer"
##       Total_Damage_Dealt_azul_jng      Total_Damage_Dealt_azul_mid
##                         "integer"                         "integer"
##       Total_Damage_Dealt_azul_adc      Total_Damage_Dealt_azul_sup
##                         "integer"                         "integer"
##       Total_Damage_Dealt_rojo_top      Total_Damage_Dealt_rojo_jng
##                         "integer"                         "integer"
##       Total_Damage_Dealt_rojo_mid      Total_Damage_Dealt_rojo_adc
##                         "integer"                         "integer"
##       Total_Damage_Dealt_rojo_sup   Physical_Damage_Dealt_azul_top
##                         "integer"                         "integer"
##    Physical_Damage_Dealt_azul_jng   Physical_Damage_Dealt_azul_mid
##                         "integer"                         "integer"
##    Physical_Damage_Dealt_azul_adc   Physical_Damage_Dealt_azul_sup
##                         "integer"                         "integer"
##    Physical_Damage_Dealt_rojo_top   Physical_Damage_Dealt_rojo_jng
##                         "integer"                         "integer"
##    Physical_Damage_Dealt_rojo_mid   Physical_Damage_Dealt_rojo_adc
##                         "integer"                         "integer"
##    Physical_Damage_Dealt_rojo_sup      Magic_Damage_Dealt_azul_top
##                         "integer"                         "integer"
##       Magic_Damage_Dealt_azul_jng      Magic_Damage_Dealt_azul_mid
##                         "integer"                         "integer"
##       Magic_Damage_Dealt_azul_adc      Magic_Damage_Dealt_azul_sup
##                         "integer"                         "integer"
##       Magic_Damage_Dealt_rojo_top      Magic_Damage_Dealt_rojo_jng
##                         "integer"                         "integer"
##       Magic_Damage_Dealt_rojo_mid      Magic_Damage_Dealt_rojo_adc
##                         "integer"                         "integer"
##       Magic_Damage_Dealt_rojo_sup       True_Damage_Dealt_azul_top
##                         "integer"                         "integer"
##        True_Damage_Dealt_azul_jng       True_Damage_Dealt_azul_mid
##                         "integer"                         "integer"
##        True_Damage_Dealt_azul_adc       True_Damage_Dealt_azul_sup
##                         "integer"                         "integer"
##        True_Damage_Dealt_rojo_top       True_Damage_Dealt_rojo_jng
```

```
##                          "integer"                          "integer"
##         True_Damage_Dealt_rojo_mid         True_Damage_Dealt_rojo_adc
##                          "integer"                          "integer"
##         True_Damage_Dealt_rojo_sup   Total_Damage_Objectives_azul_top
##                          "integer"                          "integer"
##   Total_Damage_Objectives_azul_jng   Total_Damage_Objectives_azul_mid
##                          "integer"                          "integer"
##   Total_Damage_Objectives_azul_adc   Total_Damage_Objectives_azul_sup
##                          "integer"                          "integer"
##   Total_Damage_Objectives_rojo_top   Total_Damage_Objectives_rojo_jng
##                          "integer"                          "integer"
##   Total_Damage_Objectives_rojo_mid   Total_Damage_Objectives_rojo_adc
##                          "integer"                          "integer"
##   Total_Damage_Objectives_rojo_sup             Damage_Taken_azul_top
##                          "integer"                          "integer"
##              Damage_Taken_azul_jng             Damage_Taken_azul_mid
##                          "integer"                          "integer"
##              Damage_Taken_azul_adc             Damage_Taken_azul_sup
##                          "integer"                          "integer"
##              Damage_Taken_rojo_top             Damage_Taken_rojo_jng
##                          "integer"                          "integer"
##              Damage_Taken_rojo_mid             Damage_Taken_rojo_adc
##                          "integer"                          "integer"
##              Damage_Taken_rojo_sup    Physical_Damage_Taken_azul_top
##                          "integer"                          "integer"
##     Physical_Damage_Taken_azul_jng    Physical_Damage_Taken_azul_mid
##                          "integer"                          "integer"
##     Physical_Damage_Taken_azul_adc    Physical_Damage_Taken_azul_sup
##                          "integer"                          "integer"
##     Physical_Damage_Taken_rojo_top    Physical_Damage_Taken_rojo_jng
##                          "integer"                          "integer"
##     Physical_Damage_Taken_rojo_mid    Physical_Damage_Taken_rojo_adc
##                          "integer"                          "integer"
##     Physical_Damage_Taken_rojo_sup       Magic_Damage_Taken_azul_top
##                          "integer"                          "integer"
##        Magic_Damage_Taken_azul_jng       Magic_Damage_Taken_azul_mid
##                          "integer"                          "integer"
##        Magic_Damage_Taken_azul_adc       Magic_Damage_Taken_azul_sup
##                          "integer"                          "integer"
##        Magic_Damage_Taken_rojo_top       Magic_Damage_Taken_rojo_jng
##                          "integer"                          "integer"
##        Magic_Damage_Taken_rojo_mid       Magic_Damage_Taken_rojo_adc
##                          "integer"                          "integer"
##        Magic_Damage_Taken_rojo_sup        True_Damage_Taken_azul_top
##                          "integer"                          "integer"
##         True_Damage_Taken_azul_jng        True_Damage_Taken_azul_mid
##                          "integer"                          "integer"
##         True_Damage_Taken_azul_adc        True_Damage_Taken_azul_sup
##                          "integer"                          "integer"
##         True_Damage_Taken_rojo_top        True_Damage_Taken_rojo_jng
##                          "integer"                          "integer"
##         True_Damage_Taken_rojo_mid        True_Damage_Taken_rojo_adc
##                          "integer"                          "integer"
##         True_Damage_Taken_rojo_sup          cs_in_jung_team_azul_top
```

```
##                      "integer"                      "integer"
##          cs_in_jung_team_azul_jng          cs_in_jung_team_azul_mid
##                      "integer"                      "integer"
##          cs_in_jung_team_azul_adc          cs_in_jung_team_azul_sup
##                      "integer"                      "integer"
##          cs_in_jung_team_rojo_top          cs_in_jung_team_rojo_jng
##                      "integer"                      "integer"
##          cs_in_jung_team_rojo_mid          cs_in_jung_team_rojo_adc
##                      "integer"                      "integer"
##          cs_in_jung_team_rojo_sup        cs_in_jung_enemy_azul_top
##                      "integer"                      "integer"
##         cs_in_jung_enemy_azul_jng        cs_in_jung_enemy_azul_mid
##                      "integer"                      "integer"
##         cs_in_jung_enemy_azul_adc        cs_in_jung_enemy_azul_sup
##                      "integer"                      "integer"
##         cs_in_jung_enemy_rojo_top        cs_in_jung_enemy_rojo_jng
##                      "integer"                      "integer"
##         cs_in_jung_enemy_rojo_mid        cs_in_jung_enemy_rojo_adc
##                      "integer"                      "integer"
##         cs_in_jung_enemy_rojo_sup                  CSM_azul_top
##                      "integer"                       "factor"
##                  CSM_azul_jng                  CSM_azul_mid
##                       "factor"                       "factor"
##                  CSM_azul_adc                  CSM_azul_sup
##                       "factor"                       "factor"
##                  CSM_rojo_top                  CSM_rojo_jng
##                       "factor"                       "factor"
##                  CSM_rojo_mid                  CSM_rojo_adc
##                       "factor"                       "factor"
##                  CSM_rojo_sup                Golds_azul_top
##                       "factor"                      "integer"
##                Golds_azul_jng                Golds_azul_mid
##                      "integer"                      "integer"
##                Golds_azul_adc                Golds_azul_sup
##                      "integer"                      "integer"
##                Golds_rojo_top                Golds_rojo_jng
##                      "integer"                      "integer"
##                Golds_rojo_mid                Golds_rojo_adc
##                      "integer"                      "integer"
##                Golds_rojo_sup                  GPM_azul_top
##                      "integer"                      "integer"
##                  GPM_azul_jng                  GPM_azul_mid
##                      "integer"                      "integer"
##                  GPM_azul_adc                  GPM_azul_sup
##                      "integer"                      "integer"
##                  GPM_rojo_top                  GPM_rojo_jng
##                      "integer"                      "integer"
##                  GPM_rojo_mid                  GPM_rojo_adc
##                      "integer"                      "integer"
##                  GPM_rojo_sup                 GOLD_azul_top
##                      "integer"                       "factor"
##                 GOLD_azul_jng                 GOLD_azul_mid
##                       "factor"                       "factor"
##                 GOLD_azul_adc                 GOLD_azul_sup
```

```
##                       "factor"                       "factor"
##                  GOLD_rojo_top                  GOLD_rojo_jng
##                       "factor"                       "factor"
##                  GOLD_rojo_mid                  GOLD_rojo_adc
##                       "factor"                       "factor"
##                  GOLD_rojo_sup            Vision_Score_azul_top
##                       "factor"                      "integer"
##           Vision_Score_azul_jng            Vision_Score_azul_mid
##                      "integer"                      "integer"
##           Vision_Score_azul_adc            Vision_Score_azul_sup
##                      "integer"                      "integer"
##           Vision_Score_rojo_top            Vision_Score_rojo_jng
##                      "integer"                      "integer"
##           Vision_Score_rojo_mid            Vision_Score_rojo_adc
##                      "integer"                      "integer"
##           Vision_Score_rojo_sup            Wards_placed_azul_top
##                      "integer"                      "integer"
##           Wards_placed_azul_jng            Wards_placed_azul_mid
##                      "integer"                      "integer"
##           Wards_placed_azul_adc            Wards_placed_azul_sup
##                      "integer"                      "integer"
##           Wards_placed_rojo_top            Wards_placed_rojo_jng
##                      "integer"                      "integer"
##           Wards_placed_rojo_mid            Wards_placed_rojo_adc
##                      "integer"                      "integer"
##           Wards_placed_rojo_sup       Wards_destroyed_azul_top
##                      "integer"                      "integer"
##          Wards_destroyed_azul_jng       Wards_destroyed_azul_mid
##                      "integer"                      "integer"
##          Wards_destroyed_azul_adc       Wards_destroyed_azul_sup
##                      "integer"                      "integer"
##          Wards_destroyed_rojo_top       Wards_destroyed_rojo_jng
##                      "integer"                      "integer"
##          Wards_destroyed_rojo_mid       Wards_destroyed_rojo_adc
##                      "integer"                      "integer"
##          Wards_destroyed_rojo_sup         Control_Wards_azul_top
##                      "integer"                      "integer"
##            Control_Wards_azul_jng         Control_Wards_azul_mid
##                      "integer"                      "integer"
##            Control_Wards_azul_adc         Control_Wards_azul_sup
##                      "integer"                      "integer"
##            Control_Wards_rojo_top         Control_Wards_rojo_jng
##                      "integer"                      "integer"
##            Control_Wards_rojo_mid         Control_Wards_rojo_adc
##                      "integer"                      "integer"
##            Control_Wards_rojo_sup                   VS._azul_top
##                      "integer"                       "factor"
##                     VS._azul_jng                   VS._azul_mid
##                         "factor"                       "factor"
##                     VS._azul_adc                   VS._azul_sup
##                         "factor"                       "factor"
##                     VS._rojo_top                   VS._rojo_jng
##                         "factor"                       "factor"
##                     VS._rojo_mid                   VS._rojo_adc
```

```
##                           "factor"                           "factor"
##                       VS._rojo_sup     Total_damage_Champios_azul_top
##                           "factor"                          "integer"
##     Total_damage_Champios_azul_jng     Total_damage_Champios_azul_mid
##                          "integer"                          "integer"
##     Total_damage_Champios_azul_adc     Total_damage_Champios_azul_sup
##                          "integer"                          "integer"
##     Total_damage_Champios_rojo_top     Total_damage_Champios_rojo_jng
##                          "integer"                          "integer"
##     Total_damage_Champios_rojo_mid     Total_damage_Champios_rojo_adc
##                          "integer"                          "integer"
##     Total_damage_Champios_rojo_sup Physical_Damage_Champions_azul_top
##                          "integer"                          "integer"
## Physical_Damage_Champions_azul_jng Physical_Damage_Champions_azul_mid
##                          "integer"                          "integer"
## Physical_Damage_Champions_azul_adc Physical_Damage_Champions_azul_sup
##                          "integer"                          "integer"
## Physical_Damage_Champions_rojo_top Physical_Damage_Champions_rojo_jng
##                          "integer"                          "integer"
## Physical_Damage_Champions_rojo_mid Physical_Damage_Champions_rojo_adc
##                          "integer"                          "integer"
## Physical_Damage_Champions_rojo_sup    Magic_Damage_Champions_azul_top
##                          "integer"                          "integer"
##    Magic_Damage_Champions_azul_jng    Magic_Damage_Champions_azul_mid
##                          "integer"                          "integer"
##    Magic_Damage_Champions_azul_adc    Magic_Damage_Champions_azul_sup
##                          "integer"                          "integer"
##    Magic_Damage_Champions_rojo_top    Magic_Damage_Champions_rojo_jng
##                          "integer"                          "integer"
##    Magic_Damage_Champions_rojo_mid    Magic_Damage_Champions_rojo_adc
##                          "integer"                          "integer"
##    Magic_Damage_Champions_rojo_sup     True_Damage_Champions_azul_top
##                          "integer"                          "integer"
##     True_Damage_Champions_azul_jng     True_Damage_Champions_azul_mid
##                          "integer"                          "integer"
##     True_Damage_Champions_azul_adc     True_Damage_Champions_azul_sup
##                          "integer"                          "integer"
##     True_Damage_Champions_rojo_top     True_Damage_Champions_rojo_jng
##                          "integer"                          "integer"
##     True_Damage_Champions_rojo_mid     True_Damage_Champions_rojo_adc
##                          "integer"                          "integer"
##     True_Damage_Champions_rojo_sup                      DPM_azul_top
##                          "integer"                         "integer"
##                       DPM_azul_jng                      DPM_azul_mid
##                          "integer"                         "integer"
##                       DPM_azul_adc                      DPM_azul_sup
##                          "integer"                         "integer"
##                       DPM_rojo_top                      DPM_rojo_jng
##                          "integer"                         "integer"
##                       DPM_rojo_mid                      DPM_rojo_adc
##                          "integer"                         "integer"
##                       DPM_rojo_sup                      DMG_azul_top
##                          "integer"                          "factor"
##                       DMG_azul_jng                      DMG_azul_mid
```

```
##                      "factor"                      "factor"
##                 DMG_azul_adc                 DMG_azul_sup
##                      "factor"                      "factor"
##                 DMG_rojo_top                 DMG_rojo_jng
##                      "factor"                      "factor"
##                 DMG_rojo_mid                 DMG_rojo_adc
##                      "factor"                      "factor"
##                 DMG_rojo_sup           Solo_kills_azul_top
##                      "factor"                    "integer"
##           Solo_kills_azul_jng           Solo_kills_azul_mid
##                    "integer"                    "integer"
##           Solo_kills_azul_adc           Solo_kills_azul_sup
##                    "integer"                    "integer"
##           Solo_kills_rojo_top           Solo_kills_rojo_jng
##                    "integer"                    "integer"
##           Solo_kills_rojo_mid           Solo_kills_rojo_adc
##                    "integer"                    "integer"
##           Solo_kills_rojo_sup         Double_kills_azul_top
##                    "integer"                    "integer"
##         Double_kills_azul_jng         Double_kills_azul_mid
##                    "integer"                    "integer"
##         Double_kills_azul_adc         Double_kills_azul_sup
##                    "integer"                    "integer"
##         Double_kills_rojo_top         Double_kills_rojo_jng
##                    "integer"                    "integer"
##         Double_kills_rojo_mid         Double_kills_rojo_adc
##                    "integer"                    "integer"
##         Double_kills_rojo_sup         Triple_kills_azul_top
##                    "integer"                    "integer"
##         Triple_kills_azul_jng         Triple_kills_azul_mid
##                    "integer"                    "integer"
##         Triple_kills_azul_adc         Triple_kills_azul_sup
##                    "integer"                    "integer"
##         Triple_kills_rojo_top         Triple_kills_rojo_jng
##                    "integer"                    "integer"
##         Triple_kills_rojo_mid         Triple_kills_rojo_adc
##                    "integer"                    "integer"
##         Triple_kills_rojo_sup         Quadra_kills_azul_top
##                    "integer"                    "integer"
##         Quadra_kills_azul_jng         Quadra_kills_azul_mid
##                    "integer"                    "integer"
##         Quadra_kills_azul_adc         Quadra_kills_azul_sup
##                    "integer"                    "integer"
##         Quadra_kills_rojo_top         Quadra_kills_rojo_jng
##                    "integer"                    "integer"
##         Quadra_kills_rojo_mid         Quadra_kills_rojo_adc
##                    "integer"                    "integer"
##         Quadra_kills_rojo_sup          Penta_kills_azul_top
##                    "integer"                    "integer"
##          Penta_kills_azul_jng          Penta_kills_azul_mid
##                    "integer"                    "integer"
##          Penta_kills_azul_adc          Penta_kills_azul_sup
##                    "integer"                    "integer"
##          Penta_kills_rojo_top          Penta_kills_rojo_jng
```

```
##                     "integer"                     "integer"
##          Penta_kills_rojo_mid          Penta_kills_rojo_adc
##                     "integer"                     "integer"
##          Penta_kills_rojo_sup               CSD.15_azul_top
##                     "integer"                     "integer"
##               CSD.15_azul_jng               CSD.15_azul_mid
##                     "integer"                     "integer"
##               CSD.15_azul_adc               CSD.15_azul_sup
##                     "integer"                     "integer"
##               CSD.15_rojo_top               CSD.15_rojo_jng
##                     "integer"                     "integer"
##               CSD.15_rojo_mid               CSD.15_rojo_adc
##                     "integer"                     "integer"
##               CSD.15_rojo_sup               XPD.15_azul_top
##                     "integer"                     "integer"
##               XPD.15_azul_jng               XPD.15_azul_mid
##                     "integer"                     "integer"
##               XPD.15_azul_adc               XPD.15_azul_sup
##                     "integer"                     "integer"
##               XPD.15_rojo_top               XPD.15_rojo_jng
##                     "integer"                     "integer"
##               XPD.15_rojo_mid               XPD.15_rojo_adc
##                     "integer"                     "integer"
##               XPD.15_rojo_sup              LVLD.15_azul_top
##                     "integer"                     "integer"
##              LVLD.15_azul_jng              LVLD.15_azul_mid
##                     "integer"                     "integer"
##              LVLD.15_azul_adc              LVLD.15_azul_sup
##                     "integer"                     "integer"
##              LVLD.15_rojo_top              LVLD.15_rojo_jng
##                     "integer"                     "integer"
##              LVLD.15_rojo_mid              LVLD.15_rojo_adc
##                     "integer"                     "integer"
##              LVLD.15_rojo_sup        Damage_towers_azul_top
##                     "integer"                     "integer"
##        Damage_towers_azul_jng        Damage_towers_azul_mid
##                     "integer"                     "integer"
##        Damage_towers_azul_adc        Damage_towers_azul_sup
##                     "integer"                     "integer"
##        Damage_towers_rojo_top        Damage_towers_rojo_jng
##                     "integer"                     "integer"
##        Damage_towers_rojo_mid        Damage_towers_rojo_adc
##                     "integer"                     "integer"
##        Damage_towers_rojo_sup                 heal_azul_top
##                     "integer"                     "integer"
##                heal_azul_jng                 heal_azul_mid
##                     "integer"                     "integer"
##                heal_azul_adc                 heal_azul_sup
##                     "integer"                     "integer"
##                heal_rojo_top                 heal_rojo_jng
##                     "integer"                     "integer"
##                heal_rojo_mid                 heal_rojo_adc
##                     "integer"                     "integer"
##                heal_rojo_sup                ccing_azul_top
```

```
##                       "integer"                       "integer"
##                  ccing_azul_jng                  ccing_azul_mid
##                       "integer"                       "integer"
##                  ccing_azul_adc                  ccing_azul_sup
##                       "integer"                       "integer"
##                  ccing_rojo_top                  ccing_rojo_jng
##                       "integer"                       "integer"
##                  ccing_rojo_mid                  ccing_rojo_adc
##                       "integer"                       "integer"
##                  ccing_rojo_sup
##                       "integer"
```

2. Selección de los datos.

Como tenemos muchas variables, tenemos que eliminar aquellas que nos vayamos a usar o que no sean muy importantes o supongan información que se recoge en otras variables.

Variables no necesarias.

De primeras, podemos eliminar la fecha, parche, nombre de equipos, nombre de jugadores o campeones o los summoners utilizados. Aunque estos aspectos son importantes para ganar, como por ejemplo jugar un campeón agresivo o defensivo, los eliminamos del conjunto de datos puesto que son factores difíciles de analizar en un análisis estadístico debido a la cantidad de factores que tiene.

Antes de eliminar el equipo hay que cambiar el valor de las variables primera_sangre y primera_torre, para identificar si la obtuvo el equipo rojo o el azul.

```
lol$primera_sangre <- as.factor(ifelse(as.character(lol$primera_sangre) == as.character(lol$nombre_azul)
lol$primera_torre <- as.factor(ifelse(as.character(lol$primera_torre) == as.character(lol$nombre_azul),

eliminar <- c('torneo','parte', 'fecha','semana', 'parche','nombre_rojo','nombre_azul', 'top_azul', 'jng

lol<-lol %>% select(-(all_of(eliminar)))
lol<-lol %>% select(-(contains("ban")| contains("pick")| contains("summoner")))

head(lol) %>% select(c("primera_sangre", "primera_torre"))
```

```
##   primera_sangre primera_torre
## 1           rojo          azul
## 2           rojo          rojo
## 3           azul          azul
## 4           rojo          rojo
## 5           rojo          rojo
## 6           azul          rojo
```

Variables duplicadas

Además, en el dataset hay información duplicada respecto a algunas variables, como por ejemplo el oro o los subditos asesinados. Es importante mencionar que siempre que se puede, se intenta trabajar con el valor de la variable por minuto, puesto que no sirve de nada si un equipo consigue mucho oro pero simplemente es porque pierde la partida y esta partida es muy larga.

A continuación voy seleccionando los datos que me interesan del dataset respecto a diferentes aspectos de la partida, para reducir el dataset y obtener los mismos datos solo de una manera.

Creeps (Subditos)

Para los subditos, tenemos la variable CSM_posición_equipo que queremos mantener, ya que representa los subditos por minuto, el resto de variables respecto a los subditos se eliminan. Por una parte css_posicion_equipo es la misma información sin tener encuenta el tiempo. Por otra parte, las variables cs_in_jung_team y cs_in_jung_enemy, son variables muy específicas, que actualmente no vamos a usar en los análisis y que se encuentran ya sus valores dentro de la variable CSM_posicion_equipo. Respecto a las variables de CSD.15_equipo_posicion, que representan la diferencia de cs al minuto 15 para cada posición de los equipos, no la selecciono, porque la diferencia de cs en media partida, aunque es importante, está incluida en los CS por minuto.

```
lol<-lol %>% select(-(contains("css_")| contains("cs_in_jung")| contains("CSD.15")))
head(lol) %>% select(contains("CSM"))
```

```
##   CSM_azul_top CSM_azul_jng CSM_azul_mid CSM_azul_adc CSM_azul_sup CSM_rojo_top
## 1          7.4          7.3          9.1          7.3            1          7.8
## 2          7.5          6.8            8          6.7          1.2          7.7
## 3          8.2          6.6          8.1          9.6          0.9          8.3
## 4          7.6          6.8          8.4          9.9          1.2          9.7
## 5          6.2          5.9          7.3          7.2            1          8.9
## 6            8          6.6          9.2          7.8          1.4          7.6
##   CSM_rojo_jng CSM_rojo_mid CSM_rojo_adc CSM_rojo_sup
## 1          6.2          7.4          7.7          1.3
## 2          6.5          9.3          7.7          0.9
## 3          6.5          6.9          9.3          1.2
## 4          7.2          9.1          9.6          1.1
## 5          6.7          9.2          8.5          0.9
## 6          7.2          8.7          9.6          1.1
```

Asesinatos, Muertes y Asistencias.

La información de asesinatos, muertes y asistencias se puede condensar en una métrica que se suele utilizar en el juego, que es el KDA, este KDA se compone por (Asesinatos+Asistencias)/Muertes, de tal manera que generamos un KDA para cada jugador de la partida, condensando la información de Kills, Deaths y Assists de cada jugador en una variable.

Por otra parte, aunque el desempeño de un jugador es muy importante de estudiar para ver si el equipo gana o pierde, ya que puede ser que solo un jugador haga que un equipo gane, en estos análisis nos vamos a centrar más en estudios respecto al desempeño del equipo, mirando solo en pocas variables clave el desempeño de los jugadores. Es por esto, que mientras que conservamos los KDA para cada jugador, la información de solokills, doblekills, triplekills, cuadrakills y pentakills, las vamos a agrupar en una por equipo.

```
lol$kda_top_azul <- (lol$kills_top_azul+lol$assists_top_azul)/lol$deaths_top_azul
lol$kda_jng_azul <- (lol$kills_jng_azul+lol$assists_jng_azul)/lol$deaths_jng_azul
lol$kda_mid_azul <- (lol$kills_mid_azul+lol$assists_mid_azul)/lol$deaths_mid_azul
lol$kda_adc_azul <- (lol$kills_adc_azul+lol$assists_adc_azul)/lol$deaths_adc_azul
lol$kda_sup_azul <- (lol$kills_sup_azul+lol$assists_sup_azul)/lol$deaths_sup_azul

lol$kda_top_rojo <- (lol$kills_top_rojo+lol$assists_top_rojo)/lol$deaths_top_rojo
```

```r
lol$kda_jng_rojo <- (lol$kills_jng_rojo+lol$assists_jng_rojo)/lol$deaths_jng_rojo
lol$kda_mid_rojo <- (lol$kills_mid_rojo+lol$assists_mid_rojo)/lol$deaths_mid_rojo
lol$kda_adc_rojo <- (lol$kills_adc_rojo+lol$assists_adc_rojo)/lol$deaths_adc_rojo
lol$kda_sup_rojo <- (lol$kills_sup_rojo+lol$assists_sup_rojo)/lol$deaths_sup_rojo

lol$solo_k_azul <- lol %>% select(contains("Solo_kills_azul"))%>% rowSums(na.rm = TRUE)
lol$double_k_azul <- lol %>% select(contains("Double_kills_azul"))%>% rowSums(na.rm = TRUE)
lol$triple_k_azul <- lol %>% select(contains("Triple_kills_azul"))%>% rowSums(na.rm = TRUE)
lol$quadra_k_azul <- lol %>% select(contains("Quadra_kills_azul"))%>% rowSums(na.rm = TRUE)
lol$penta_k_azul <- lol %>% select(contains("Penta_kills_azul"))%>% rowSums(na.rm = TRUE)
lol$solo_k_rojo <- lol %>% select(contains("Solo_kills_rojo"))%>% rowSums(na.rm = TRUE)
lol$double_k_rojo <- lol %>% select(contains("Double_kills_rojo"))%>% rowSums(na.rm = TRUE)
lol$triple_k_rojo <- lol %>% select(contains("Triple_kills_rojo"))%>% rowSums(na.rm = TRUE)
lol$quadra_k_rojo <- lol %>% select(contains("Quadra_kills_rojo"))%>% rowSums(na.rm = TRUE)
lol$penta_k_rojo <- lol %>% select(contains("Penta_kills_rojo"))%>% rowSums(na.rm = TRUE)


lol<-lol %>% select(-(contains("kills")| contains("deaths")| contains("assists")))
head(lol) %>% select(contains("kda")| contains("_k_"))
```

```
##   kda_top_azul kda_jng_azul kda_mid_azul kda_adc_azul kda_sup_azul kda_top_rojo
## 1    3.6666667    16.000000           16         5.00         3.25     1.750000
## 2    1.6000000     1.600000            1         2.00         1.25     2.666667
## 3    5.0000000     4.666667            4         6.50          Inf     2.500000
## 4    0.6666667     3.000000            2         1.50         1.50          Inf
## 5    3.5000000     8.500000            8         3.75         3.20     2.800000
## 6    2.0000000     1.000000            1         0.00         1.00     3.000000
##   kda_jng_rojo kda_mid_rojo kda_adc_rojo kda_sup_rojo solo_k_azul double_k_azul
## 1     5.000000     2.000000     1.571429          2.5           1             4
## 2    10.000000     7.000000     7.500000          7.0           0             1
## 3     1.600000     2.666667     2.666667          1.5           2             4
## 4          Inf     6.000000     7.000000          5.0           0             0
## 5     3.666667     3.000000     5.000000          4.0           1             1
## 6    10.000000          Inf          Inf          9.0           0             0
##   triple_k_azul quadra_k_azul penta_k_azul solo_k_rojo double_k_rojo
## 1             1             0            0           0             1
## 2             0             0            0           3             1
## 3             0             0            0           0             0
## 4             0             0            0           0             1
## 5             1             0            0           0             0
## 6             0             0            0           0             1
##   triple_k_rojo quadra_k_rojo penta_k_rojo
## 1             0             0            0
## 2             2             0            0
## 3             0             0            0
## 4             0             0            0
## 5             1             1            0
## 6             0             0            0
```

Oro.

El oro es la estadística más importante del juego. Los asesinatos, los subditos, las torres, los monstruos de la jungla, todos los objetivos del juego cuando se consiguen te otorgan oro, por tanto, está claro que un equipo

que gana en la gran mayoría de partidas, tiene más oro que el equipo perdedor. Lo demuestro para eliminar la variable que se refiere al oro final del dataset.

```
lol$gana <- as.factor(ifelse(lol$gana_azul == 1, 'azul','rojo'))

require(stringr)
lol$num_oro_azul <- as.integer(str_replace(lol$num_oro_azul,'k',''))*1000
lol$num_oro_rojo <- as.integer(str_replace(lol$num_oro_rojo,'k',''))*1000
lol$mas_oro <- as.factor(ifelse(lol$num_oro_azul == lol$num_oro_rojo, 'iguales', ifelse(lol$num_oro_azul

table(lol$mas_oro,lol$gana)
```

```
##
##            azul rojo
##   iguales     1    2
##   mas_azul  319    5
##   mas_rojo    1  286
```

```
eliminar2 <- c('num_oro_azul','num_oro_rojo', 'mas_oro','gana_azul', 'gana_rojo')

lol<-lol %>% select(-(all_of(eliminar2)))
```

Se observa perfectamente que es cierto, y que posiblemente los pocos casos donde el equipo con menos oro gana, es una partida muy igualada. Por tanto, como hacer una correlación o un modelo de regresión con la variable de oro al final de la partida sería un poco trampa, puesto que ya sabemos que hay una gran correlación entre el oro y ganar la partida, la metodología típica para analizar el oro de los equipos se suele medir por el oro en el minuto 15 de los equipos. En este minuto, las partidas no están decididas, pero si podemos ver si hay un equipo que ha dominado claramente los primeros minutos de la partida. Por tanto, vamos a trabajar con el oro del equipo en el minuto 15, y con la diferencia de oro en estos minutos.

El oro de cada jugador no lo vamos a usar, porque aunque sea muy util, realmente este oro se consigue con los objetivos que ya se ven representados en el dataset, asique no hace falta tener información de oro por cada jugador. Por tanto eliminamos todas las variables ya sean temporales o de final de partida, que hacen referencia al oro. Eso sí, vamos a usar la variable GPM_equipo_posicion para detectar cual fue el jugador con más oro por equipo y nos quedamos con esta posición como jugador más valioso del equipo. Generado una nueva variable.

```
oro_azul<-str_split(lol$oro_azul, ',')
lol$oro_al_15_azul<-unlist(lapply(oro_azul, function(x) as.integer(str_replace(unlist(x)[3], " '15 min'

oro_rojo<-str_split(lol$oro_rojo, ',')
lol$oro_al_15_rojo<-unlist(lapply(oro_rojo, function(x) as.integer(str_replace(unlist(x)[3], " '15 min'

# la diferencia será positiva si es para el azul o negativa para el rojo
diferencia_oro<-str_split(lol$diferencia_oro_azul, ',')
lol$diff_oro_al_15<-unlist(lapply(diferencia_oro, function(x) as.integer(str_replace(unlist(x)[3], " '15

lol<-lol %>% mutate(MasValioso_azul = case_when(
    (GPM_azul_top >= GPM_azul_jng) & (GPM_azul_top >= GPM_azul_mid) & (GPM_azul_top >= GPM_azul_adc) &
    (GPM_azul_jng >= GPM_azul_top) & (GPM_azul_jng >= GPM_azul_mid) & (GPM_azul_jng >= GPM_azul_adc) &
    (GPM_azul_mid >= GPM_azul_top) & (GPM_azul_mid >= GPM_azul_jng) & (GPM_azul_mid >= GPM_azul_adc) &
    (GPM_azul_adc >= GPM_azul_top) & (GPM_azul_adc >= GPM_azul_jng) & (GPM_azul_adc >= GPM_azul_mid) &
    TRUE ~ 'sup'
```

```
  )
)
lol<-lol %>% mutate(MasValioso_rojo = case_when(
    (GPM_rojo_top >= GPM_rojo_jng) & (GPM_rojo_top >= GPM_rojo_mid) & (GPM_rojo_top >= GPM_rojo_adc) &
    (GPM_rojo_jng >= GPM_rojo_top) & (GPM_rojo_jng >= GPM_rojo_mid) & (GPM_rojo_jng >= GPM_rojo_adc) &
    (GPM_rojo_mid >= GPM_rojo_top) & (GPM_rojo_mid >= GPM_rojo_jng) & (GPM_rojo_mid >= GPM_rojo_adc) &
    (GPM_rojo_adc >= GPM_rojo_top) & (GPM_rojo_adc >= GPM_rojo_jng) & (GPM_rojo_adc >= GPM_rojo_mid) &
    TRUE ~ 'sup'
  )
)



eliminar3 <- c('oro_azul','oro_rojo', 'diferencia_oro_azul','diferencia_oro_rojo')

lol<-lol %>% select(-(all_of(eliminar3)))
lol<-lol %>% select(-(contains("Golds")| contains("GPM")| contains("gold")))
head(lol) %>% select(c("oro_al_15_azul", "oro_al_15_rojo", "diff_oro_al_15", "MasValioso_azul", "MasVali
```

```
##   oro_al_15_azul oro_al_15_rojo diff_oro_al_15 MasValioso_azul MasValioso_rojo
## 1          25400          24200           1200             mid             adc
## 2          23200          25800          -2600             adc             mid
## 3          25500          22400           3100             mid             adc
## 4          23700          26200          -2500             adc             mid
## 5          22900          24800          -1900             adc             mid
## 6          23800          24100           -346             top             mid
```

Vision

En el juego se consigue visión el mapa gracias a la visión, está claro que cuanto más tiempo pasa, más wards se colocan en el mapa y se destruyen, por tanto las variables tienen que estar estandarizadas por el tiempo de la partida.

Debido a que se pueden poner wards normales o de control de visión, además que es importante no solo poner los wards sino eliminarlos, se creó la métrica Vision Score o VS. Esta métrica representa una puntuación que recoje wards colocados, eliminados y la calidad de la visión que otorgan los wards. Por tanto, en el estudio vamos a usar solo la puntuación de Vision Score, eliminando el resto de variables. Además, aunque puede ser útil saber el desempeño de cada jugador en la visión del equipo, debido a que no vamos a realizar grandes análisis sobre la visión en el trabajo, agrupamos la vision score por equipos, generando una vision score para el equipo azul y otra para el rojo.

```
fecha <- str_split(lol$tiempo,':')
lol$tiempo<-unlist(lapply(fecha, function(x) as.integer(unlist(x)[1])))

lol<-lol %>% mutate(ScoreVision_azul = (Vision_Score_azul_top + Vision_Score_azul_jng + Vision_Score_azu
lol<-lol %>% mutate(ScoreVision_rojo = (Vision_Score_rojo_top + Vision_Score_rojo_jng + Vision_Score_ro

lol<-lol %>% select(-(contains("Wards")| contains("Vision_Score")| contains("VS")))
head(lol) %>% select(contains("ScoreVision"))
```

```
##   ScoreVision_azul ScoreVision_rojo
## 1         7.290323         5.612903
```

```
## 2         5.687500         8.687500
## 3         8.300000         5.366667
## 4         6.428571         7.428571
## 5         6.378378         8.243243
## 6         5.750000         6.708333
```

Daño

Respecto al daño tenemos muchas variables. Tenemos información sobre el daño total realizado y de que tipo de daño es, el daño recibido, el daño realizado a objetivos, el daño a campeones enemigos, el daño por minuto a enemigos, el porcentaje de daño por persona en cada equipo, el daño a torres, etc.

De nuevo, aunque es importante el daño por cada jugador, es obvio que eso son muchas variables y por tanto, la única variable que vamos a mantener por jugador es la de daño por minuto (DPM). Como este daño es daño a campeones, podemos eliminar las variables que hacen referencia a daño a campeones total y porcentaje de daño. La información de daño mágico, físico o verdadero a campeones que tenemos por jugador, la condensamos en equipo y la dividimos por minuto, para saber la cantidad de cada tipo a campeones. Además, el daño realizado a objetivos se combina por equipos y se divide por minuto. Respecto al daño total y daño recibido,se eliminan las variables porque el daño total no es tan importante, sino que nos centramos en el daño en campeones y objetivos y porque el daño recibido por un equipo es inverso al daño realizado por el otro y si ya tenemos los daños realizados por equipo no tiene sentido tener el recibido.

Por último, el daño a torres se recoge obviamente en el número de torres eliminadas, por tanto se puede eliminar la variable.

```
lol<-lol %>% mutate(PhysicalDamageChampions_azul = (Physical_Damage_Champions_azul_top + Physical_Damage
lol<-lol %>% mutate(PhysicalDamageChampions_rojo = (Physical_Damage_Champions_rojo_top + Physical_Damage
lol<-lol %>% mutate(MagicDamageChampions_azul = (Magic_Damage_Champions_azul_top + Magic_Damage_Champion
lol<-lol %>% mutate(MagicDamageChampions_rojo = (Magic_Damage_Champions_rojo_top + Magic_Damage_Champion
lol<-lol %>% mutate(TrueDamageChampions_azul = (True_Damage_Champions_azul_top + True_Damage_Champions_a
lol<-lol %>% mutate(TrueDamageChampions_rojo = (True_Damage_Champions_rojo_top + True_Damage_Champions_r
lol<-lol %>% mutate(DamageObjectives_azul = (Total_Damage_Objectives_azul_top + Total_Damage_Objectives_
lol<-lol %>% mutate(DamageObjectives_rojo = (Total_Damage_Objectives_rojo_top + Total_Damage_Objectives_

lol<-lol %>% select(-(contains("Damage_Dealt")| contains("Damage_Objectives")| contains("Damage_Taken")
head(lol) %>% select(contains("DPM") | contains( "Damage"))
```

```
##   DPM_azul_top DPM_azul_jng DPM_azul_mid DPM_azul_adc DPM_azul_sup DPM_rojo_top
## 1          392          470          696          553          206          425
## 2          388          551          411          601          111          320
## 3          386          500          962          500          164          290
## 4          460          270          154          333           46          324
## 5          545          531          567          584          174          370
## 6          534          239          209          149           78          390
##   DPM_rojo_jng DPM_rojo_mid DPM_rojo_adc DPM_rojo_sup
## 1          569          345          473          144
## 2          428          546          683          117
## 3          430          355          590          123
## 4          232          258          412           77
## 5          383          580         1009          123
## 6          315          391          526          107
##   PhysicalDamageChampions_azul PhysicalDamageChampions_rojo
## 1                     931.6129                     491.4839
## 2                     991.5938                     752.9688
```

```
## 3                    1402.5333                    983.0333
## 4                     645.8571                    923.6071
## 5                    1055.1892                   1015.4865
## 6                     482.9167                   1038.5417
##   MagicDamageChampions_azul MagicDamageChampions_rojo TrueDamageChampions_azul
## 1                 1198.5484                 1189.9032                234.58065
## 2                  888.7812                 1340.7500                241.28125
## 3                  812.4667                  589.2000                354.00000
## 4                  595.6786                  371.3929                 64.57143
## 5                 1121.4595                 1227.9730                270.91892
## 6                  480.3333                  679.5833                291.66667
##   TrueDamageChampions_rojo DamageObjectives_azul DamageObjectives_rojo
## 1                313.96774             3067.7419              664.5161
## 2                 59.65625             1134.3750             3012.5000
## 3                256.93333             3260.0000              456.6667
## 4                 52.67857              714.2857             3246.4286
## 5                269.21622             2837.8378             1597.2973
## 6                 76.87500             1700.0000             2283.3333
```

Experiencia y niveles

Para la información de experiencia y niveles en el juego, ocurre como con el oro, que evidentemente cuando un equipo gana la partida, tiene más nivel y experiencia que el contrario. Por tanto, se trabaja con las variables al minuto 15. Concretamente tenemos la diferencia de experiencia y niveles. Mantenemos ambas variables porque mientras que tener un nivel de ventaja puede suponer mucho en el minuto 15, realmente no sabemos si esto es porque se tiene una pequeña ventaja y justo al 15 había un jugador superior al otro, o porque la diferencia entre jugadores es muy grande realmente. Como trabajamos con diferencia de experiencia entre cada jugador de cada posición para los equipos, tendremos solo una variable, que será positiva si la ventaja la tiene el equipo azul y negativa si la tiene el rojo.

```
lol<-lol %>% mutate(diferencia_exp = XPD.15_azul_top + XPD.15_azul_jng + XPD.15_azul_mid + XPD.15_azul_a
lol<-lol %>% mutate(diferencia_nivel = LVLD.15_azul_top + LVLD.15_azul_jng + LVLD.15_azul_mid + LVLD.15_

lol<-lol %>% select(-(contains("XPD.15")| contains("LVLD.15")))
head(lol) %>% select(contains("exp")| contains( "nivel"))
```

```
##   diferencia_exp diferencia_nivel
## 1            -39               0
## 2          -1816              -2
## 3           1544               0
## 4           1527               0
## 5          -2507              -1
## 6           -815               1
```

Curaciones y control de campeones.

Las curaciones se pueden realizar en el juego por diversas maneras, ya sean robo de vida, campeones que curan o con pociones. Por tanto se mantiene la variable pero se agrupa y convierte a curación por minuto. El control de campeones es vital en muchas partidas, consiste en el tiempo que un campeon enemigo es inmovilizado o ralentizado por un campeon. Igual que las curaciones, agrupamos por equipo y se convierte a cc por minuto.

```
lol<-lol %>% mutate(cura_azul = (heal_azul_top + heal_azul_jng + heal_azul_mid + heal_azul_adc + heal_a
lol<-lol %>% mutate(cura_rojo = (heal_rojo_top + heal_rojo_jng + heal_rojo_mid + heal_rojo_adc + heal_r
lol<-lol %>% mutate(cc_azul = (ccing_azul_top + ccing_azul_jng + ccing_azul_mid + ccing_azul_adc + ccin
lol<-lol %>% mutate(cc_rojo = (ccing_rojo_top + ccing_rojo_jng + ccing_rojo_mid + ccing_rojo_adc + ccin

lol<-lol %>% select(-(contains("ccing")| contains("heal")))
head(lol) %>% select(contains("cura") | contains( "cc"))
```

```
##   cura_azul cura_rojo  cc_azul  cc_rojo
## 1 1413.0323  699.7419 3.709677 3.677419
## 2  978.5625 1195.9375 3.093750 2.625000
## 3 1112.7333  535.9333 2.633333 3.766667
## 4  811.1786  780.2500 1.107143 2.142857
## 5 1098.2432 1500.1081 3.702703 2.513514
## 6  610.0000  775.2083 3.000000 3.041667
```

Porcentaje de jungla.

Las variables jng_share_15 y jng_share, representan el porcentaje de monstruos de la jungla que ha tenido un equipo respecto al otro. Evidentemente, ambas variables son complementarias para cada equipo, por lo que podemos quedarnos solo con una, que sea el extra de porcentaje de jungla que ha tenido el equipo azul respecto al rojo. Si el valor es positivo es que el azul ha matado más monstruos de la jungla y si es negativo es el rojo el equipo que ha matado más monstruos de la jungla.

```
lol<-lol %>% mutate(ventaja_jung_15 = as.numeric(as.character(jng_share_15_azul)) - as.numeric(as.chara
lol<-lol %>% mutate(ventaja_jung = as.numeric(as.character(jng_share_azul)) - as.numeric(as.character(j

lol<-lol %>% select(-(contains("share")))
head(lol) %>% select(contains("ventaja"))
```

```
##   ventaja_jung_15 ventaja_jung
## 1            -4.2         19.0
## 2            19.0         -0.6
## 3            10.9          5.0
## 4            10.4         -0.4
## 5           -10.6        -11.4
## 6            -5.2         -2.6
```

3. Limpieza de los datos

Como primer paso de la limpieza de los datos, tenemos que mirar el tipo de los diferentes atributos del dataset.

```
sapply(lol, class)
```

```
##                      tiempo          num_asesinatos_azul
##                   "integer"                    "integer"
##          num_asesinatos_rojo               primera_sangre
##                   "integer"                     "factor"
##             num_torres_azul              num_torres_rojo
```

```
##                   "integer"                   "integer"
##               primera_torre           num_dragones_azul
##                    "factor"                   "integer"
##    num_dragones_viento_azul  num_dragones_infierno_azul
##                   "integer"                   "integer"
##    num_dragones_oceano_azul  num_dragones_montaÃ.a_azul
##                   "integer"                   "integer"
##           num_dragones_rojo    num_dragones_viento_rojo
##                   "integer"                   "integer"
##  num_dragones_infierno_rojo    num_dragones_oceano_rojo
##                   "integer"                   "integer"
##  num_dragones_montaÃ.a_rojo            num_nashors_azul
##                   "integer"                   "integer"
##            num_nashors_rojo               heraldos_azul
##                   "integer"                   "integer"
##               heraldos_rojo                 inhibs_azul
##                   "integer"                   "integer"
##                 inhibs_rojo                First_Blood
##                   "integer"                    "factor"
##                CSM_azul_top                CSM_azul_jng
##                    "factor"                    "factor"
##                CSM_azul_mid                CSM_azul_adc
##                    "factor"                    "factor"
##                CSM_azul_sup                CSM_rojo_top
##                    "factor"                    "factor"
##                CSM_rojo_jng                CSM_rojo_mid
##                    "factor"                    "factor"
##                CSM_rojo_adc                CSM_rojo_sup
##                    "factor"                    "factor"
##                DPM_azul_top                DPM_azul_jng
##                   "integer"                   "integer"
##                DPM_azul_mid                DPM_azul_adc
##                   "integer"                   "integer"
##                DPM_azul_sup                DPM_rojo_top
##                   "integer"                   "integer"
##                DPM_rojo_jng                DPM_rojo_mid
##                   "integer"                   "integer"
##                DPM_rojo_adc                DPM_rojo_sup
##                   "integer"                   "integer"
##                kda_top_azul                kda_jng_azul
##                   "numeric"                   "numeric"
##                kda_mid_azul                kda_adc_azul
##                   "numeric"                   "numeric"
##                kda_sup_azul                kda_top_rojo
##                   "numeric"                   "numeric"
##                kda_jng_rojo                kda_mid_rojo
##                   "numeric"                   "numeric"
##                kda_adc_rojo                kda_sup_rojo
##                   "numeric"                   "numeric"
##                 solo_k_azul               double_k_azul
##                   "numeric"                   "numeric"
##               triple_k_azul               quadra_k_azul
##                   "numeric"                   "numeric"
##                penta_k_azul                 solo_k_rojo
```

```
##                   "numeric"                   "numeric"
##                double_k_rojo               triple_k_rojo
##                   "numeric"                   "numeric"
##                quadra_k_rojo                penta_k_rojo
##                   "numeric"                   "numeric"
##                        gana               oro_al_15_azul
##                    "factor"                    "integer"
##               oro_al_15_rojo              diff_oro_al_15
##                    "integer"                    "integer"
##             MasValioso_azul             MasValioso_rojo
##                "character"                 "character"
##            ScoreVision_azul            ScoreVision_rojo
##                   "numeric"                   "numeric"
## PhysicalDamageChampions_azul PhysicalDamageChampions_rojo
##                   "numeric"                   "numeric"
##    MagicDamageChampions_azul    MagicDamageChampions_rojo
##                   "numeric"                   "numeric"
##     TrueDamageChampions_azul     TrueDamageChampions_rojo
##                   "numeric"                   "numeric"
##        DamageObjectives_azul        DamageObjectives_rojo
##                   "numeric"                   "numeric"
##               diferencia_exp             diferencia_nivel
##                    "integer"                   "integer"
##                   cura_azul                   cura_rojo
##                   "numeric"                   "numeric"
##                     cc_azul                     cc_rojo
##                   "numeric"                   "numeric"
##              ventaja_jung_15                ventaja_jung
##                   "numeric"                   "numeric"
```

Vemos que todas las variables tienen el tipo correcto salvo las de subditos por minuto (CSM). Por tanto, hemos de corregir su tipo a numeric. También falla el tipo de las variables MasValiosoAzul y MasValiosoRojo, que aparecen como character y tienen que ser factores.

```
lol$CSM_azul_top <- as.numeric(as.character(lol$CSM_azul_top))
lol$CSM_azul_jng <- as.numeric(as.character(lol$CSM_azul_jng))
lol$CSM_azul_mid <- as.numeric(as.character(lol$CSM_azul_mid))
lol$CSM_azul_adc <- as.numeric(as.character(lol$CSM_azul_adc))
lol$CSM_azul_sup <- as.numeric(as.character(lol$CSM_azul_sup))
lol$CSM_rojo_top <- as.numeric(as.character(lol$CSM_rojo_top))
lol$CSM_rojo_jng <- as.numeric(as.character(lol$CSM_rojo_jng))
lol$CSM_rojo_mid <- as.numeric(as.character(lol$CSM_rojo_mid))
lol$CSM_rojo_adc <- as.numeric(as.character(lol$CSM_rojo_adc))
lol$CSM_rojo_sup <- as.numeric(as.character(lol$CSM_rojo_sup))

lol$MasValioso_azul<- as.factor(lol$MasValioso_azul)
lol$MasValioso_rojo<- as.factor(lol$MasValioso_rojo)
```

A continuación muestro las dimensiones del dataset y el str, sin limpiar.

```
# Comprobar que los datos se han cargado en el dataframe correctamente
str(lol)
```

```
## 'data.frame':    614 obs. of  88 variables:
```

```
##  $ tiempo                     : int  31 32 30 28 37 24 36 31 32 31 ...
##  $ num_asesinatos_azul        : int  21 11 17 3 18 4 22 14 21 13 ...
##  $ num_asesinatos_rojo        : int  12 24 10 9 17 14 14 9 8 7 ...
##  $ primera_sangre             : Factor w/ 2 levels "azul","rojo": 2 2 1 2 2 1 1 1 2 2 ...
##  $ num_torres_azul            : int  11 0 9 1 7 3 7 10 8 8 ...
##  $ num_torres_rojo            : int  1 11 2 11 7 10 4 1 2 3 ...
##  $ primera_torre              : Factor w/ 2 levels "azul","rojo": 1 2 1 2 2 2 1 1 1 2 ...
##  $ num_dragones_azul          : int  3 2 4 1 3 1 3 3 3 1 ...
##  $ num_dragones_viento_azul   : int  1 0 1 0 0 1 1 0 1 0 ...
##  $ num_dragones_infierno_azul : int  2 0 1 1 1 0 1 0 0 1 ...
##  $ num_dragones_oceano_azul   : int  0 1 0 0 1 0 1 0 1 0 ...
##  $ num_dragones_montaÃ.a_azul : int  0 1 2 0 1 0 0 3 1 0 ...
##  $ num_dragones_rojo          : int  1 3 0 4 2 2 3 2 1 3 ...
##  $ num_dragones_viento_rojo   : int  0 0 0 3 0 0 0 1 0 1 ...
##  $ num_dragones_infierno_rojo : int  0 3 0 0 0 0 0 1 0 1 ...
##  $ num_dragones_oceano_rojo   : int  1 0 0 0 0 1 3 0 0 1 ...
##  $ num_dragones_montaÃ.a_rojo : int  0 0 0 1 2 1 0 0 1 0 ...
##  $ num_nashors_azul           : int  2 0 2 0 2 0 0 1 1 1 ...
##  $ num_nashors_rojo           : int  0 1 0 1 0 1 0 0 0 0 ...
##  $ heraldos_azul              : int  2 1 2 0 0 2 1 2 1 2 ...
##  $ heraldos_rojo              : int  0 1 0 2 2 0 1 0 1 0 ...
##  $ inhibs_azul                : int  3 0 2 0 2 0 1 2 2 1 ...
##  $ inhibs_rojo                : int  0 3 0 2 1 2 0 0 0 0 ...
##  $ First_Blood                : Factor w/ 11 levels "","adc_azul",..: 3 7 6 5 7 10 4 6 11 5 ...
##  $ CSM_azul_top               : num  7.4 7.5 8.2 7.6 6.2 8 6.2 4.8 7.1 7 ...
##  $ CSM_azul_jng               : num  7.3 6.8 6.6 6.8 5.9 6.6 7.4 6.4 7.4 6.6 ...
##  $ CSM_azul_mid               : num  9.1 8 8.1 8.4 7.3 9.2 6.8 9.2 7.9 10.2 ...
##  $ CSM_azul_adc               : num  7.3 6.7 9.6 9.9 7.2 7.8 8 6.4 8.9 8.1 ...
##  $ CSM_azul_sup               : num  1 1.2 0.9 1.2 1 1.4 1 1.1 0.9 0.9 ...
##  $ CSM_rojo_top               : num  7.8 7.7 8.3 9.7 8.9 7.6 6.1 6.7 7.2 8.5 ...
##  $ CSM_rojo_jng               : num  6.2 6.5 6.5 7.2 6.7 7.2 5.2 6.2 6.8 7.6 ...
##  $ CSM_rojo_mid               : num  7.4 9.3 6.9 9.1 9.2 8.7 9.7 7.1 9.4 6.8 ...
##  $ CSM_rojo_adc               : num  7.7 7.7 9.3 9.6 8.5 9.6 7.6 7.5 7.4 9.1 ...
##  $ CSM_rojo_sup               : num  1.3 0.9 1.2 1.1 0.9 1.1 1 1.3 0.9 1.2 ...
##  $ DPM_azul_top               : int  392 388 386 460 545 534 471 225 439 402 ...
##  $ DPM_azul_jng               : int  470 551 500 270 531 239 455 541 507 390 ...
##  $ DPM_azul_mid               : int  696 411 962 154 567 209 402 734 414 582 ...
##  $ DPM_azul_adc               : int  553 601 500 333 584 149 796 455 588 734 ...
##  $ DPM_azul_sup               : int  206 111 164 46 174 78 218 140 90 135 ...
##  $ DPM_rojo_top               : int  425 320 290 324 370 390 601 287 546 388 ...
##  $ DPM_rojo_jng               : int  569 428 430 232 383 315 178 388 358 522 ...
##  $ DPM_rojo_mid               : int  345 546 355 258 580 391 526 704 662 211 ...
##  $ DPM_rojo_adc               : int  473 683 590 412 1009 526 495 486 195 287 ...
##  $ DPM_rojo_sup               : int  144 117 123 77 123 107 321 108 277 107 ...
##  $ kda_top_azul               : num  3.667 1.6 5 0.667 3.5 ...
##  $ kda_jng_azul               : num  16 1.6 4.67 3 8.5 ...
##  $ kda_mid_azul               : num  16 1 4 2 8 ...
##  $ kda_adc_azul               : num  5 2 6.5 1.5 3.75 0 16 6 7 9 ...
##  $ kda_sup_azul               : num  3.25 1.25 Inf 1.5 3.2 ...
##  $ kda_top_rojo               : num  1.75 2.67 2.5 Inf 2.8 ...
##  $ kda_jng_rojo               : num  5 10 1.6 Inf 3.67 ...
##  $ kda_mid_rojo               : num  2 7 2.67 6 3 ...
##  $ kda_adc_rojo               : num  1.57 7.5 2.67 7 5 ...
##  $ kda_sup_rojo               : num  2.5 7 1.5 5 4 ...
```

```
##  $ solo_k_azul                : num  1 0 2 0 1 0 1 0 1 0 ...
##  $ double_k_azul              : num  4 1 4 0 1 0 1 1 4 2 ...
##  $ triple_k_azul              : num  1 0 0 0 1 0 0 0 0 0 ...
##  $ quadra_k_azul              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ penta_k_azul               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ solo_k_rojo                : num  0 3 0 0 0 0 1 0 1 0 ...
##  $ double_k_rojo              : num  1 1 0 1 0 1 0 0 2 1 ...
##  $ triple_k_rojo              : num  0 2 0 0 1 0 1 0 0 0 ...
##  $ quadra_k_rojo              : num  0 0 0 0 1 0 0 0 0 0 ...
##  $ penta_k_rojo               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gana                       : Factor w/ 2 levels "azul","rojo": 1 2 1 2 2 2 1 1 1 1 ...
##  $ oro_al_15_azul             : int  25400 23200 25500 23700 22900 23800 25200 25300 24800 24000 ..
##  $ oro_al_15_rojo             : int  24200 25800 22400 26200 24800 24100 23100 22700 23000 23700 ..
##  $ diff_oro_al_15             : int  1200 -2600 3100 -2500 -1900 -346 2100 2600 1800 255 ...
##  $ MasValioso_azul            : Factor w/ 4 levels "adc","jng","mid",..: 3 1 3 1 1 4 1 3 1 3 ...
##  $ MasValioso_rojo            : Factor w/ 4 levels "adc","jng","mid",..: 1 3 1 3 3 3 3 3 3 4 ...
##  $ ScoreVision_azul           : num  7.29 5.69 8.3 6.43 6.38 ...
##  $ ScoreVision_rojo           : num  5.61 8.69 5.37 7.43 8.24 ...
##  $ PhysicalDamageChampions_azul: num  932 992 1403 646 1055 ...
##  $ PhysicalDamageChampions_rojo: num  491 753 983 924 1015 ...
##  $ MagicDamageChampions_azul  : num  1199 889 812 596 1121 ...
##  $ MagicDamageChampions_rojo  : num  1190 1341 589 371 1228 ...
##  $ TrueDamageChampions_azul   : num  234.6 241.3 354 64.6 270.9 ...
##  $ TrueDamageChampions_rojo   : num  314 59.7 256.9 52.7 269.2 ...
##  $ DamageObjectives_azul      : num  3068 1134 3260 714 2838 ...
##  $ DamageObjectives_rojo      : num  665 3012 457 3246 1597 ...
##  $ diferencia_exp             : int  -39 -1816 1544 1527 -2507 -815 2475 -1252 2733 -122 ...
##  $ diferencia_nivel           : int  0 -2 0 0 -1 1 2 -2 1 0 ...
##  $ cura_azul                  : num  1413 979 1113 811 1098 ...
##  $ cura_rojo                  : num  700 1196 536 780 1500 ...
##  $ cc_azul                    : num  3.71 3.09 2.63 1.11 3.7 ...
##  $ cc_rojo                    : num  3.68 2.62 3.77 2.14 2.51 ...
##  $ ventaja_jung_15            : num  -4.2 19 10.9 10.4 -10.6 ...
##  $ ventaja_jung               : num  19 -0.6 5 -0.4 -11.4 ...
```

```
# Comprobar que las dimensiones del dataframe son correctas, 614 filasx517columnas
dim(lol)
```

```
## [1] 614  88
```

3.1 Datos vacíos.

Vamos a estudiar si el conjunto de datos presenta elementos vacíos o ceros. En caso de que haya una variable igual a 0, esto puede ser porque el valor es realmente 0 o porque es un elemento perdido, igual que los elementos vacíos. Estos elementos vacíos por tanto, pueden aparacer como 0, NA o como un valor indicativo de que falta el valor como '-'.

En caso de que un valor sea 0 tenemos que identificar su causa. Para todos aquellos elementos vacíos tendremos que decidir como solucionar la falta de información. Por una parte, se puede eliminar la instancia, suponiendo una perdida de datos. Otra solución es dejar claro que falta ese dato con una etiqueta como por ejemplo 'Desconocido', esta solución puede ser efectiva sobre todo para las variables que son un factor. Una solución algo más interesante es sustituir el valor por una medida de tendencia central como la media o mediana en las variables numéricas, o la clase más utilizada en las variables categóricas. Por último,

se pueden imputar estos valores vacíos en función de los valores del conjunto de datos mediante métodos probabilistas. Esta última solución suele ser la mejor porque el valor no es el mismo para todas las instancias que se encuentran vacías.

Compruebo si los datos tienen elementos iguales a 0 o elementos vacíos:

```
# Compruebo los elementos que son 0
sapply(lol, function(x) sum(x==0, na.rm=T))
```

```
##                       tiempo          num_asesinatos_azul
##                            0                            2
##          num_asesinatos_rojo               primera_sangre
##                            6                            0
##              num_torres_azul              num_torres_rojo
##                           16                           29
##                primera_torre            num_dragones_azul
##                            0                           72
##     num_dragones_viento_azul   num_dragones_infierno_azul
##                          363                          351
##     num_dragones_oceano_azul   num_dragones_montaÃ.a_azul
##                          340                          369
##            num_dragones_rojo     num_dragones_viento_rojo
##                           86                          339
##   num_dragones_infierno_rojo     num_dragones_oceano_rojo
##                          331                          343
##   num_dragones_montaÃ.a_rojo            num_nashors_azul
##                          362                          298
##             num_nashors_rojo               heraldos_azul
##                          306                          127
##                heraldos_rojo                 inhibs_azul
##                          284                          266
##                  inhibs_rojo                 First_Blood
##                          309                            0
##                 CSM_azul_top                CSM_azul_jng
##                            0                            0
##                 CSM_azul_mid                CSM_azul_adc
##                            0                            0
##                 CSM_azul_sup                CSM_rojo_top
##                            0                            0
##                 CSM_rojo_jng                CSM_rojo_mid
##                            0                            0
##                 CSM_rojo_adc                CSM_rojo_sup
##                            0                            1
##                 DPM_azul_top                DPM_azul_jng
##                            0                            0
##                 DPM_azul_mid                DPM_azul_adc
##                            0                            0
##                 DPM_azul_sup                DPM_rojo_top
##                            0                            0
##                 DPM_rojo_jng                DPM_rojo_mid
##                            0                            0
##                 DPM_rojo_adc                DPM_rojo_sup
##                            0                            0
##                 kda_top_azul                kda_jng_azul
```

```
##                          17                           3
##                 kda_mid_azul                kda_adc_azul
##                          12                          11
##                 kda_sup_azul                kda_top_rojo
##                          10                          21
##                 kda_jng_rojo                kda_mid_rojo
##                          12                           9
##                 kda_adc_rojo                kda_sup_rojo
##                          17                          21
##                  solo_k_azul               double_k_azul
##                          322                         245
##                triple_k_azul               quadra_k_azul
##                          529                         600
##                 penta_k_azul                 solo_k_rojo
##                          611                         347
##                double_k_rojo               triple_k_rojo
##                          273                         529
##                quadra_k_rojo                penta_k_rojo
##                          602                         611
##                         gana              oro_al_15_azul
##                            0                           0
##               oro_al_15_rojo              diff_oro_al_15
##                            0                           0
##             MasValioso_azul             MasValioso_rojo
##                            0                           0
##            ScoreVision_azul            ScoreVision_rojo
##                            0                           0
## PhysicalDamageChampions_azul PhysicalDamageChampions_rojo
##                            0                           0
##    MagicDamageChampions_azul    MagicDamageChampions_rojo
##                            0                           0
##     TrueDamageChampions_azul     TrueDamageChampions_rojo
##                            0                           0
##        DamageObjectives_azul        DamageObjectives_rojo
##                            0                           0
##               diferencia_exp             diferencia_nivel
##                            0                         111
##                    cura_azul                   cura_rojo
##                            0                           0
##                     cc_azul                     cc_rojo
##                            0                           0
##              ventaja_jung_15                 ventaja_jung
##                           19                           9
```

Vemos que hay muchas variables con 0, pero tenemos que tener en cuenta que en todas estas variables es algo normal. Es muy posible que los asesinatos de un equipo o otro sean 0, lo mismo puede ocurrir con torres, dragones, nashors, heraldos, inhibidores, rachas de asesinatos o asesinatos en solitario. Todo esto ocurre porque un equipo puede ir muy mal y no conseguir asesinatos o objetivos, lo cual hace que su valor sea 0. Otras variables con 0 son la diferencia de nivel, y la ventaja en la jungla, las cuáles es posible también que sean 0 porque no haya diferencias entre ambos equipos. Las últimas variables a comentar que es lógico que tengan 0 son los kda de los jugadores, de nuevo, esto se explica porque un jugador ha muerto varias veces y no ha asesinado ni ayudado en nada. Se da en pocas ocasiones, pero en partidas a gran nivel es posible, ya que muchas veces un equipo es capaz de dominar toda la partida sin dar opciones al rival. Por último, se observa un 0 en CSM_rojo_sup, esto se puede deber a que un jugador no asesinó ningún súbdito en la

partida, aunque extraño, es posible y sobretodo en los support, se puede estar jugando algún support muy defensivo (como puede ser una soraka o yuumi) que no haya asesinado ningún súbdito.

Para estudiar porque puede ocurrir el valor de 0 en CSM_rojo_sup, accedemos al dataset lol_base y busco el campeón jugado para ver si es cierta mi suposición.

```
as.character(lol_base[which(lol_base$CSM_rojo_sup=='0'),]$pick5_rojo)
```

```
## [1] "Yuumi"
```

Efectivamente, vemos que el campeón que consiguió 0 de CSM era Yuumi, este campeón se caracteriza por que se sube a un compañero suyo y no ataca, sino que se dedica a protegerle, por lo que es bastante aceptable que tuviera 0 subditos en la partida.

Por tanto, mantenemos todos los 0 en el conjunto de datos, puesto que todos parecen ser valores lógicos y correctos.

A continuación estudio los valores vacíos del dataset.

```
sapply(lol, function(x) sum(is.na(x) | x==''))
```

```
##                       tiempo          num_asesinatos_azul
##                            0                            0
##          num_asesinatos_rojo                primera_sangre
##                            0                            0
##              num_torres_azul              num_torres_rojo
##                            0                            0
##                primera_torre            num_dragones_azul
##                            0                            0
##     num_dragones_viento_azul   num_dragones_infierno_azul
##                            0                            0
##     num_dragones_oceano_azul   num_dragones_montaÃ.a_azul
##                            0                            0
##            num_dragones_rojo     num_dragones_viento_rojo
##                            0                            0
##   num_dragones_infierno_rojo     num_dragones_oceano_rojo
##                            0                            0
##   num_dragones_montaÃ.a_rojo             num_nashors_azul
##                            0                            0
##             num_nashors_rojo                heraldos_azul
##                            0                            2
##                heraldos_rojo                  inhibs_azul
##                            2                            2
##                  inhibs_rojo                  First_Blood
##                            2                            2
##                 CSM_azul_top                 CSM_azul_jng
##                            0                            0
##                 CSM_azul_mid                 CSM_azul_adc
##                            0                            0
##                 CSM_azul_sup                 CSM_rojo_top
##                            0                            0
##                 CSM_rojo_jng                 CSM_rojo_mid
##                            0                            0
##                 CSM_rojo_adc                 CSM_rojo_sup
```

```
##                           0                           0
##                DPM_azul_top                DPM_azul_jng
##                           0                           0
##                DPM_azul_mid                DPM_azul_adc
##                           0                           0
##                DPM_azul_sup                DPM_rojo_top
##                           0                           0
##                DPM_rojo_jng                DPM_rojo_mid
##                           0                           0
##                DPM_rojo_adc                DPM_rojo_sup
##                           0                           0
##                kda_top_azul                kda_jng_azul
##                           1                           1
##                kda_mid_azul                kda_adc_azul
##                           0                           0
##                kda_sup_azul                kda_top_rojo
##                           0                           0
##                kda_jng_rojo                kda_mid_rojo
##                           1                           1
##                kda_adc_rojo                kda_sup_rojo
##                           5                           2
##                 solo_k_azul                double_k_azul
##                           0                           0
##                triple_k_azul                quadra_k_azul
##                           0                           0
##                 penta_k_azul                  solo_k_rojo
##                           0                           0
##                double_k_rojo                triple_k_rojo
##                           0                           0
##                quadra_k_rojo                 penta_k_rojo
##                           0                           0
##                        gana               oro_al_15_azul
##                           0                          45
##               oro_al_15_rojo               diff_oro_al_15
##                          45                          45
##              MasValioso_azul              MasValioso_rojo
##                           0                           0
##             ScoreVision_azul             ScoreVision_rojo
##                           1                           1
## PhysicalDamageChampions_azul PhysicalDamageChampions_rojo
##                           0                           0
##    MagicDamageChampions_azul    MagicDamageChampions_rojo
##                           0                           0
##     TrueDamageChampions_azul     TrueDamageChampions_rojo
##                           0                           0
##        DamageObjectives_azul        DamageObjectives_rojo
##                           2                           2
##               diferencia_exp             diferencia_nivel
##                           1                           1
##                   cura_azul                   cura_rojo
##                           1                           1
##                    cc_azul                     cc_rojo
##                           1                           1
##              ventaja_jung_15                 ventaja_jung
```

```
##                                0                                0
```

Vemos que hay varias variables con valores vacíos. Para todas estas variables podemos considerar que se debe a errores en el web scraping o que faltaba la información en la propia web. Por tanto, habrá que solucionar estos valores vacíos para las variables que hacen referencia a heraldos, inhibidores, oro al 15, puntuación de visión, daño a objetivos, diferencia de nivel o experiencia, valores de curación o de cc, o FirstBlood.

Eso sí quiero estudiar porque se produce el NA en las variables de kda de los jugadores, puesto que es un campo calculado por nosotros y es posible que haya un problema en el cálculo. Para esto miro en lol_base, los asesinatos, muertes y asistencias de aquellos jugadores que tienen NA en su kda. Para simplificarlo todo, solo miro los de kda_adc_rojo que es el que más NA tiene.

```
lol_base[which(is.na(lol$kda_adc_rojo)),]$kills_adc_rojo
```

```
## [1] 0 0 0 0 0
```

```
lol_base[which(is.na(lol$kda_adc_rojo)),]$deaths_adc_rojo
```

```
## [1] 0 0 0 0 0
```

```
lol_base[which(is.na(lol$kda_adc_rojo)),]$assists_adc_rojo
```

```
## [1] 0 0 0 0 0
```

Vemos que estos jugadores, no asesinaron, ni asistieron, ni fueron asesinados en la partida, por lo que el cálculo de su KDA es 0+0/0 que es NA, por tanto, realmente este valor de las variables KDA que es NA, tiene que ser sustituido por el valor 0.

Para el resto de variables tenemos que imputar los NA. Lo realizo mediante el método de kNN. Este método se basa en la similitud entre diferentes atributos del dataset, de manera que se puede estimar el valor de un atributo vacío en funcion de los atributos más parecidos a este.

```
lol$kda_top_azul <- ifelse(is.na(lol$kda_top_azul), 0, lol$kda_top_azul)
lol$kda_jng_azul <- ifelse(is.na(lol$kda_jng_azul), 0, lol$kda_jng_azul)
lol$kda_mid_azul <- ifelse(is.na(lol$kda_mid_azul), 0, lol$kda_mid_azul)
lol$kda_adc_azul <- ifelse(is.na(lol$kda_adc_azul), 0, lol$kda_adc_azul)
lol$kda_sup_azul <- ifelse(is.na(lol$kda_sup_azul), 0, lol$kda_sup_azul)

lol$kda_top_rojo <- ifelse(is.na(lol$kda_top_rojo), 0, lol$kda_top_rojo)
lol$kda_jng_rojo <- ifelse(is.na(lol$kda_jng_rojo), 0, lol$kda_jng_rojo)
lol$kda_mid_rojo <- ifelse(is.na(lol$kda_mid_rojo), 0, lol$kda_mid_rojo)
lol$kda_adc_rojo <- ifelse(is.na(lol$kda_adc_rojo), 0, lol$kda_adc_rojo)
lol$kda_sup_rojo <- ifelse(is.na(lol$kda_sup_rojo), 0, lol$kda_sup_rojo)

lol$First_Blood<-ifelse(lol$First_Blood=='',NA, as.character(lol$First_Blood))

# Consigo el nombre de las variables con nulos
nulls<-as.data.frame(sapply(lol, function(x) sum(is.na(x))))
colnames(nulls) <- 'nulls'
nulls$index <-rownames(nulls)
vars_with_nulls<-nulls[which(nulls$nulls!=0),]$index
```

```
lol_imputed<-kNN(lol, k=3)

# muestro que se han corregido los nulos.
sapply(lol_imputed[vars_with_nulls], function(x) sum(is.na(x)))
```

```
##         heraldos_azul         heraldos_rojo           inhibs_azul
##                     0                     0                     0
##           inhibs_rojo           First_Blood        oro_al_15_azul
##                     0                     0                     0
##        oro_al_15_rojo        diff_oro_al_15      ScoreVision_azul
##                     0                     0                     0
##      ScoreVision_rojo DamageObjectives_azul DamageObjectives_rojo
##                     0                     0                     0
##        diferencia_exp      diferencia_nivel             cura_azul
##                     0                     0                     0
##             cura_rojo               cc_azul               cc_rojo
##                     0                     0                     0
```

```
lol_imputed$First_Blood <- as.factor(as.character(lol_imputed$First_Blood))
```

3.2 Valores extremos.

Los valores extremos son datos que se encuentran tan alejados de los valores normales de una variable que hacen sospechar si estos valores son realmente válidos o si por el contrario se deben a un error en la recolección de los datos. En caso de observar que realmente los errores no son erróneos aunque si extremos, se pueden mantener los valores. Si se detectan que los valores extremos son debido a problemas y erróneos, entonces se tienen que corregir, ya sea mediante una transformación lógica (quizás se tengan que cambiar decimales) o mediante una imputación de los valores como si de valores vacíos se trataran.

Es importante destacar que los valores extremos, obviamente solo se pueden obtener para aquellas variables que son numéricas.

Primero vamos a estudiar los valores extremos para detectarlos:

```
lol_numericas <- lol_imputed %>% select(where(is.numeric) | where(is.integer))

sapply(lol_numericas, function(x) boxplot.stats(x)$out)
```

```
## $tiempo
##  [1] 48 51 47 50 49 56 50 49 47 61 47 47 49 47 48 50 47
##
## $num_asesinatos_azul
## [1] 37
##
## $num_asesinatos_rojo
## [1] 36
##
## $num_torres_azul
## integer(0)
##
## $num_torres_rojo
## integer(0)
```

```
##
## $num_dragones_azul
## integer(0)
##
## $num_dragones_viento_azul
##  [1] 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3
##
## $num_dragones_infierno_azul
##  [1] 3 3 3 3 3 4 3 3 3 3 3 3 3 3 4 3 3 3 4 3 3
##
## $num_dragones_oceano_azul
##  [1] 3 3 3 4 3 3 4 3 3 3 3 3 4 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3
##
## $num_dragones_montaÃ.a_azul
##  [1] 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3
##
## $num_dragones_rojo
## integer(0)
##
## $num_dragones_viento_rojo
##  [1] 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3
##
## $num_dragones_infierno_rojo
##  [1] 3 3 3 3 4 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3
##
## $num_dragones_oceano_rojo
##  [1] 3 3 3 3 4 3 3 3 3 3 3 3 4 4 3 3 3 3 3 3 3 3 3 4 4 3 3 3 3 3 3
##
## $num_dragones_montaÃ.a_rojo
##  [1] 3 3 3 3 3 3 3 4 3 3 3 3
##
## $num_nashors_azul
##  [1] 4 3 3 3 3 3 3 3 3 3
##
## $num_nashors_rojo
## [1] 3 3 3 3 3 3 3 3 3
##
## $heraldos_azul
## integer(0)
##
## $heraldos_rojo
## integer(0)
##
## $inhibs_azul
## integer(0)
##
## $inhibs_rojo
## [1] 6 6
##
## $CSM_azul_top
## [1]  4.8  4.8 10.6  3.1  4.8  4.4
##
## $CSM_azul_jng
## [1] 8.6 8.5 8.9 9.0 8.9 9.1 9.0 9.5
```

```
##
## $CSM_azul_mid
## [1] 13.2 12.0  5.2  5.4
##
## $CSM_azul_adc
## [1] 5.7 5.6 5.5 4.2 4.9 5.2 6.0 5.8 3.4
##
## $CSM_azul_sup
##  [1] 2.0 2.2 2.9 3.0 0.1 2.9 0.1 2.2 0.2 2.6 0.1 2.2 0.1 2.2 3.1 0.2 0.1 0.2 2.1
## [20] 2.6 0.2 2.1 0.1 2.0 0.1 0.1 0.1 0.2 0.2
##
## $CSM_rojo_top
## [1]  2.8  3.4  4.3 10.8  4.6
##
## $CSM_rojo_jng
## [1] 8.7 8.5 8.6 9.1
##
## $CSM_rojo_mid
## [1] 5.1 4.9
##
## $CSM_rojo_adc
##  [1]  6.1  4.2 12.1  4.9 12.5  3.6  6.1  6.2  5.9  5.5  5.7
##
## $CSM_rojo_sup
##  [1] 1.7 1.9 1.8 0.2 2.3 1.7 1.8 2.5 2.0 2.0 2.1 0.2 0.3 1.8 0.1 0.1 0.1 3.7 0.4
## [20] 0.3 1.7 1.9 1.8 0.4 0.1 0.1 2.7 0.1 2.1 0.2 0.3 0.2 0.3 1.9 0.2 0.3 0.4 0.0
## [39] 0.3
##
## $DPM_azul_top
##  [1] 858 861 956 874 883 935 865 854 960 857
##
## $DPM_azul_jng
##  [1]  666  685  643  739  710  639  767  648  660  748 1084  774  628  749  639
## [16]  784
##
## $DPM_azul_mid
##  [1]  962  983  991  980  954 1012 1059 1169 1477  950 1003 1073 1158  994 1362
## [16] 1021  967 1095  993 1106
##
## $DPM_azul_adc
## [1] 1074 1225 1101 1069 1068 1329 1222
##
## $DPM_azul_sup
##  [1] 369 329 320 319 366 383 472 559 457 434 311 801 434 354 426 450 429 338 417
## [20] 431 661 467 479 338 376 315 329
##
## $DPM_rojo_top
##  [1]  800  802  808 1005  795  846  881  885  817  834  921
##
## $DPM_rojo_jng
##  [1] 702 626 694 647 636 884 706 650 621 651 817 647 625 725 667 706 784 658
##
## $DPM_rojo_mid
## [1] 1116 1117 1029 1033 1119 1108 1016
```

```
##
## $DPM_rojo_adc
##  [1] 1009 1030 1232 1376 1168 1070 1077 1148 1173 1072 1105 1116
##
## $DPM_rojo_sup
##  [1] 321 328 402 360 324 420 456 296 350 348 436 310 329 380 338 374 416 301 308
## [20] 360 645 354 296 456 326 299 316 445
##
## $kda_top_azul
##  [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf  16 Inf Inf Inf Inf Inf Inf Inf
## [20]  17 Inf Inf Inf Inf Inf  18 Inf Inf Inf Inf Inf Inf Inf  17 Inf Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [58] Inf Inf Inf Inf Inf Inf Inf  21 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [77] Inf Inf Inf Inf
##
## $kda_jng_azul
##  [1] Inf Inf Inf Inf  21 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [20] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [58] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [77] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##
## $kda_mid_azul
##   [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [19] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [37] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [55] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [73] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [91] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [109] Inf Inf Inf Inf
##
## $kda_adc_azul
##   [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [19] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [37] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [55] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [73] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [91] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [109] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [127] Inf
##
## $kda_sup_azul
##  [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [20] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [58] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [77] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##
## $kda_top_rojo
##  [1] Inf  21  16 Inf Inf Inf Inf Inf Inf Inf  16 Inf Inf Inf Inf  15 Inf Inf  15
## [20] Inf Inf Inf Inf Inf Inf  15 Inf Inf Inf Inf Inf Inf Inf Inf  16 Inf Inf Inf
## [39] Inf Inf Inf Inf  15 Inf Inf Inf Inf Inf  15 Inf Inf Inf Inf Inf Inf Inf Inf
## [58] Inf  21  16 Inf Inf Inf Inf Inf Inf  16  15 Inf Inf Inf Inf  20 Inf Inf Inf
## [77] Inf Inf Inf
```

```
##
## $kda_jng_rojo
##  [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf  19 Inf Inf Inf Inf Inf Inf Inf Inf
## [20] Inf Inf  25 Inf Inf  22  19 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf  19 Inf Inf Inf Inf Inf Inf Inf Inf  22 Inf
## [58] Inf Inf Inf Inf Inf Inf
##
## $kda_mid_rojo
##  [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [20] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [58] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [77] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##
## $kda_adc_rojo
##   [1] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [19] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [37] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [55] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [73] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
##  [91] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [109] Inf Inf
##
## $kda_sup_rojo
##  [1] Inf  17 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
## [20] Inf Inf Inf Inf Inf Inf Inf Inf  17 Inf  21 Inf Inf Inf Inf  18 Inf Inf Inf
## [39] Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf  20 Inf Inf Inf Inf  17 Inf
## [58] Inf Inf Inf  18 Inf Inf  23 Inf Inf Inf Inf Inf  19 Inf  19 Inf Inf  19 Inf
## [77]  17 Inf Inf Inf  18 Inf
##
## $solo_k_azul
##  [1] 3 5 3 3 3 3 3 4 3 3 4 3 9 5 4 4 3 3 3 3 3 3 3 4 3 3 5 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 3
##
## $double_k_azul
## [1] 6 7
##
## $triple_k_azul
##  [1] 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1
## [39] 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1
## [77] 2 1 1 1 1 2 1 1 1
##
## $quadra_k_azul
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $penta_k_azul
## [1] 1 1 1
##
## $solo_k_rojo
##  [1] 3 3 4 3 5 3 3 7 3 3 3 4 5 4 4 3 3 3 3 3 3 4 3 3 4 3 3 3 3 3 3 3 3 5 3 3 3 6 3
## [39] 3
##
## $double_k_rojo
## [1] 6 6 6 6
```

```
##
## $triple_k_rojo
##  [1] 2 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1
## [77] 1 1 1 1 1 1 1 1 1
##
## $quadra_k_rojo
##  [1] 1 1 1 1 1 2 1 1 1 1 1 1
##
## $penta_k_rojo
## [1] 1 1 1
##
## $oro_al_15_azul
##  [1] 30200 28700 30100 29500 31500 28600 28700 20200 29700 29800 29200
##
## $oro_al_15_rojo
## [1] 28800 28800 29100 28800 29000 28500
##
## $diff_oro_al_15
## [1]  7200  7200  7100 -7800  7000
##
## $ScoreVision_azul
## [1] 10.73333 10.74286
##
## $ScoreVision_rojo
##  [1] 10.66667 10.84375 10.87879  3.24000  4.20000 11.12821 10.45946 10.40000
##  [9]  4.25000 10.58537 10.73333
##
## $PhysicalDamageChampions_azul
## [1] 1939.885 1827.333 1869.059 2194.600
##
## $PhysicalDamageChampions_rojo
##  [1] 1616.393 1669.615 1715.667 1922.071 1616.588 1960.314 1617.162 1796.571
##  [9] 1639.909 1905.279 1676.707 1816.894
##
## $MagicDamageChampions_azul
## [1] 1674.000 1640.548 1680.818 2378.088 1836.231
##
## $MagicDamageChampions_rojo
##  [1] 1618.556 1933.857 1757.725 1650.160 1743.195 1640.062 1633.463 1591.028
##  [9] 1848.278 1626.209
##
## $TrueDamageChampions_azul
##  [1] 354.0000 384.6552 338.2759 343.8235 331.4103 426.1562 337.8108 322.8333
##  [9] 329.6222 306.1064 346.2857 347.4000 333.6667 317.6500 321.7429 351.6667
## [17] 341.4595 422.6735 407.2368 313.0938 404.6383 387.7742 315.8049 383.0769
## [25] 320.8571 591.3404 470.8519 377.7551
##
## $TrueDamageChampions_rojo
##  [1] 313.9677 316.6129 305.6765 303.2895 362.3548 307.6452 367.8235 331.3448
##  [9] 436.2105 365.3673 295.3871 311.7419 297.9111 382.4464 446.6286 330.2727
## [17] 324.7872 313.2500 461.1316 330.7576 309.6562 392.1515
##
## $DamageObjectives_azul
```

```
## numeric(0)
##
## $DamageObjectives_rojo
## numeric(0)
##
## $diferencia_exp
## [1] -5623  5204 -5219  5942 -5909
##
## $diferencia_nivel
##  [1] -5  5 -5 -5 -6  5  6 -5  5 -5  5  5
##
## $cura_azul
##  [1] 2430.444 1813.171 2032.743 2424.704 2060.176 1851.667 1989.378 2075.033
##  [9] 1959.486 1738.070 2323.356 1750.830
##
## $cura_rojo
## [1] 2254.750 1955.581 1824.769 1956.552 1820.889 2053.532 1859.091
##
## $cc_azul
##  [1] 5.444444 5.230769 6.687500 5.205882 6.250000 4.967742 8.694444 4.939394
##  [9] 5.192308 7.800000 7.642857 4.906250 6.515152 5.484848
##
## $cc_rojo
## [1] 5.935484 8.272727 5.550000 5.323529 5.085714 5.636364 7.400000
##
## $ventaja_jung_15
## [1] -25.0  34.4 -25.5  24.4 -28.4  30.7 -29.4
##
## $ventaja_jung
## [1]  44.2  35.6 -42.0 -34.6
```

Se observa que hay una gran cantidad de variables que tienen valores extremos, eso sí, vemos que todos los outliers tienen más o menos sentido en las distribuciones de los datos. Es decir, que aunque en el boxplot se detecta un outlier, este es simplemente un candidato a valor extremo, pero al compararlo con los datos en los boxplots, vemos que realmente este valor es un valor real válido, y que aparece como un outlier por simple distribución de los datos. Es normal, que haya alguna partida que un jugador sobresalga y se haga una cantidad de subditos por minuto alta pero POSIBLE, o que un equipo juegue una composición de curar y por tanto su valor de cura sea muy alto pero POSIBLE, o que se detecten 6 inhibidores como un valor extremo porque normalmente un equipo puede ganar uno o dos pero los inhibidores pueden reaparecer a los 5 minutos con lo que el enemigo tienen que volver a destruirlo.

Vemos como se le puede identificar una lógica a la posible aparación de los outliers en los datos para casi todas las variables, una importante de mencionar es el valor de Infinito para las variables de KDA que se produce cuando un jugador no ha muerto en la partida pero al menos ha conseguido una asistencia o asesinato. Para estas variables podríamos dejar el valor Inf puesto que no está mal, es cierto, pero se puede optar por simplemente volver a los datos iniciales (lol_base) y recalcular el KDA teniendo en cuenta que si las muertes de un jugador son 0, se sustituye el valor por 1. De tal manera que si un jugador asesinó 5 veces, asistió en 2 asesinatos y no murió su KDA será 7 en vez de Infinito. Así podemos distinguir cuando un jugador domina mucho la partida matando mucho y no muere, que podría tener un KDA de 25, y un jugador que no domina pero tampoco muere y podría tener un KDA de 3. Sin hacer esto ambos tendrían Infinito.

```
lol2<-lol_base
```

```
lista<-c('deaths_top_azul','deaths_jng_azul','deaths_mid_azul','deaths_adc_azul','deaths_sup_azul','dea
lol2[lista]<-as.data.frame(sapply(lol2[lista], function(x) ifelse(x==0,1,x)))

lol2$kda_top_azul <- (lol2$kills_top_azul+lol2$assists_top_azul)/lol2$deaths_top_azul
lol2$kda_jng_azul <- (lol2$kills_jng_azul+lol2$assists_jng_azul)/lol2$deaths_jng_azul
lol2$kda_mid_azul <- (lol2$kills_mid_azul+lol2$assists_mid_azul)/lol2$deaths_mid_azul
lol2$kda_adc_azul <- (lol2$kills_adc_azul+lol2$assists_adc_azul)/lol2$deaths_adc_azul
lol2$kda_sup_azul <- (lol2$kills_sup_azul+lol2$assists_sup_azul)/lol2$deaths_sup_azul

lol2$kda_top_rojo <- (lol2$kills_top_rojo+lol2$assists_top_rojo)/lol2$deaths_top_rojo
lol2$kda_jng_rojo <- (lol2$kills_jng_rojo+lol2$assists_jng_rojo)/lol2$deaths_jng_rojo
lol2$kda_mid_rojo <- (lol2$kills_mid_rojo+lol2$assists_mid_rojo)/lol2$deaths_mid_rojo
lol2$kda_adc_rojo <- (lol2$kills_adc_rojo+lol2$assists_adc_rojo)/lol2$deaths_adc_rojo
lol2$kda_sup_rojo <- (lol2$kills_sup_rojo+lol2$assists_sup_rojo)/lol2$deaths_sup_rojo

# tengo los kda bien calculados en lol2
lista_kda<-c('kda_top_azul','kda_jng_azul','kda_mid_azul','kda_adc_azul','kda_sup_azul','kda_top_rojo',

lol2<-lol2 %>% select(all_of(lista_kda))
lista_kda_2 <- paste("2",lista_kda, sep='_')
colnames(lol2) <- lista_kda_2

lol[lista_kda_2]<- lol2


lol$kda_top_azul <- ifelse(lol$kda_top_azul == Inf, lol$`2_kda_top_azul`, lol$kda_top_azul)
lol$kda_jng_azul <- ifelse(lol$kda_jng_azul == Inf, lol$`2_kda_jng_azul`, lol$kda_jng_azul)
lol$kda_mid_azul <- ifelse(lol$kda_mid_azul == Inf, lol$`2_kda_mid_azul`, lol$kda_mid_azul)
lol$kda_adc_azul <- ifelse(lol$kda_adc_azul == Inf, lol$`2_kda_adc_azul`, lol$kda_adc_azul)
lol$kda_sup_azul <- ifelse(lol$kda_sup_azul == Inf, lol$`2_kda_sup_azul`, lol$kda_sup_azul)

lol$kda_top_rojo <- ifelse(lol$kda_top_rojo == Inf, lol$`2_kda_top_rojo`, lol$kda_top_rojo)
lol$kda_jng_rojo <- ifelse(lol$kda_jng_rojo == Inf, lol$`2_kda_jng_rojo`, lol$kda_jng_rojo)
lol$kda_mid_rojo <- ifelse(lol$kda_mid_rojo == Inf, lol$`2_kda_mid_rojo`, lol$kda_mid_rojo)
lol$kda_adc_rojo <- ifelse(lol$kda_adc_rojo == Inf, lol$`2_kda_adc_rojo`, lol$kda_adc_rojo)
lol$kda_sup_rojo <- ifelse(lol$kda_sup_rojo == Inf, lol$`2_kda_sup_rojo`, lol$kda_sup_rojo)

lol<-lol %>% select(-all_of(lista_kda_2))

#repito la imputación realizada antes para tener los datos bien
lol_imputed<-kNN(lol, k=3)

# muestro que se han corregido los nulos. (como ya se producia antes)
sapply(lol_imputed[vars_with_nulls], function(x) sum(is.na(x)))
```

```
##         heraldos_azul         heraldos_rojo           inhibs_azul
##                     0                     0                     0
##           inhibs_rojo           First_Blood        oro_al_15_azul
##                     0                     0                     0
##        oro_al_15_rojo        diff_oro_al_15      ScoreVision_azul
##                     0                     0                     0
##      ScoreVision_rojo DamageObjectives_azul DamageObjectives_rojo
##                     0                     0                     0
```

```
##        diferencia_exp      diferencia_nivel             cura_azul
##                     0                     0                     0
##             cura_rojo               cc_azul               cc_rojo
##                     0                     0                     0
```

```
# muestro que en las imputadas ya no hay estos problemas con las variables de KDA
sapply(lol_imputed[lista_kda], function(x) boxplot.stats(x)$out)
```

```
## $kda_top_azul
##  [1] 15 19 15 14 14 14 16 14 15 17 18 14 17 15 14 14 14 14 14 14 21
##
## $kda_jng_azul
##  [1] 17 17 20 21 17 20 21 18 18 18 17 18 18 17 17 17
##
## $kda_mid_azul
##  [1] 19 22 19 18 22 20 20 20 19 18 20 20 21
##
## $kda_adc_azul
## [1] 21 21
##
## $kda_sup_azul
##  [1] 20 20 22 19 19 20 19 20 19 21
##
## $kda_top_rojo
##  [1] 21 14 16 16 14 16 16 14 15 15 16 14 14 15 20 16 16 14 14 15 14 15 21 16 16
## [26] 15 20 15 14 14
##
## $kda_jng_rojo
##  [1] 16 17 17 18 19 17 16 16 16 16 17 25 23 22 19 16 18 18 18 16 16 17 19 17 18
## [26] 19 18 22 16 18
##
## $kda_mid_rojo
## [1] 22 22 22 19 20
##
## $kda_adc_rojo
## [1] 23 22 22
##
## $kda_sup_rojo
##  [1] 17 20 20 20 17 21 18 20 22 18 20 17 18 23 19 17 19 18 19 18 17 18
```

```
lol_imputed$First_Blood <- as.factor(as.character(lol_imputed$First_Blood))
```

4. Análisis de los datos.

Los análisis de datos son una parte imprescindible en este trabajo, debido a que al realizarlos podemos extraer conocimiento de los datos que tenemos en nuestro dataset. El proceso de analizar los datos no se puede dividir en fases concretas, ya que muchas veces el realizar un análisis o otro depende de los análisis ya realizados y el conocimiento extraído. Por tanto, yo voy a seguir los títulos que se siguen en la práctica, pero de manera que por ejemplo, para la selección de datos voy a utilizar una aplicación de prueba estadística. Los análisis de datos a realizar por tanto son:

- Análisis de correlación de las variables con la variable objetivo 'gana'. Este análisis estadístico lo realizo en la fase de selección para selección un dataset reducido que usar más adelante.
- Aplicación de un análisis estadístico descriptivo de las variables.
- Realización de un contraste de probabilidades.
- Realización de un contraste de hipótesis entre dos grupos, que se definirá posteriormente entre que variables en función de las variables seleccionadas y el estudio de la normalidad y la homocedasticidad.
- Realización de un contraste de hipótesis entre más de dos grupos.
- Desarrollo de un modelo de regresión logística para predecir el ganador de una partida.

4.1 Selección de los grupos de datos que se quieren analizar/comparar.

Para realizar la selección de los grupos de datos que se quieren analizar, podemos hacerlo de diferentes maneras, en definitiva lo importante es seleccionar el conjunto de datos que sea útil para realizar los análisis. En este caso, todavía tenemos muchos datos por cada instancia, así que lo interesante es reducir el conjunto de datos por una reducción de dimensionalidad.

Por una parte, esto se puede hacer mediante algoritmos como PCA, que aplica una reducción de las dimensiones seleccionando m nuevas variables no correlacionadas entre ellas a partir de las variables del dataset. En nuestro caso, para aprovechar la realización de un análisis estadístico, vamos a detectar las variables más correlacionadas con la variable objetivo 'gana'. Seleccionaremos aquellas variables más correlacionadas. De manera que en esta selección hacemos una reducción de la dimensionalidad por la correlación.

Para tener todas las variables numéricas y poder hacer la correlación de Pearson, hemos de hacer un One-HotEncoding de las variables categóricas. Primero muestro cuales son binarias y cuales son nominales

```
lol_imputed<-lol_imputed[,1:ncol(lol)]

vars_2_niveles<-sapply(lol_imputed,function(x) length(levels(x))==2)
vars_2_niveles<-names(vars_2_niveles[which(vars_2_niveles==TRUE)])
print("Las variables binarias son: " )
```

```
## [1] "Las variables binarias son: "
```

```
print(vars_2_niveles)
```

```
## [1] "primera_sangre" "primera_torre"  "gana"
```

```
vars_mas_niveles<-sapply(lol_imputed,function(x) length(levels(x))>2)
vars_mas_niveles<-names(vars_mas_niveles[which(vars_mas_niveles==TRUE)])
print("Las variables nominales son: " )
```

```
## [1] "Las variables nominales son: "
```

```
print(vars_mas_niveles)
```

```
## [1] "First_Blood"     "MasValioso_azul" "MasValioso_rojo"
```

Por tanto, las variables binarias podemos aplicarles un OneHotEncoding sin ningún problema, nos quedaremos solo con una de las categorías sin problema. Por tanto, nos quedaremos con la primera_sangre_azul, la primera_torre_azul y nuestra variable objetivo será gana_azul. De manera que en todas estas variables un 1, será el equipo azul y un 0 el equipo rojo.

Para las variables nominales hay varias posibilidades, podemos mantenerlas como nominales y calcular su coeficiente de correlación respecto a gana_azul mediante un análisis ANOVA, obteniendo el coeficiente de correlación como la raíz cuadrada del eta cuadrado. Otra posibilidad es simplemente aplicar OneHotEncoding a las variables, de manera que cada categoría de la variable será una variable nueva y estudiaremos su correlación con la variable gana.

```
lol_imputed_onehot <- as.data.frame(one_hot(as.data.table(lol_imputed)))
lol_imputed_onehot$gana_rojo<-NULL
lol_imputed_onehot$primera_sangre_rojo<-NULL
lol_imputed_onehot$primera_torre_rojo<-NULL

dim(lol_imputed_onehot)
```

```
## [1] 614 103
```

Vemos que por el OneHotEncoding, ahora tenemos 104 variables debido a que hay 3 variables que se han dividido en varias variables. Esto no supone ningún problema porque vamos a aplicar la correlación sobre todas las variables y reducir el dataset a las más correlacionadas.

```
variables_a_correlacion<-c(colnames(lol_imputed_onehot %>% select(-'gana_azul')))

correlaciones<-as.data.frame(sapply(lol_imputed_onehot[variables_a_correlacion], function(x) cor(x,lol_

colnames(correlaciones)<-'correlacion'

correlaciones$correlacion_abs<-abs(correlaciones$correlacion)

correlaciones<-correlaciones[order(-correlaciones$correlacion_abs),]
correlaciones_importantes<-correlaciones[which(correlaciones$correlacion_abs>0.5),]
correlaciones_importantes %>% kable()
```

	correlacion	correlacion_abs
num_torres_rojo	-0.9055308	0.9055308
num_torres_azul	0.8807422	0.8807422
DamageObjectives_rojo	-0.8565516	0.8565516
DamageObjectives_azul	0.8104547	0.8104547
inhibs_rojo	-0.7709685	0.7709685
kda_adc_rojo	-0.7392104	0.7392104
num_asesinatos_rojo	-0.7341746	0.7341746
inhibs_azul	0.7255286	0.7255286
kda_mid_rojo	-0.7138056	0.7138056
kda_adc_azul	0.7028311	0.7028311
num_nashors_rojo	-0.7023939	0.7023939
kda_jng_rojo	-0.6949891	0.6949891
kda_mid_azul	0.6878344	0.6878344
kda_jng_azul	0.6860955	0.6860955
kda_sup_azul	0.6707849	0.6707849
kda_top_rojo	-0.6531709	0.6531709
kda_sup_rojo	-0.6516090	0.6516090
num_asesinatos_azul	0.6494288	0.6494288
num_dragones_rojo	-0.6468314	0.6468314
kda_top_azul	0.6095378	0.6095378
num_dragones_azul	0.6047383	0.6047383
num_nashors_azul	0.5984568	0.5984568
double_k_rojo	-0.5594135	0.5594135
diff_oro_al_15	0.5523255	0.5523255
oro_al_15_rojo	-0.5137206	0.5137206
diferencia_exp	0.5128507	0.5128507

Vemos que al sacar el valor absoluto de las correlaciones, podemos ordenarlas para sacar aquellas variables que más correlacion tienen con la variable objetivo gana. Al seleccionar las variables, definimos el límite de correlación en 0.5 absoluto. Las 26 variables que tienen una correlación mayor de 0.5 absoluto, y por tanto utilizaremos en nuestro estudio analítico son:

- El número de torres para los dos equipos.
- El daño a objetivos para los dos equipos.
- El número de inhibidores para los dos equipos.
- El KDA para cada uno de los jugadores de la partida.
- El número de asesinatos de cada equipo.
- El número de dragones totales para cada equipo.
- El número de nashors para cada equipo.
- El número de dobles asesinatos realizados por el equipo rojo.
- El oro al minuto 15 para el equipo rojo.
- La diferencia de oro al minuto 15.
- La diferencia de experiencia al minuto 15.

Es importante destacar que, que todas las variables seleccionadas son individuales o si tienen una pareja han aparecido ambas en la selección, salvo la variable oro_al_15_rojo, que no tiene su pareja oro_al_15_azul y la variable double_k_rojo que no tiene su pareja double_k_azul. Por tanto las añado manualmente aunque no sobrepasen el límite de selección de correlación.

```
correlaciones[which(rownames(correlaciones)=='oro_al_15_azul'),]
```

```
##                correlacion correlacion_abs
## oro_al_15_azul   0.3831157       0.3831157
```

```
correlaciones[which(rownames(correlaciones)=='double_k_azul'),]
```

```
##                correlacion correlacion_abs
## double_k_azul    0.4867814       0.4867814
```

Vemos que la variable de asesinatos dobles del equipo azul se ha quedado muy cerca de pasar el límite, lo cual es normal si los asesinatos dobles del equipo rojo lo han pasado. Pero se observa que la correlación de oro_al_15_azul es muy baja, de 0.38. De igual manera, las añado puesto que son dos variables y le otorgan una "lógica" al dataset.

Por último, si miramos las correlaciones vemos que las correlaciones de las variables derivadas de first_blood, MasValiosoAzul y MasValiosoRojo son muy bajas y están lejos del límite, por tanto está claro que pertenecer a una categoría de estas variables no es útil para determinar que equipo gano en la partida.

```
require(tibble)
correlaciones<-rownames_to_column(correlaciones, var = "variable")
correlaciones %>% filter(grepl('MasValioso|First_Blood', `variable`) )
```

```
##                variable correlacion correlacion_abs
## 1  First_Blood_sup_rojo -0.11664760      0.11664760
## 2  First_Blood_adc_azul  0.10789341      0.10789341
## 3  First_Blood_adc_rojo -0.10393502      0.10393502
## 4   MasValioso_rojo_adc -0.09205589      0.09205589
## 5  First_Blood_top_azul  0.08466061      0.08466061
## 6   MasValioso_azul_mid  0.08075076      0.08075076
## 7  First_Blood_mid_rojo -0.07573569      0.07573569
## 8   MasValioso_azul_adc -0.07405478      0.07405478
## 9   MasValioso_rojo_top  0.07027817      0.07027817
## 10 First_Blood_jng_azul  0.07004555      0.07004555
## 11 First_Blood_sup_azul  0.06938505      0.06938505
## 12 First_Blood_jng_rojo -0.06931167      0.06931167
## 13 First_Blood_mid_azul  0.06515693      0.06515693
## 14  MasValioso_rojo_mid  0.05043309      0.05043309
## 15 First_Blood_top_rojo -0.04776586      0.04776586
## 16  MasValioso_azul_jng  0.03346854      0.03346854
## 17  MasValioso_rojo_jng -0.01913386      0.01913386
## 18  MasValioso_azul_top -0.01473929      0.01473929
```

Ariiba observamos que la correlación absoluta mayor es de 0.11 y por tanto estas variables aportarían muy poco.

Por si acaso, y para asegurarnos de que aunque cada categoría de las variables no tiene una gran correlación, si usáramos la variable por si misma tampoco tendría correlación mayor de 0.5, vamos a obtener la correlación mediante un análisis ANOVA para estas variables frente a la variable gana_azul.

```
library(heplots) # for eta
model.aov <- aov(lol_imputed_onehot$gana_azul ~ lol_imputed$First_Blood)
summary(model.aov)
```

```
##                          Df Sum Sq Mean Sq F value  Pr(>F)
## lol_imputed$First_Blood   9   9.75  1.0829    4.56 7.8e-06 ***
## Residuals               604 143.43  0.2375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
paste0("El coeficiente de correlación de FirstBlood sería: ",sqrt(etasq(model.aov, partial = FALSE)[1,1
```

```
## [1] "El coeficiente de correlación de FirstBlood sería: 0.252245209111898"
```

```
library(heplots) # for eta
model.aov <- aov(lol_imputed_onehot$gana_azul ~ lol_imputed$MasValioso_rojo)
summary(model.aov)
```

```
##                              Df Sum Sq Mean Sq F value Pr(>F)
## lol_imputed$MasValioso_rojo   3    1.6  0.5340   2.149 0.0929 .
## Residuals                   610  151.6  0.2485
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
paste0("El coeficiente de correlación de MasValioso_rojo sería: ",sqrt(etasq(model.aov, partial = FALSE)
```

```
## [1] "El coeficiente de correlación de MasValioso_rojo sería: 0.102265373545839"
```

```
library(heplots) # for eta
model.aov <- aov(lol_imputed_onehot$gana_azul ~ lol_imputed$MasValioso_azul)
summary(model.aov)
```

```
##                              Df Sum Sq Mean Sq F value Pr(>F)
## lol_imputed$MasValioso_azul   3    1.3   0.432   1.735  0.159
## Residuals                   610  151.9   0.249
```

```
paste0("El coeficiente de correlación de MasValioso_azul sería: ",sqrt(etasq(model.aov, partial = FALSE)
```

```
## [1] "El coeficiente de correlación de MasValioso_azul sería: 0.091981504975099"
```

Vemos que sin duda, la correlación de First_Blood por sí sola es superior a la de sus categorías por separado. De igual manera, su correlación es la mitad del límite, por tanto podemos olvidarnos de esta variable y mantener las 26 variables seleccionadas por el límite más las dos que vamos a añadir por ser pareja de alguna variable seleccionada. Las otras variables siguen teniendo correlaciones muy bajas.

```
variables_importantes<-rownames(correlaciones_importantes)
variables_importantes<-c(variables_importantes, 'oro_al_15_azul', 'double_k_azul', 'gana_azul')

lol_imputed_onehot<-lol_imputed_onehot[variables_importantes]

dim(lol_imputed_onehot)
```

```
## [1] 614  29
```

Vemos que el dataset se ha reducido a 29 variables, que son las 28 variables seleccionadas por correlación más la variable objetivo.

4.2 Comprobación de la normalidad y homogeneidad de la varianza.

Para poder realizar un contraste de hipótesis, ya sea una comparación entre dos grupos o entre más grupos, es necesario comparar la normalidad y homocedasticidad de las variables. En caso de que las variables sean normales y tengan homogeneidad de la varianza, se pueden aplicar test paramétricos como la t de Student o ANOVA. Sino, se tendrán que aplicar test no paramétricos como Wilcoxon o Mann-Whitney o Kruskal-Wallis.

Respecto a la comprobación de normalidad y homocedasticidad, yo voy a realizarlo todo en el conjunto de datos que tenemos actualmente. Pero si queremos realizar un nuevo análisis de una variable aquí no presente, tendremos que comprobar su normalidad, y en caso de que queramos realizar un contraste de hipótesis de una variable presente pero utilizando una variable categórica eliminada para separar en grupos, como por ejemplo el equipo o la liga, tendremos que comprobar la homocedasticidad en función de esta variable categórica.

4.2.1 Comprobación de la normalidad

Primero compruebo la normalidad de las variables numéricas. Esta claro que la variable objetivo gana_azul no se tiene que estudiar su normalidad puesto que es una variable binaria. La normalidad de las variables se comprueba mediante el test de Saphiro-Wilk, su hipótesis nula es que la variable se encuentra distribuída normalmente, por tanto, si su p-valor es menor a 0.05 es que la variable no se encuentra distribuída normalmente.

De igual manera, por el teorema del límite central a partir de una muestra de más de 30, se puede considerar que la distribución es normal aunque los test como Saphiro-Wilk digan que no.

```
lol_imputed_onehot$gana_azul<-as.factor(ifelse(lol_imputed_onehot$gana_azul == 1, 'si','no'))
lol_imputed_onehot$gana_azul <- factor(lol_imputed_onehot$gana_azul, levels=c("si", "no") )

lol_numericas <- lol_imputed_onehot %>% select(where(is.numeric) | where(is.integer))

pvalores<-sapply(lol_numericas, function(x) shapiro.test(x)$p.value)

print("Las variables con distribución normal y sus p-valores son:")
```

```
## [1] "Las variables con distribución normal y sus p-valores son:"
```

```
pvalores[which(pvalores>=0.05)]
```

```
## diff_oro_al_15 diferencia_exp
##      0.6060099      0.6419978
```

```
normales<-names(pvalores[which(pvalores>=0.05)])

print("Las variables sin distribución normal y sus p-valores son:")
```

```
## [1] "Las variables sin distribución normal y sus p-valores son:"
```

```
pvalores[which(pvalores<0.05)]
```

```
##       num_torres_rojo       num_torres_azul DamageObjectives_rojo
##          4.270106e-21          9.371585e-21          3.004768e-13
```

```
## DamageObjectives_azul           inhibs_rojo          kda_adc_rojo
##          7.539488e-10          3.430318e-28          9.513651e-20
##   num_asesinatos_rojo           inhibs_azul          kda_mid_rojo
##          3.715099e-09          1.169276e-25          1.273242e-22
##          kda_adc_azul      num_nashors_rojo          kda_jng_rojo
##          1.682133e-18          1.760100e-28          1.135086e-24
##          kda_mid_azul          kda_jng_azul          kda_sup_azul
##          1.650853e-21          5.189400e-22          2.054771e-22
##          kda_top_rojo          kda_sup_rojo   num_asesinatos_azul
##          8.510736e-25          1.434896e-25          2.310811e-07
##     num_dragones_rojo          kda_top_azul     num_dragones_azul
##          2.011767e-16          3.130283e-23          4.001844e-16
##      num_nashors_azul        double_k_rojo        oro_al_15_rojo
##          2.209091e-28          3.382190e-27          8.614231e-07
##        oro_al_15_azul        double_k_azul
##          3.250350e-07          1.495339e-25
```

```
nonormales<-names(pvalores[which(pvalores<0.05)])
```

Vemos que las unicas variables con distribución normal son la diferencia de oro en el minuto 15 y la diferencia de experiencia en el minuto 15. Las otras variables están muy lejos de tener una distribución normal al ver sus p-valores. Esto lo tendremos en cuenta al hacer contraste de hipótesis.

4.2.2 Comprobación de la homocedasticidad.

Partiendo de la normalidad, hay dos maneras de comprobar la homocedasticidad de las variables. Si la variable sigue una distribución normal, la varianza se ha comprobar por el Levene Test, mientras que si la variable no sigue no distribución normal se ha de comprobar la homocedasticidad por el Fligner Test. En ambos test, la hipótesis nula asume homocedasticidad de las varianzas, por lo que si el p-valor es de menos de 0.05 se rechaza la hipótesis nula y hay heterocedasticidad de las varianzas.

Es importante mencionar, que yo voy a obtener la homocedasticidad para las variables numéricas frente a la variable objetivo gana_azul.

```
pvalores<-sapply(lol_numericas[normales], function(x) leveneTest(x ~ lol_imputed_onehot$gana_azul)$`Pr(

print("Las variables con distribución normal, tienen todas homocedasticidad porque sus p-valores son:")
```

```
## [1] "Las variables con distribución normal, tienen todas homocedasticidad porque sus p-valores son:"
```

```
pvalores[which(pvalores>=0.05)]
```

```
## diff_oro_al_15 diferencia_exp
##      0.5424268      0.1286190
```

```
normales_homocedasticidad<-names(pvalores[which(pvalores>=0.05)])

pvalores<-sapply(lol_numericas[nonormales], function(x) fligner.test(x ~ lol_imputed_onehot$gana_azul)$p

print("Las variables sin distribución normal, que no tienen homocedasticidad y sus p-valores son:")
```

```
## [1] "Las variables sin distribución normal, que no tienen homocedasticidad y sus p-valores son:"
```

```
pvalores[which(pvalores<0.05)]
```

```
##     num_torres_azul         inhibs_rojo        kda_adc_rojo num_asesinatos_rojo
##        1.943917e-06        4.057637e-52        2.327070e-28        1.043875e-02
##         inhibs_azul        kda_mid_rojo        kda_adc_azul    num_nashors_rojo
##        1.075699e-31        3.696383e-47        2.451684e-26        7.219937e-09
##        kda_jng_rojo        kda_mid_azul        kda_jng_azul        kda_sup_azul
##        3.768592e-55        2.471075e-42        3.480358e-35        2.720269e-43
##        kda_top_rojo        kda_sup_rojo num_asesinatos_azul   num_dragones_rojo
##        1.220514e-41        5.278554e-48        1.572984e-02        5.314515e-03
##        kda_top_azul   num_dragones_azul    num_nashors_azul       double_k_rojo
##        2.247062e-36        1.205833e-02        2.792483e-07        1.319538e-21
##      oro_al_15_rojo       double_k_azul
##        4.157288e-04        9.542415e-13
```

```
nonormales_heterocedasticidad<-names(pvalores[which(pvalores<0.05)])

print("Las variables sin distribución normal, que tienen homocedasticidad y sus p-valores son:")
```

```
## [1] "Las variables sin distribución normal, que tienen homocedasticidad y sus p-valores son:"
```

```
pvalores[which(pvalores>=0.05)]
```

```
##       num_torres_rojo DamageObjectives_rojo DamageObjectives_azul
##             0.3792955             0.5732751             0.2556820
##        oro_al_15_azul
##             0.1869097
```

```
nonormales_heterocedasticidad<-names(pvalores[which(pvalores>=0.05)])
```

Por tanto, podemos concluir que las 2 únicas variables con distribución normal, además tienen homocedasticidad debido a que sus p-valores son superiores 0.05, son la diferencia de oro al 15 y la diferencia de experiencia al 15.

Por otra parte, el resto de variables no tienen una distribución normal, pero hay un pequeño grupo de variables que si que tiene varianzas iguales para los diferentes grupos en función de la variable objetivo, estas son el daño a objetivos por parte del equipo rojo y azul, el número de torres del equipo rojo y el oro al 15 del equipo azul. El resto de variables presentan heterocedasticidad.

4.3 Aplicación de pruebas estadísticas para comparar los grupos de datos.

Hay que recordar que la primera prueba estadística ya la hemos realizado en la selección de atributos con el objetivo de reducir la dimensionalidad mediante la identificación de la correlación de las diferentes variables con la variable objetivo.

Las pruebas a realizar son:

- Análisis estadístico descriptivo.
- Contraste de proporción de victoria del azul es superior al rojo.
- Contraste de que el daño a objetivos por parte del equipo azul es superior cuando gana a cuando pierde.
- Contraste de si la diferencia de experiencia en el minuto 15 es igual independientemente de si gana o pierde el equipo azul.
- Contraste de diferencia de KDA entre posiciones para el equipo ganador.

4.3.1 Análisis estadístico descriptivo.

Antes de realizar análisis estadísticos más profundos para determinar características del dataset, es importante realizar un análisis descriptivo que nos permita identificar aspectos importantes del conjunto de datos. Dentro de este análisis descriptivo se podría incluir el estudio de normalidad y varianza ya realizado hasta ahora. Por tanto, me voy a rescindir a estudiar el sumario del conjunto de datos y representaré las variables en gráficas. **A pesar de que las gráficas deberían de pertenecer al ejercicio 5, es lógico que para estudiar el análisis descriptivo del conjunto de datos representemos un histograma de las variables, por tanto desarrollo los histogramas de las variables en este apartado.**

Para realizar este análisis y no abrumarnos, vamos a ir comparando las variables en función de su ámbito.

4.3.1.1 Descripción de las objetivos, daño a objetivos, y ganar

Debido a que la variable gana_azul es la variable objetivo la introduzco en el análisis de los objetivos. Además así la podemos tener presente en todos los análisis para describir comportamientos.

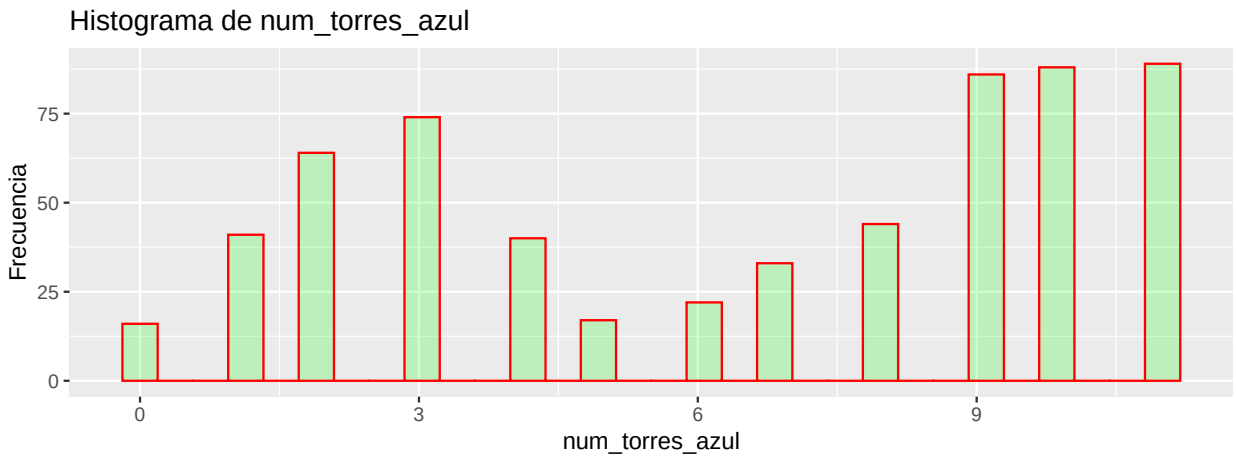
```
# Plot del histograma
plot_hist<-function(x)
{
  print(ggplot(data=lol_imputed_onehot, aes(x=unlist(lol_imputed_onehot[x]))) +
  geom_histogram(
                col="red",
                fill="green",
                alpha = .2) +
  labs(title=paste0("Histograma de ",x)) +
  labs(x=x, y="Frecuencia"))
}

vars <- c('num_torres_azul', 'num_torres_rojo', 'DamageObjectives_azul','DamageObjectives_rojo','inhibs

summary(lol_imputed_onehot[vars])
```

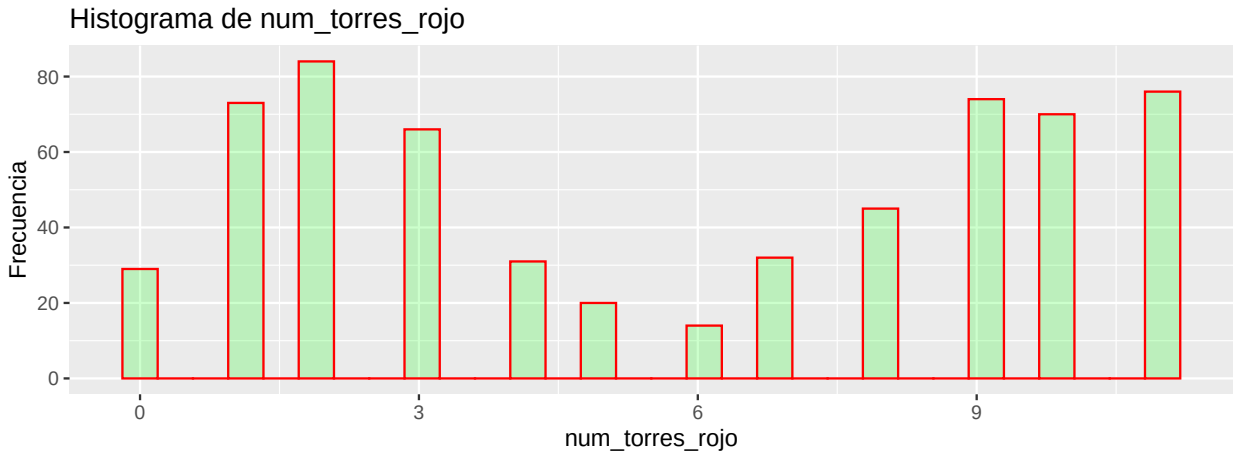
```
##  num_torres_azul  num_torres_rojo  DamageObjectives_azul DamageObjectives_rojo
##  Min.   : 0.000   Min.   : 0.000   Min.   : 235.7        Min.   : 225.0
##  1st Qu.: 3.000   1st Qu.: 2.000   1st Qu.:1221.6        1st Qu.: 975.5
##  Median : 7.500   Median : 6.000   Median :2018.8        Median :1758.0
##  Mean   : 6.489   Mean   : 5.754   Mean   :1957.4        Mean   :1791.0
##  3rd Qu.:10.000   3rd Qu.: 9.000   3rd Qu.:2644.5        3rd Qu.:2571.9
##  Max.   :11.000   Max.   :11.000   Max.   :3905.3        Max.   :3784.8
##   inhibs_azul     inhibs_rojo   gana_azul
##  Min.   :0.000   Min.   :0.00   si:321
##  1st Qu.:0.000   1st Qu.:0.00   no:293
##  Median :1.000   Median :0.00
##  Mean   :1.088   Mean   :0.93
##  3rd Qu.:2.000   3rd Qu.:2.00
##  Max.   :5.000   Max.   :6.00
```

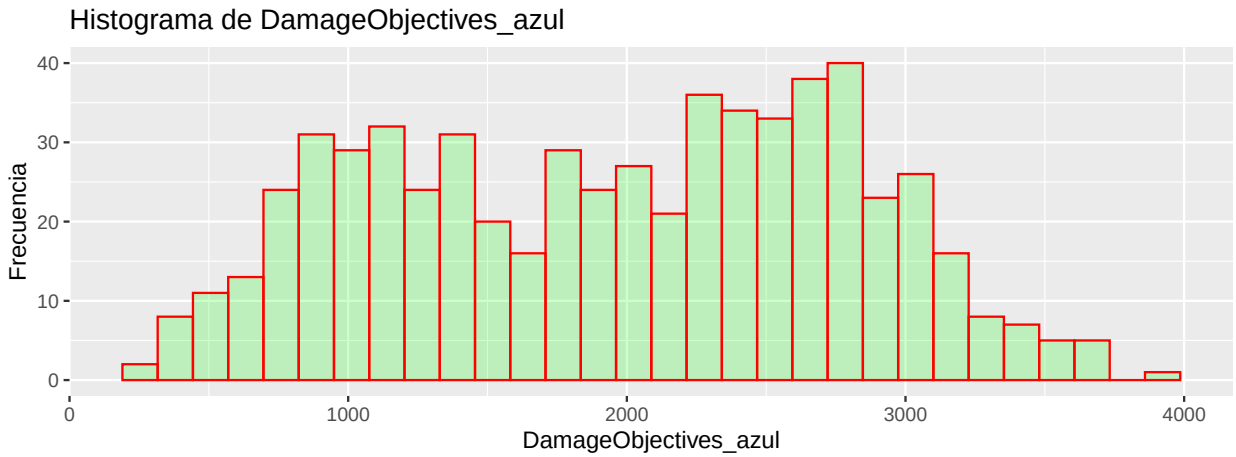
```
gana_azul_porcentaje<-nrow(lol_imputed_onehot[which(lol_imputed_onehot$gana_azul=='si'), ])/nrow(lol_imp
paste0("El equipo azul gana un ", round(gana_azul_porcentaje,3), " y el rojo un ", 1-round(gana_azul_po
```

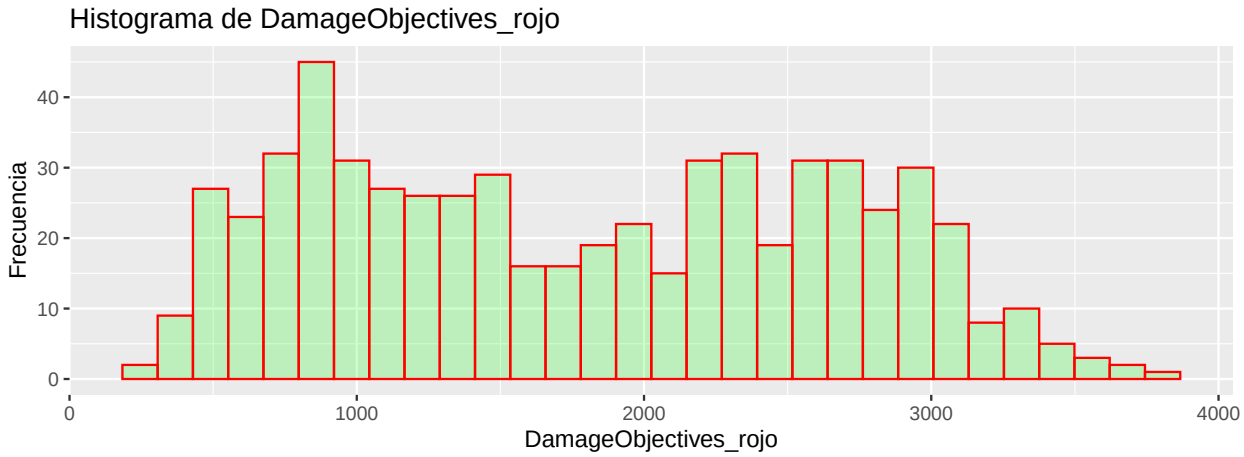
```
## [1] "El equipo azul gana un 0.523 y el rojo un 0.477"
```

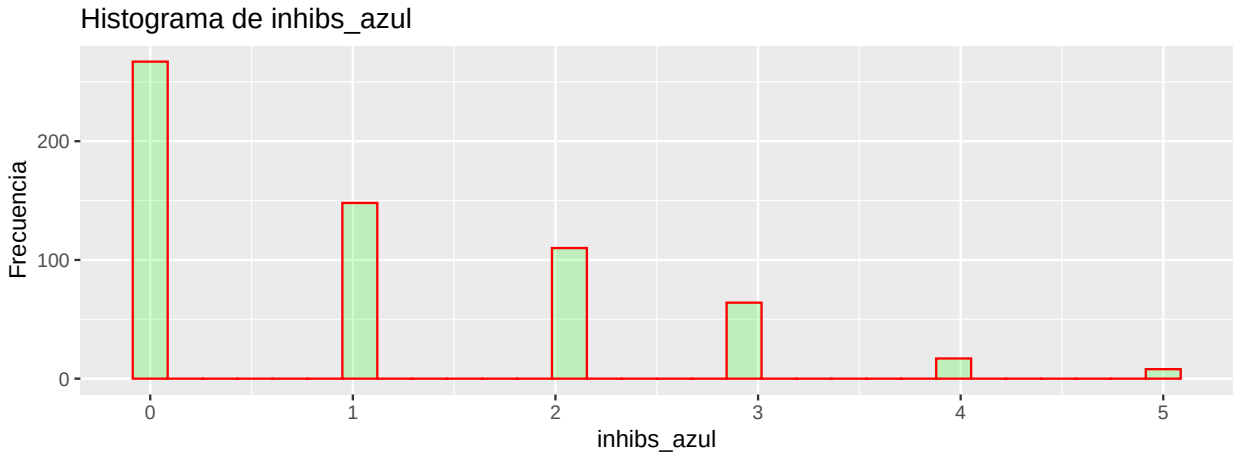
```r
invisible(sapply(vars[1:(length(vars)-1)],plot_hist))
```

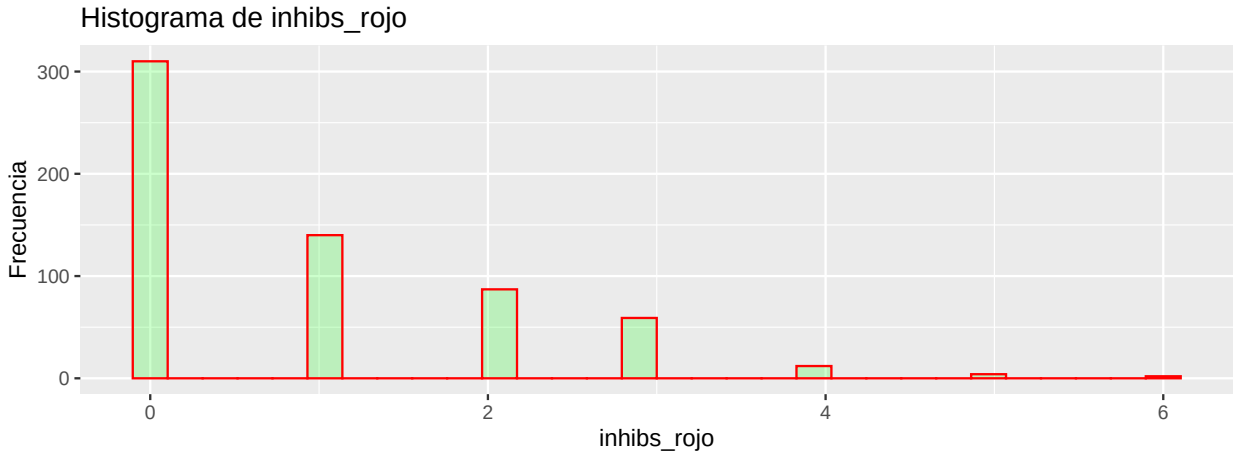












Lo primero que llama la atención es que el equipo azul gana considerablemente más que el equipo rojo. Esto puede ser pura coincidencia o puede ser debido a que el equipo del lado azul tenga una ventaja. Generalmente, en el juego se prefiere jugar como el equipo azul, debido a que selecciona el primer campeón de la partida y que tiene un mejor acceso al lugar donde está el nashor. Sin embargo el rojo tiene el último campeón y mejor acceso al lugar del dragón.

Respecto al resto de variables:

1. Respecto a las variables de torres:

- Coinciden en el máximo y en el mínimo, esto es porque hay partidas donde un equipo no destruye torres y otras partidas donde un equipo destruye todas las torres (11).
- Vemos como tanto en los cuartiles como en la media y mediana se observa que los valores de torres del equipo azul son superiores a los del rojo. Esto puede ser derivado por la ventaja que hemos visto que tiene el equipo azul, ya que sabemos que las variables tienen alta correlación con la variable objetivo, y por tanto, cuanto más torres destruye el equipo azul, más posibilidades de ganar la partida.
- Además, al estudiar los histogramas vemos que la distribución de las torres es una distribución bimodal, donde se ve claramente que un equipo consigue normalmente muchas torres o muy pocas, pero un valor torres intermedio no suele ocurrir. Esto se debe a que un equipo gana y consigue muchas torres o pierde y consigue pocas torres. Es dificil que ambos equipos consigan un mismo numero de torres.

2. Respecto a la variable de daño a objetivos:

- Se observa igual que en torres que los valores de tendencia central son superiores en el equipo azul, lo que posiblemente se debe a que el equipo azul gana más.
- En el estudio de los histogramas de nuevo vemos que ambas variables de daño a objetivos son bimodales. Hay dos "montañas" en la distribución, lo normal es que en la variable de daño a objetivos azul, sea mayor cuando el equipo azul gane y menor cuando el equipo rojo (al contrario en la variable daño a objetivos rojo), lo que hace que se generen estas dos montañas.
- Podemos hacer un contraste de hipótesis donde comprobemos si la media de daño a objetivos azul es la misma cuando gana el equipo azul o el rojo y posiblemente vemos como la media cuando gane cae en la montaña derecha y la media cuando pierde el azul cae en la montaña izquierda.

3. Respecto a las variables de inhibidores destruidos por el equipo:

- La diferencia entre equipo azul y rojo en el sumario se aprecia menos, aun así es interesante comentar como la mediana de inhibidores del rojo es 0, es decir que en más de la mitad de partidas ni consiguen un inhibidor, algo que es indispensable para ganar. Por otra parte la mediana del equipo azul si es 1. Como aspecto interesante, vemos como el máximo de inhibidores del rojo es 6 mientras que el del equipo azul es 5. Esto se puede deber a que el equipo azul gana antes de que sea necesario volver a destruir los 3 inhibidores.
- La variable es numérica, no categórica, aunque vemos que solo pueden ser enteros y los valores en el dataset para las variables son pocos de 0 a 6 (o 5). En el histograma por tanto, podemos definir que la distribución de la variable es exponencial, ya que las primeras categorias tienen mucha mas posibilidad de ocurrir que las últimas. Esto se debe a que el equipo que gana mínimo tiene que destruir un inhibidor para ganar, pero no es necesario más, por lo que incluso cuando se ganan partidas, se puede ganar con un solo inhibidor.

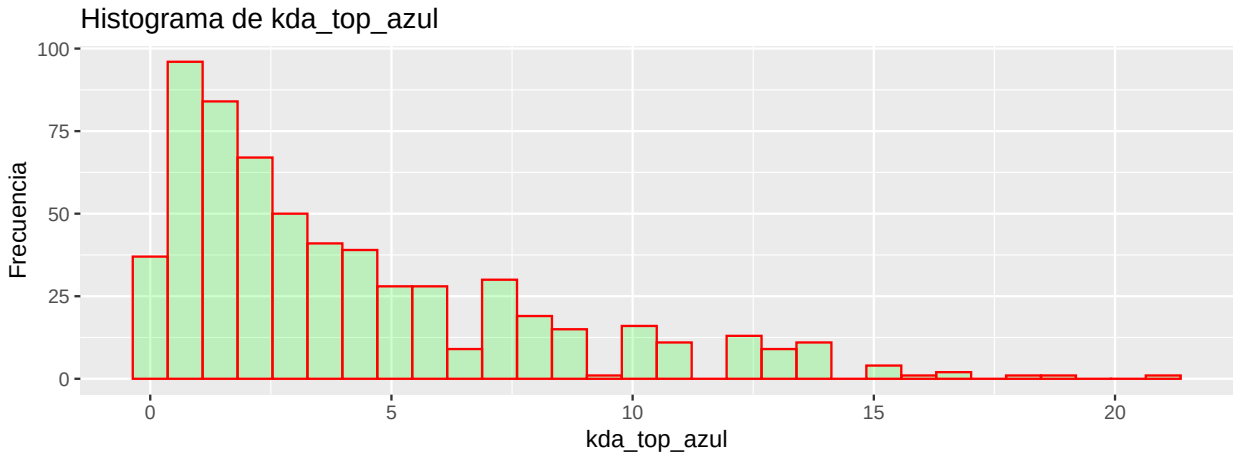
4.3.1.2 Descripción de los KDA

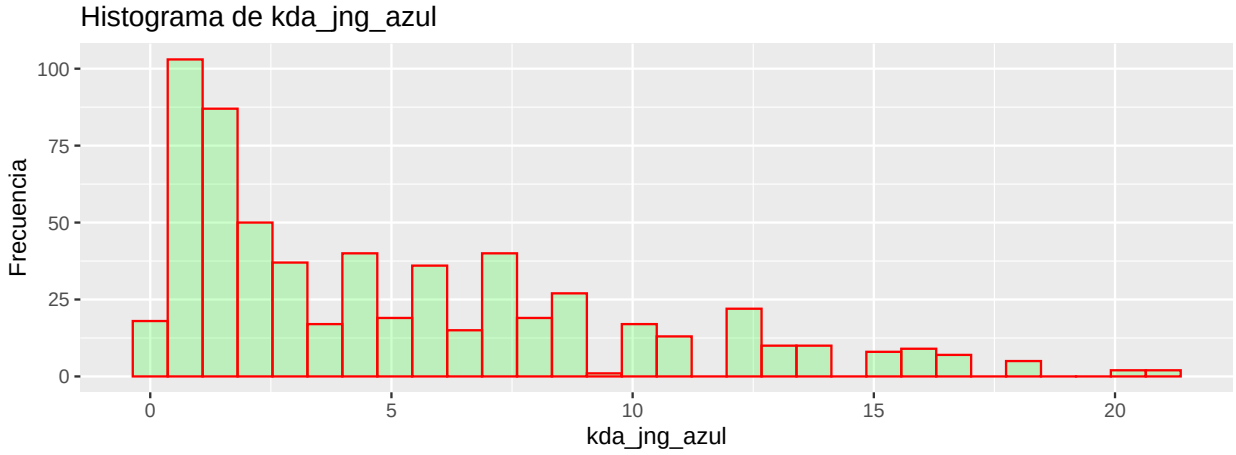
```
vars <- c('kda_top_azul', 'kda_jng_azul', 'kda_mid_azul','kda_adc_azul','kda_sup_azul','kda_top_rojo',

summary(lol_imputed_onehot[vars])
```

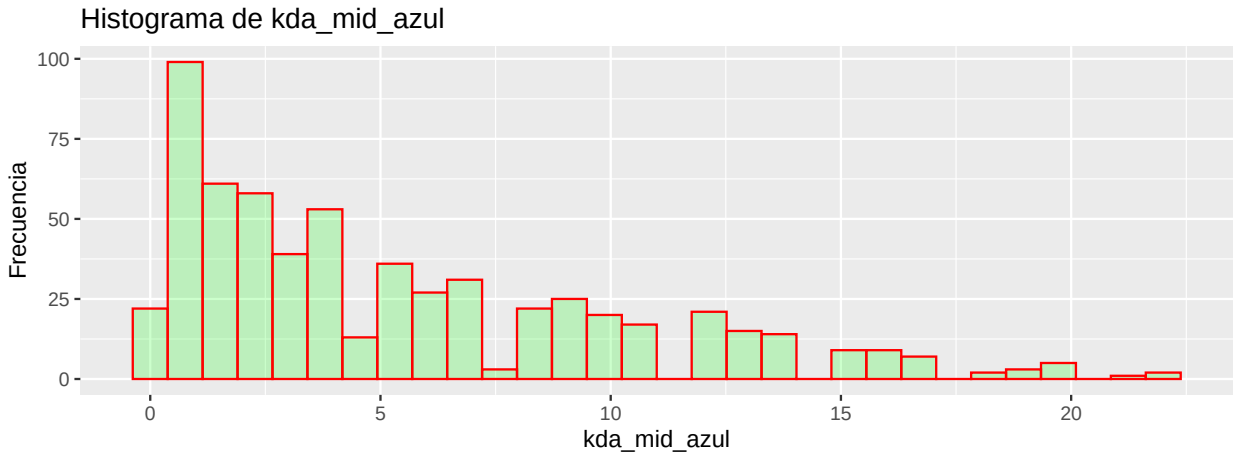
```
##   kda_top_azul     kda_jng_azul     kda_mid_azul     kda_adc_azul
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 1.259   1st Qu.: 1.400   1st Qu.: 1.500   1st Qu.: 1.667
##  Median : 3.000   Median : 3.583   Median : 4.000   Median : 4.500
##  Mean   : 4.137   Mean   : 5.133   Mean   : 5.449   Mean   : 5.780
##  3rd Qu.: 6.000   3rd Qu.: 7.500   3rd Qu.: 8.000   3rd Qu.: 9.000
##  Max.   :21.000   Max.   :21.000   Max.   :22.000   Max.   :21.000
##   kda_sup_azul     kda_top_rojo     kda_jng_rojo     kda_mid_rojo
```

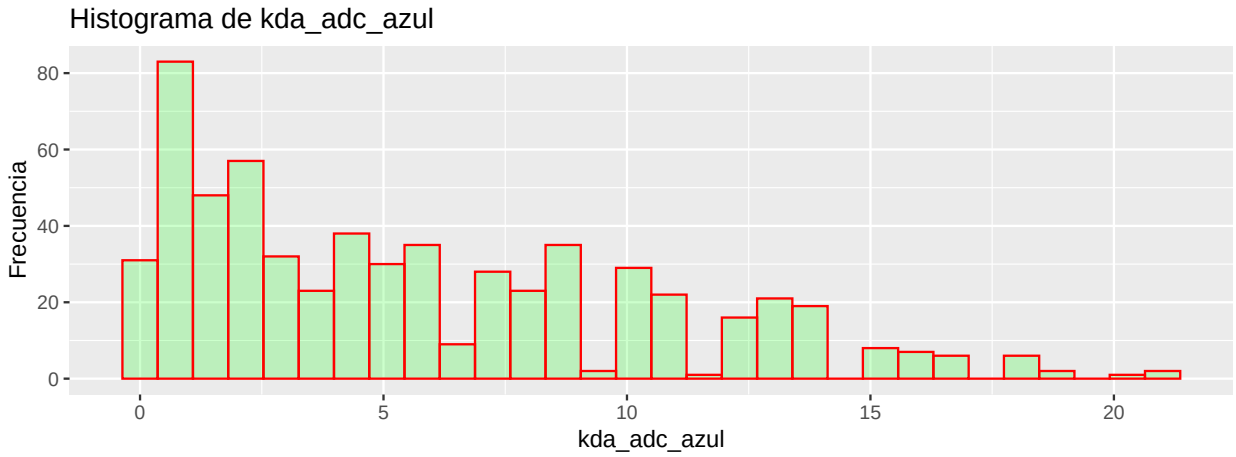
```
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 1.200   1st Qu.: 1.000   1st Qu.: 1.259   1st Qu.: 1.333
##  Median : 3.292   Median : 2.500   Median : 3.000   Median : 3.225
##  Mean   : 5.032   Mean   : 4.155   Mean   : 4.930   Mean   : 5.146
##  3rd Qu.: 8.000   3rd Qu.: 6.000   3rd Qu.: 7.000   3rd Qu.: 8.000
##  Max.   :22.000   Max.   :21.000   Max.   :25.000   Max.   :22.000
##   kda_adc_rojo    kda_sup_rojo
##  Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 1.50   1st Qu.: 1.000
##  Median : 4.00   Median : 2.571
##  Mean   : 5.67   Mean   : 4.627
##  3rd Qu.: 9.00   3rd Qu.: 7.000
##  Max.   :23.00   Max.   :23.000
```

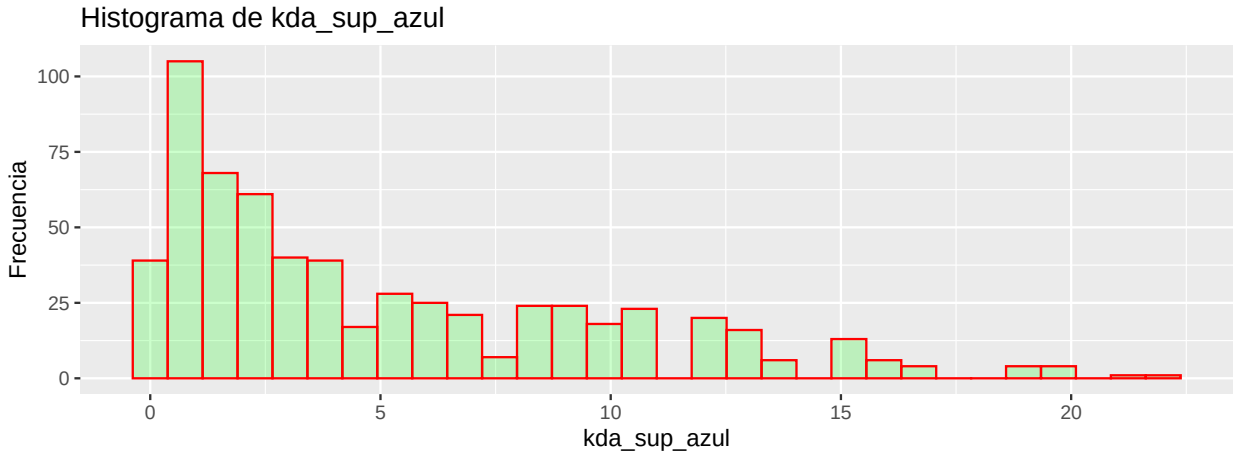
```
invisible(sapply(vars,plot_hist))
```

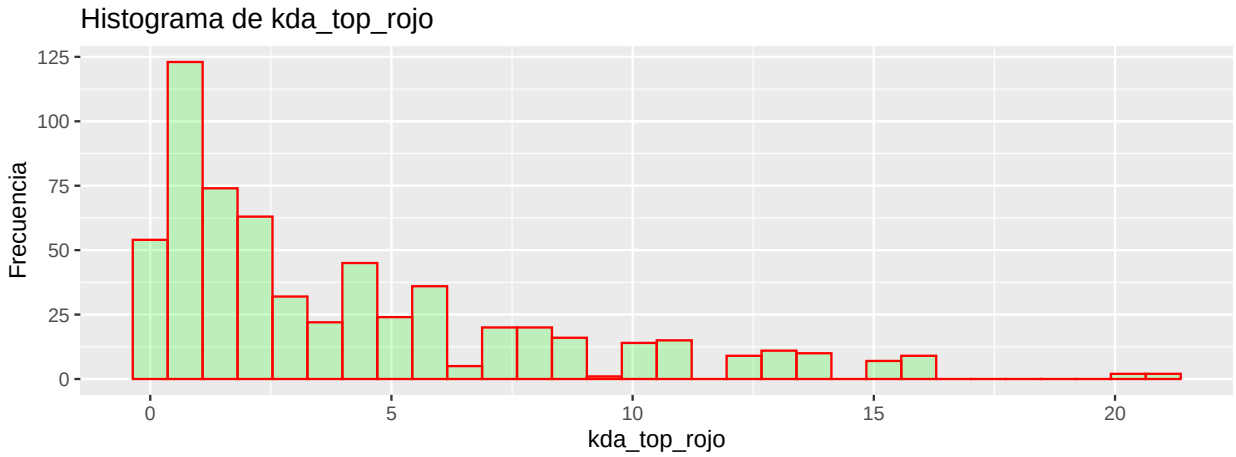


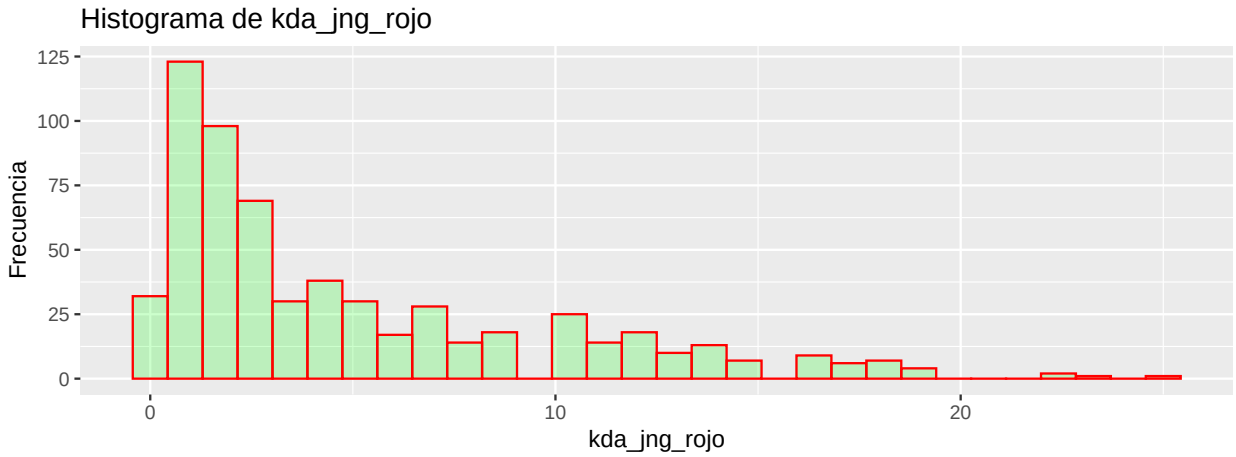


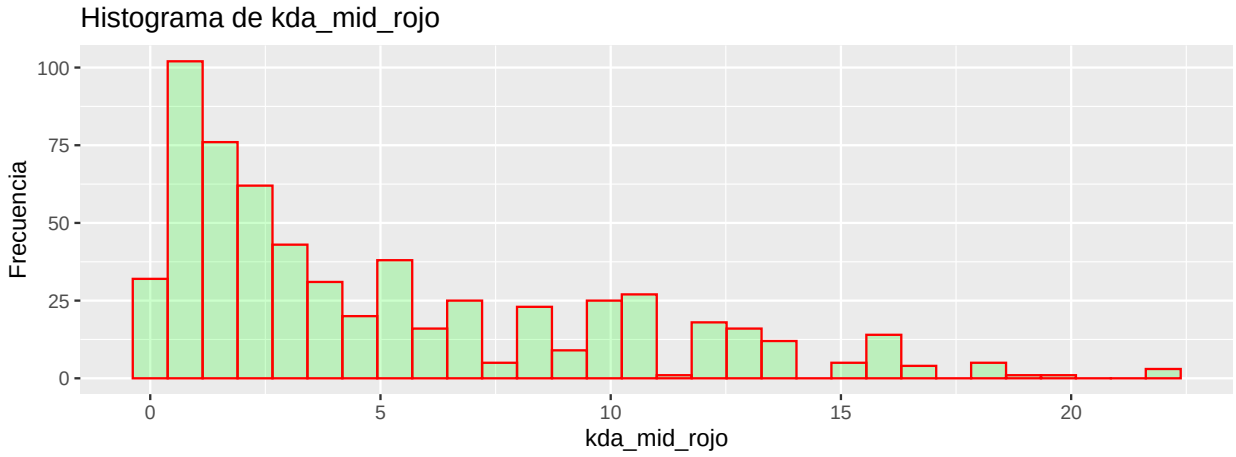

Histograma de kda_mid_azul
Frecuencia
100
75
50
25
0
0
5
10
15
20
kda_mid_azul

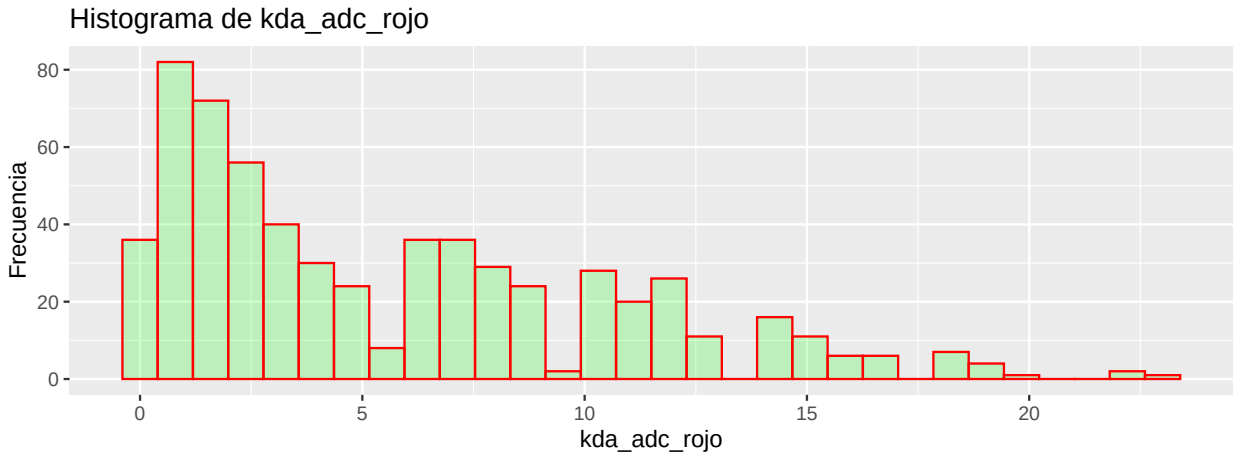

Histograma de kda_adc_azul
Frecuencia
80
60
40
20
0
0
5
10
15
20
kda_adc_azul

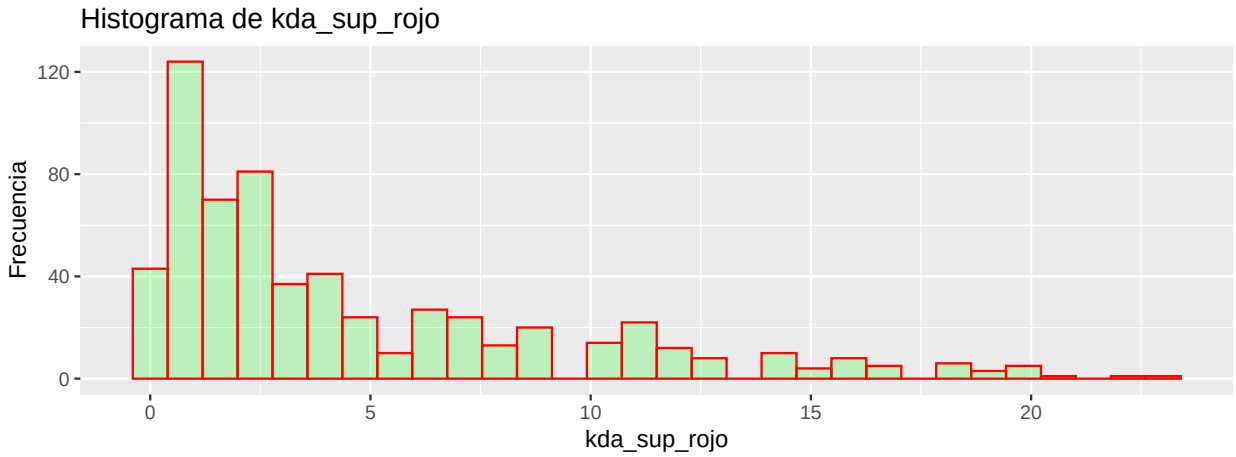

Histograma de kda_sup_azul
Frecuencia
100
75
50
25
0
0
5
10
15
20
kda_sup_azul


Histograma de kda_top_rojo
Frecuencia
125
100
75
50
25
0
0
5
10
15
20
kda_top_rojo


Histograma de kda_jng_rojo
Frecuencia
125
100
75
50
25
0
0
10
20
kda_jng_rojo


Histograma de kda_mid_rojo
Frecuencia
100
75
50
25
0
0
5
10
15
20
kda_mid_rojo





Para los KDA podemos comentar:

- Al ver el sumario, pdoemos ver que según la posición hay un KDA mayor o menor, es decir que el KDA del top azul o rojo son más o menos parecidos, igual que el KDA de los junglas, de los medios, de los adc o de los support. Vemos por tanto, que de media, los jugadores con peor KDA son el top, seguido del support. En medio se encuentra el jungla, y los mejores KDA los tienen los medios, y sobretodo los ADC.
- Por otra parte, al estudiar los histogramas, vemos que todos tienen la misma distribución, aparentemente una distribución lognormal, donde el valor que más aparece está muy cerca al límite inferior (que es 0), pero que la distribución se extiende mucho a valores muy superiores que esta moda (cada vez con menor posibilidad de aparación). Es decir, que lo normal es que el KDA de un jugador sea entre 1 y 2, pero que el valor de este KDA puede superar de 10 o 15 sin problema.

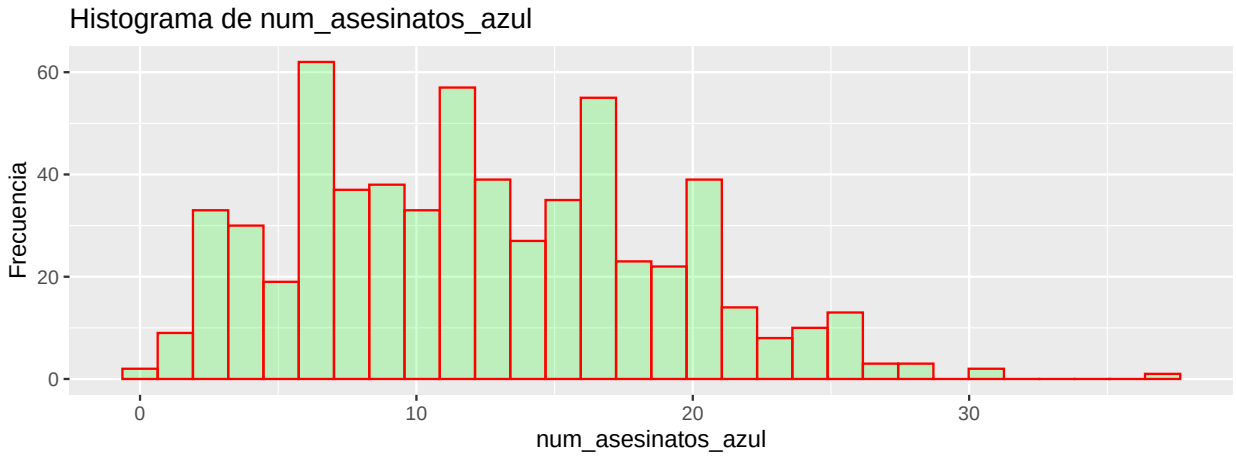
4.3.1.3 Descripción de los asesinatos

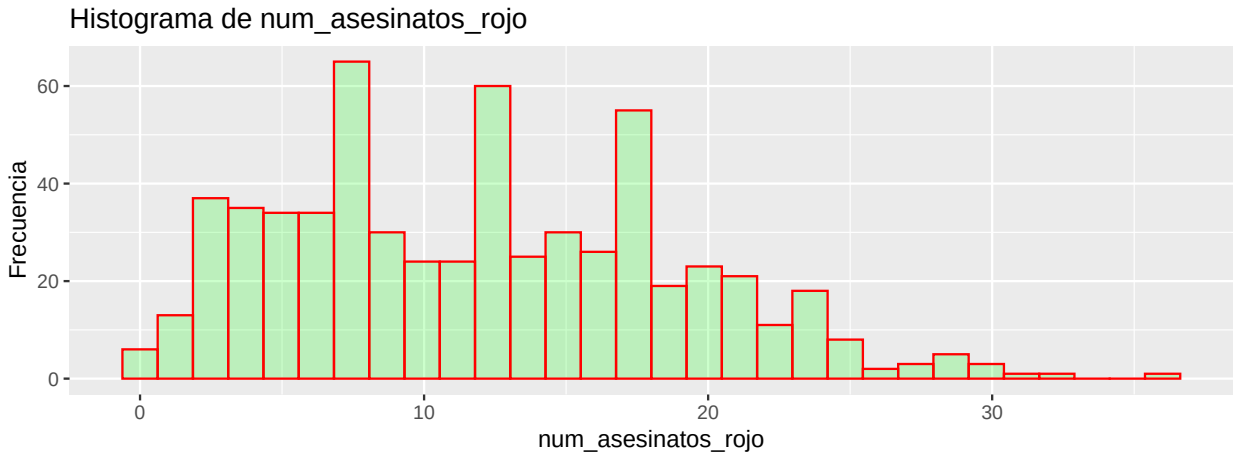
```
vars <- c('num_asesinatos_azul', 'num_asesinatos_rojo', 'double_k_azul','double_k_rojo')

summary(lol_imputed_onehot[vars])
```

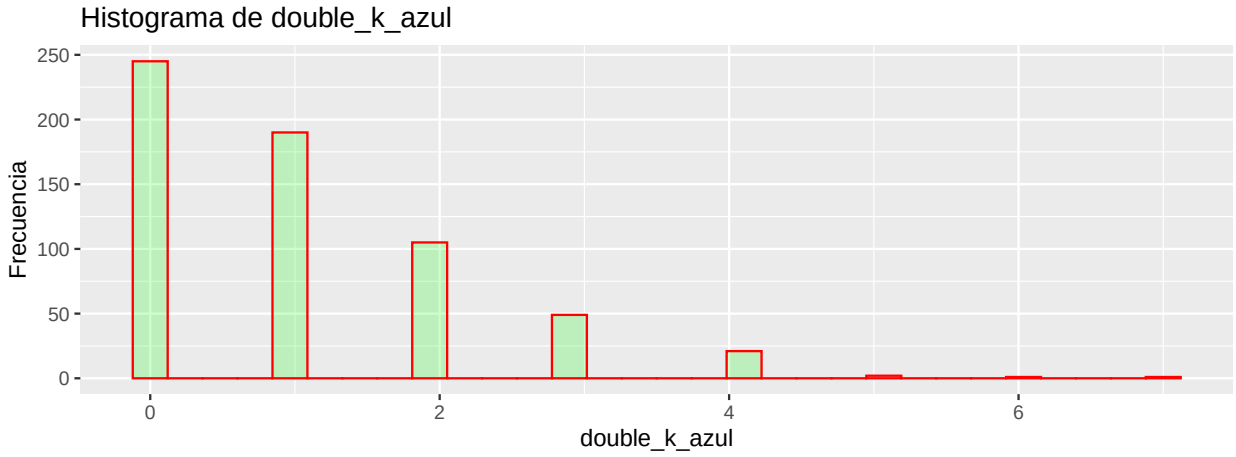
```
##  num_asesinatos_azul num_asesinatos_rojo double_k_azul   double_k_rojo
##  Min.   : 0.00       Min.   : 0.00       Min.   :0.000   Min.   :0.000
##  1st Qu.: 7.00       1st Qu.: 6.00       1st Qu.:0.000   1st Qu.:0.000
##  Median :12.00       Median :12.00       Median :1.000   Median :1.000
```

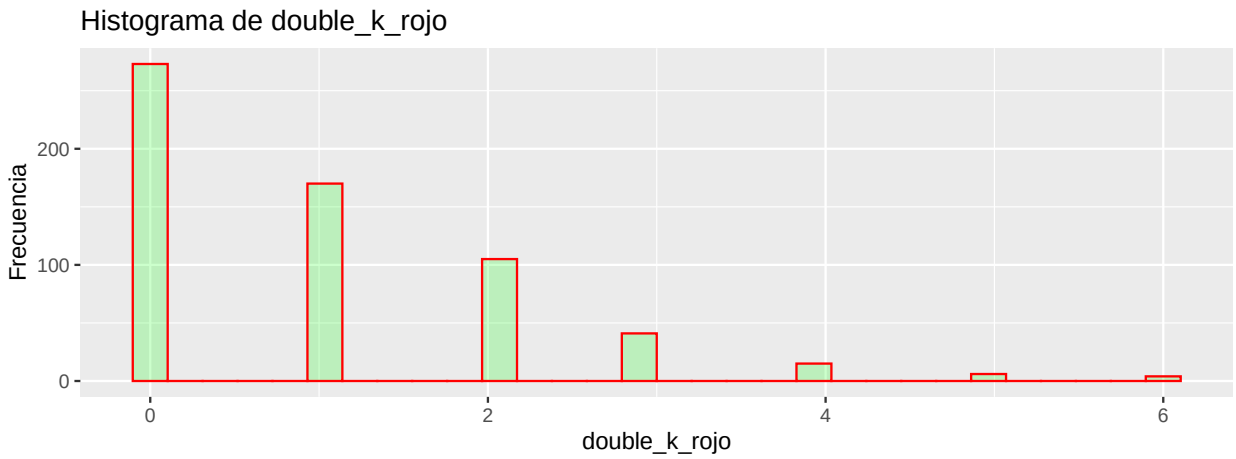
```
##  Mean   :12.38       Mean   :12.06       Mean   :1.065   Mean   :1.005
##  3rd Qu.:17.00       3rd Qu.:17.00       3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :37.00       Max.   :36.00       Max.   :7.000   Max.   :6.000
```

```
invisible(sapply(vars,plot_hist))
```









1. Al estudiar los asesinatos por equipos podemos ver:

- El número de asesinatos por equipo parece ser similar entre ambos equipos. Los valores son similares, vemos que ambos equipos de mediana tienen 12 asesinatos, y aunque el equipo azul tiene una media superior apenas se diferencian, por lo que parece que independientemente de la partida y su ganador, ambos equipos consiguen varios asesinatos. Eso sí, está claro que hay partidas donde algún equipo consigue 0 asesinatos, porque a veces un equipo es muy superior al otro.
- Al ver el histograma, podemos ver como la distribución es positiva asimétrica, donde no se puede asegurar una forma concreta con nombre, Vemos de nuevo que los asesinatos suelen ser un valor entre 5 y 20, para ambos equipos y que pocas veces se obtienen menos o más asesinatos.
- En comparación con las variables de daño a objetivos o torres, al ver esta distribución y no una distribución bimodal, vemos que no es tan claro que conseguir asesinatos repercute tan claramente como conseguir torres o objetivos en el resultado de la partida. En la distribución bimodal queda claro que normalmente caes en una montaña o en otra y en función de eso pierdes o ganas, en esta distribución puedes ganar una partida habiendo obtenido pocos asesinatos.

2. Al estudiar las dobles kill por equipos podemos ver:

- La variable es parecida a los inhibidores. Al mirar los histogramas es casi una variable categórica, aunque claramente es numérica porque se pueden conseguir infinitos dobles asesinatos en una partida. La distribución por tanto es exponencial, donde es muy posible que no se consigan 0 asesinatos dobles, y aumentar en 1 el valor de asesinatos dobles es menos probable.
- Al ver el sumario vemos que ambas variables para el equipo azul y el rojo son muy parecidas, con misma mediana y medias muy similares. Ocurre por tanto como con asesinatos, que aunque como ya sabemos que su correlación es positiva para el azul y negativa para el rojo, y que por tanto se consiguen más asesinatos cuando se gana, puede ser que más dobles asesinatos no signifique nada para ganar.

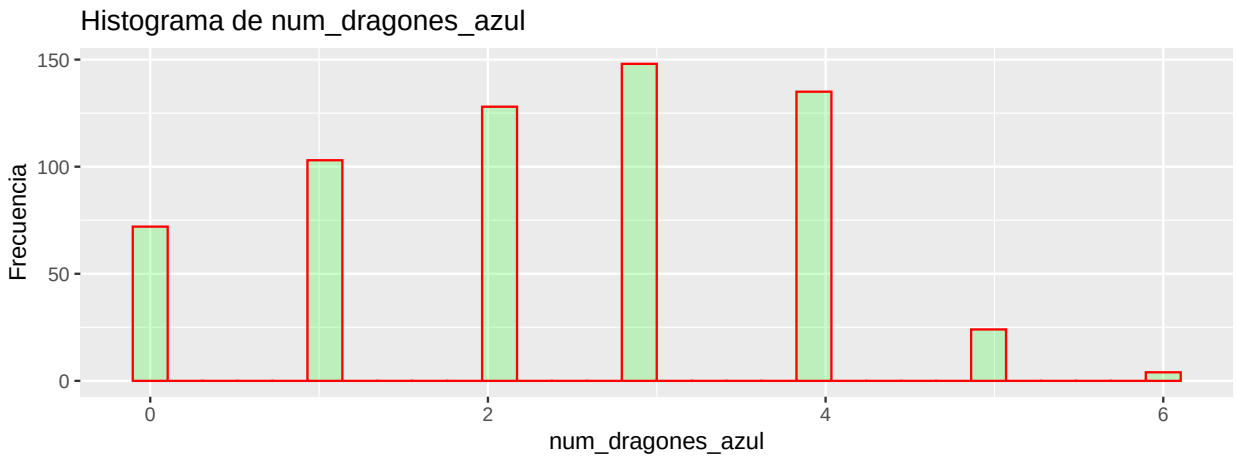
4.3.1.4 Descripción de los monstruos de la jungla

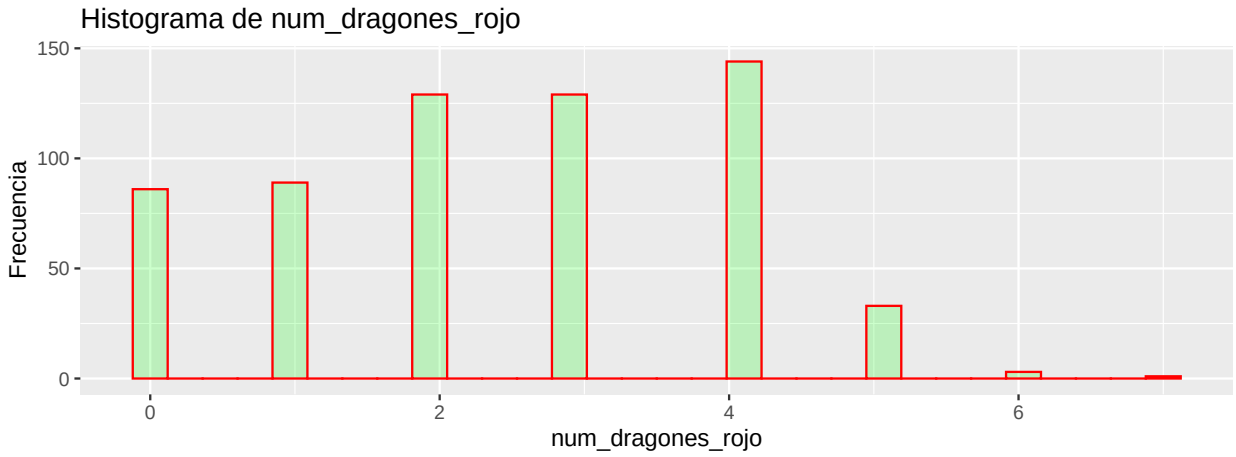
```
vars <- c('num_dragones_azul', 'num_dragones_rojo', 'num_nashors_azul','num_nashors_rojo')

summary(lol_imputed_onehot[vars])
```

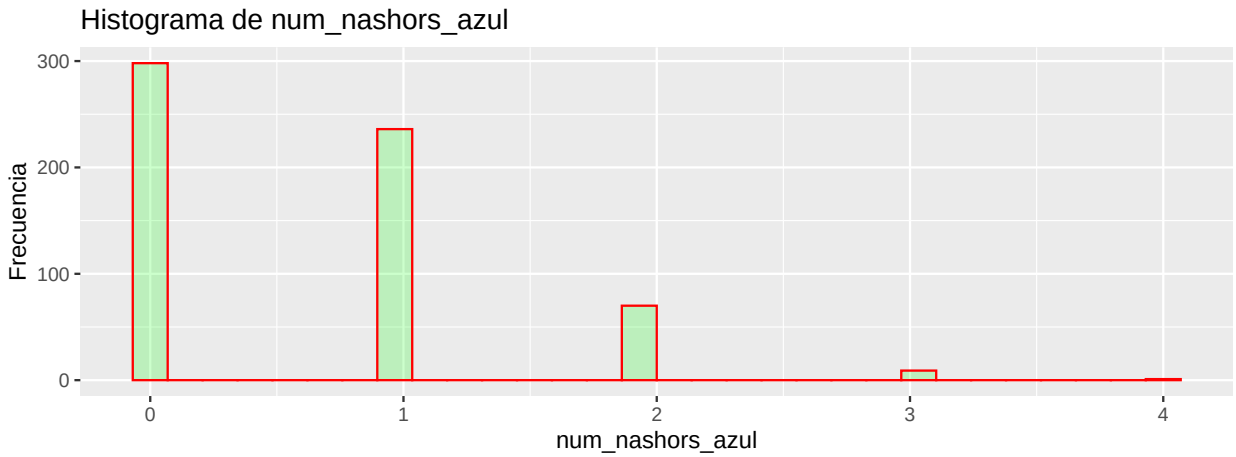
```
##  num_dragones_azul num_dragones_rojo num_nashors_azul num_nashors_rojo
##  Min.   :0.000     Min.   :0.000     Min.   :0.0000   Min.   :0.000
##  1st Qu.:1.000     1st Qu.:1.000     1st Qu.:0.0000   1st Qu.:0.000
```

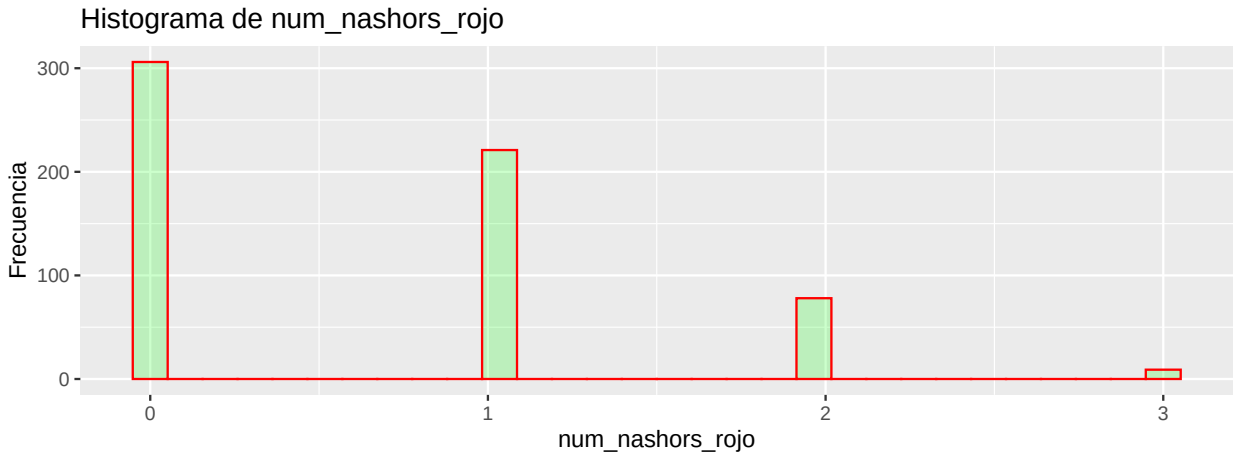
```
##  Median :3.000     Median :3.000     Median :1.0000   Median :1.000
##  Mean   :2.422     Mean   :2.443     Mean   :0.6629   Mean   :0.658
##  3rd Qu.:4.000     3rd Qu.:4.000     3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :6.000     Max.   :7.000     Max.   :4.0000   Max.   :3.000
```

```
invisible(sapply(vars,plot_hist))
```









1. Al ver los dragones de ambos equipos podemos ver que:

- Al contrario que la mayoría de variables, en caso de los dragones la media es superior en el lado rojo. Esto se debe principalmente a algo que ya comentamos antes, y es que el lado rojo tiene ventaja en el lado del dragón. Por lo demás sus valores son parecidos.
- Al observar los histogramas, vemos de nuevo que aunque las variables son numéricas tienen un rango de valores bastante corto. En general destaca que la mayoría de partidas los equipos tienen entre 0 y 4 dragones y es muy dificil que pasen de esta cifra, esto se debe a que a partir de que un equipo tiene 4 dragones, obtiene una ventaja muy grande que hace que las partidas acaben pronto.

2. Al ver los nashor de ambos equipos podemos ver que:

- En este caso, vemos que la media de nashor es superior en el equipo azul, por su mayor capacidad de ganar y por su ventaja. Además vemos que incluso su tercer cuartil en ambos equipos es 1, mientras que la máxima son 4 y 3. Es decir, que normalmente un equipo consigue 0 o 1 nashor por partida, pero que se puede llegar a obtener muchos nashors, esto será normalmente en partidas igualadas.
- Al ver el histograma, de nuevo vemos lo comentado, que normalmente un equipo tiene 0 o 1 nashors y es muy dificil obtener más.

4.3.1.3 Descripción del oro y experiencia

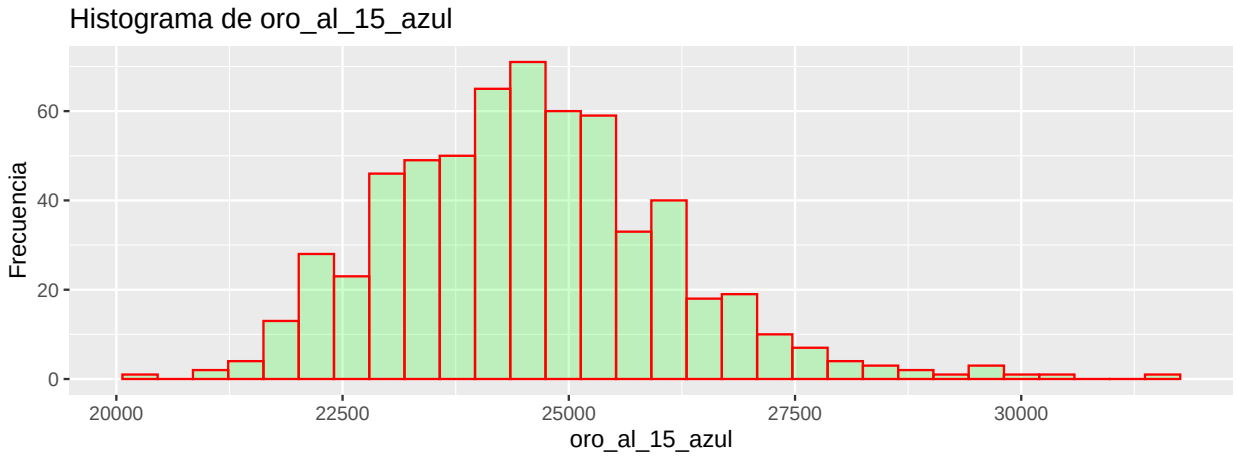
```
colnames(lol_imputed_onehot)
```

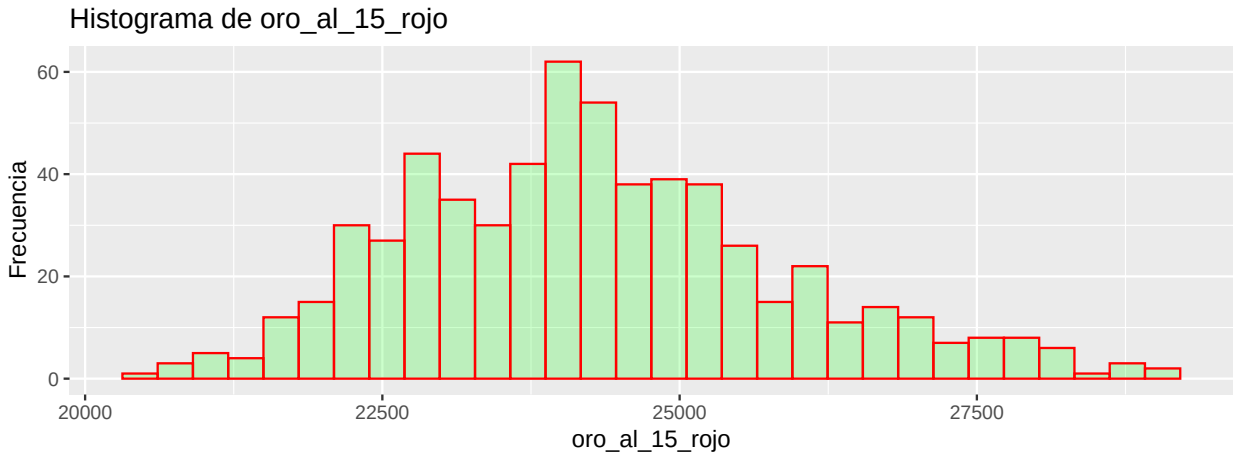
```
##  [1] "num_torres_rojo"       "num_torres_azul"       "DamageObjectives_rojo"
##  [4] "DamageObjectives_azul" "inhibs_rojo"           "kda_adc_rojo"
##  [7] "num_asesinatos_rojo"   "inhibs_azul"           "kda_mid_rojo"
## [10] "kda_adc_azul"          "num_nashors_rojo"      "kda_jng_rojo"
## [13] "kda_mid_azul"          "kda_jng_azul"          "kda_sup_azul"
## [16] "kda_top_rojo"          "kda_sup_rojo"          "num_asesinatos_azul"
## [19] "num_dragones_rojo"     "kda_top_azul"          "num_dragones_azul"
## [22] "num_nashors_azul"      "double_k_rojo"         "diff_oro_al_15"
## [25] "oro_al_15_rojo"        "diferencia_exp"        "oro_al_15_azul"
## [28] "double_k_azul"         "gana_azul"
```

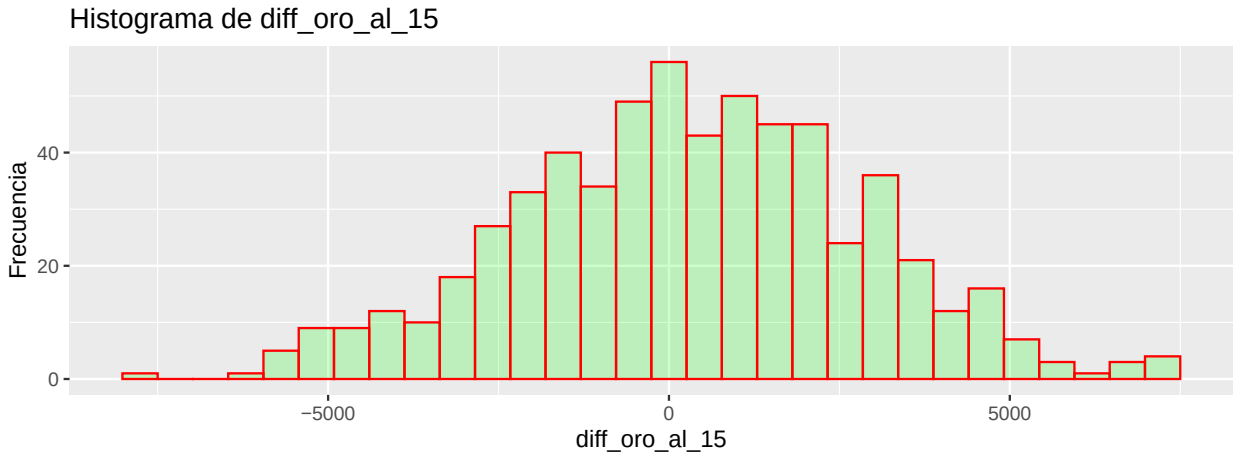
```
vars <- c('oro_al_15_azul', 'oro_al_15_rojo', 'diff_oro_al_15','diferencia_exp')

summary(lol_imputed_onehot[vars])
```

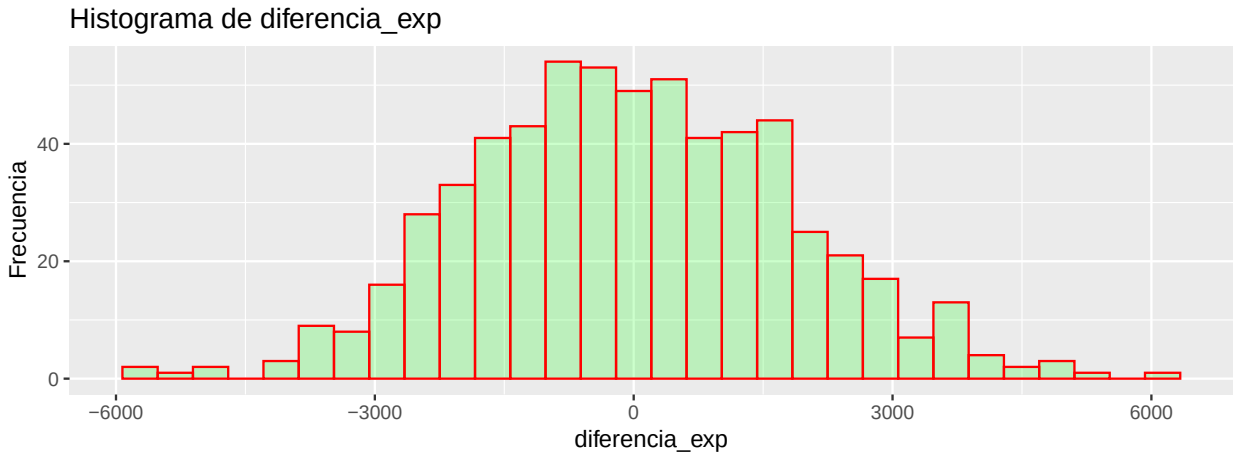
```
##  oro_al_15_azul  oro_al_15_rojo  diff_oro_al_15    diferencia_exp
##  Min.   :20200   Min.   :20500   Min.   :-7800.0   Min.   :-5909.00
##  1st Qu.:23500   1st Qu.:23000   1st Qu.:-1300.0   1st Qu.:-1336.50
##  Median :24500   Median :24100   Median :  298.5   Median :  -85.00
##  Mean   :24566   Mean   :24262   Mean   :  298.3   Mean   :  -33.16
##  3rd Qu.:25475   3rd Qu.:25200   3rd Qu.: 2000.0   3rd Qu.: 1246.50
##  Max.   :31500   Max.   :29100   Max.   : 7200.0   Max.   : 5942.00
```

```
invisible(sapply(vars,plot_hist))
```









1. Al comprobar el oro al minuto 15 de ambos equipos:

- En el sumario podemos comprobar como normalmente la ventaja en media y mediana la tiene el equipo azul, derivado posiblemente de que el equipo azul gana más partidas.
- Por otra parte vemos como el rango entre el mínimo y el máximo son unos 10000 de oro. Por lo tanto es muy importante conseguir cuanto más oro mejor, porque el valor superior es un 50% más del valor mínimo.
- Al ver los histogramas, vemos como a pesar de que el test de saphiro-wilk dió que las variables no son normales, su distribución es bastante normal, y solo está un poco movida al mínimo.

2. Al comprobar la diferencia de oro al minuto 15:

- Vemos como el valor mínimo y máximo son parecidos, lo que indica que el peor equipo rojo y el peor equipo azul han sido parecidos. Lo más importante es ver como el equipo azul normalmente tiene ventaja en esta variable puesto que la mediana y la media son positivas, y esto indica que el equipo azul tiene ventaja.
- La distribución es normal, se aprecia totalmente en el histograma.

4.3.2 Contraste de hipótesis

A continuación realizo varias pruebas de contraste de hipótesis:

4.3.2.1 Contraste de proporción de victoria del azul es superior al rojo

Ya hemos visto que el equipo azul parece que tiene una cierta ventaja, puesto que este equipo ha ganado más que el equipo rojo. Por tanto puede ser interesante estudiar si la diferencia en la variable gana_azul es significativa o no.

Voy a utilizar **un contraste de hipótesis sobre la proporción de victoria en la variable gana_azul**. Las variables binarias, al estudiar su proporción, siguen una distribución de Bernoulli de parametro p, sobre la que quiero hacer un contraste. Cuando la n es grande, sobretodo n>30, esta distribución de Bernoulli, se puede aproximar, mediante las probabilidades a una distribución normal, por esto podemos aplicar los conceptos de la normal z. El contraste es unilateral, puesto que quiero comprobar si la proporcion de victoria del equipo azul es superior a la proporción de victoria del equipo rojo, sino es inferior o igual.

Por tanto, las hipótesis de este contraste son:

$H_0 : p_{gana_azul} = p_{gana_rojo}$ -> $p_{gana_azul} = 1/2$
$H_1 : p_{gana_azul} > p_{gana_rojo}$ -> $p_{gana_azul} > 1/2$

```
prop.test( table(lol_imputed_onehot$gana_azul), conf.level=0.95, alternative = 'greater')
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  table(lol_imputed_onehot$gana_azul), null probability 0.5
## X-squared = 1.1873, df = 1, p-value = 0.1379
## alternative hypothesis: true p is greater than 0.5
## 95 percent confidence interval:
##  0.4888048 1.0000000
## sample estimates:
##         p
## 0.5228013
```

El p-valor es 0.1379, por lo que rechazamos la hipótesis alternativa y no podemos rechazar la hipótesis nula. **De esta manera se concluye que la proporción de victoria del equipo azul no es superior a la proporción de victoria del equipo rojo con un 95% de confianza.** De hecho, vemos que el intervalo de confianza es de 0.4888 hasta 1, solo si fuera de más de 0.5 hasta 1, podríamos decir que la proporción de victoria del equipo azul es superior.

Por tanto, a pesar de que en nuestro dataset hay más victorias del equipo azul, no podemos asegurar que el equipo azul tiene una ventaja solo por ser lado azul.

4.3.2.2 Comprobación de que el daño a objetivos por parte del equipo azul es superior cuando gana a cuando pierde.

Una vez que ya hemos estudiado la diferencia entre victorias, podemos pasar a estudiar aspectos de la partida en función de si un equipo o otro gana. En este caso **vamos a estudiar un contraste de hipótesis sobre la variable DamageObjectives_azul en función de los grupos de gana_azul.**

Ya hemos realizado las pruebas de normalidad y de homocedasticidad. Para esta variable, ya hemos comprobado que su distribución no es normal (mediante la prueba de Saphiro-Wilk), de hecho, al estudiar su histograma hemos visto que la distribución es bimodal. Por otra parte, hemos visto mediante el test de fligner que su varianza en función del grupo de gana_azul es igual entre los grupos.

Por tanto, ahora podemos aplicar el test no paramétrico Wilcoxon y comprobar si la distribución de daño a objetivos por parte del equipo azul en función del ganador es la misma o no. Además, como la distribución de varianza es la misma, si la distribución de un grupo es mayor, su mediana será mayor.

En el test de Wilcoxon, las hipotesis se fundamentan en estudiar la posibilidad de que un elemento de una población sea mayor que un elemento de otra población. En caso de que tengamos dos poblaciones iguales, la probabilidad para ambas poblaciones será de 0.5. Por tanto, nosotros queremos comprobar si las posibilidades de que el daño a objetivos por parte del equipo azul cuando gana sean mayores que cuando pierde, o que sean iguales estas posibilidades.

$H_0 : P(x_i > y_i) = 0.5$
$H_1 : P(x_i > y_i) > 0.5$

```
gana<-lol_imputed_onehot[which(lol_imputed_onehot$gana_azul == 'si'),]
pierde<-lol_imputed_onehot[which(lol_imputed_onehot$gana_azul == 'no'),]

wilcox.test(gana$DamageObjectives_azul, pierde$DamageObjectives_azul, alternative = "greater", mu = 0,
            paired = FALSE, conf.int = 0.95)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  gana$DamageObjectives_azul and pierde$DamageObjectives_azul
## W = 91050, p-value < 2.2e-16
## alternative hypothesis: true location shift is greater than 0
## 95 percent confidence interval:
##  1304.167      Inf
## sample estimates:
## difference in location
##               1369.415
```

```
intervalo<-wilcox.test(gana$DamageObjectives_azul, pierde$DamageObjectives_azul, alternative = "greater
```

Como vemos, el p-valor es muy muy pequeño, por lo que se rechaza la hipótesis nula y se acepta la hipótesis alternativa. Por tanto, **la distribución del daño del equipo azul a objetivos es superior que la distribución del daño a objetivos del equipo rojo con un 95% de confianza.** Algo que en realidad, ya sospechabamos al ver la correlación entre las variables. Ahora hemos comprobado que hay significancia estadística.

Esto lo comprobamos más adelante en el apartado 5, al observar la distribución de la variable en función de los grupos que se generan por gana_azul y al ver que las medianas son diferentes.

4.3.2.3 Comprobación de si la diferencia de experiencia en el minuto 15 es igual independientemente de si gana o pierde el equipo azul.

Otro aspecto interesante de la partida respecto a quién gana es la diferencia de experiencia. En este caso **vamos a estudiar un contraste de hipótesis sobre la variable diferencia_exp en función de los grupos de gana_azul.**

Ya hemos realizado las pruebas de normalidad y de homocedasticidad. Para esta variable, hemos comprobado mediante el test de Saphiro-Wilk que tiene una distribución normal y mediante el test de Levene que la varianza en función del grupo gana_azul es igual entre los grupos.

Por tanto, ahora podemos aplicar el t test para estudiar si la media de diferencia de experiencia para un grupo es la misma que la media de diferencia de experiencia para el otro grupo. En caso de que sean iguales, será que la diferencia de experiencia es parecida independientemente de quién gane y que por tanto no importa tener una ventaja de experiencia en la partida.

Las hipótesis son:

$H_0 : \mu_1 = \mu_2$ -> $\mu_1 - \mu_2 = 0$
$H_1 : \mu_1 \neq \mu_2$ -> $\mu_1 - \mu_2 \neq 0$

```
t.test(gana$diferencia_exp, pierde$diferencia_exp, alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  gana$diferencia_exp and pierde$diferencia_exp
## t = 14.83, df = 611.85, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1653.865 2158.751
## sample estimates:
##  mean of x  mean of y
##   876.5296 -1029.7782
```

```
intervalo2<-t.test(gana$diferencia_exp, pierde$diferencia_exp, alternative = "two.sided")$conf.int
```

Vemos que el p-valor es 2.2e-16, es decir, muy pequeño. Esto significa que hemos de rechazar la hipótesis nula y no podemos rechazar la hipótesis alternativa. Por tanto **la media de diferencia de experiencia cuando gana el equipo azul, es diferente a cuando gana el equipo rojo**. Este resultado es lógico puesto que la variable ha superado el filtro de correlación, y por tanto ya sabíamos que la diferencia de experiencia tenía una correlación positiva con el equipo azul para ganar.

4.3.2.4 Contraste de diferencia de KDA entre posiciones para el equipo ganador.

A continuación, **voy a estudiar un contraste de hipótesis entre más de dos grupos, concretamente la diferencia de KDA entre las diferentes posiciones, pero solo para el equipo ganador de la partida.**

De esta manera, primero hemos de generar el dataset para estudiar esto.

```
todos_kdas<-gana['kda_top_azul']
colnames(todos_kdas)<-'kda'
todos_kdas$posicion<-as.factor('top')

variable<-gana['kda_jng_azul']
colnames(variable)<-'kda'
variable$posicion<-as.factor('jng')
todos_kdas<-rbind(todos_kdas, variable)

variable<-gana['kda_mid_azul']
colnames(variable)<-'kda'
variable$posicion<-as.factor('mid')
todos_kdas<-rbind(todos_kdas, variable)

variable<-gana['kda_adc_azul']
colnames(variable)<-'kda'
variable$posicion<-as.factor('adc')
todos_kdas<-rbind(todos_kdas, variable)

variable<-gana['kda_sup_azul']
```

```
colnames(variable)<-'kda'
variable$posicion<-as.factor('sup')
todos_kdas<-rbind(todos_kdas, variable)

variable<-pierde['kda_top_rojo']
colnames(variable)<-'kda'
variable$posicion<-as.factor('top')
todos_kdas<-rbind(todos_kdas, variable)

variable<-pierde['kda_jng_rojo']
colnames(variable)<-'kda'
variable$posicion<-as.factor('jng')
todos_kdas<-rbind(todos_kdas, variable)

variable<-pierde['kda_mid_rojo']
colnames(variable)<-'kda'
variable$posicion<-as.factor('mid')
todos_kdas<-rbind(todos_kdas, variable)

variable<-pierde['kda_adc_rojo']
colnames(variable)<-'kda'
variable$posicion<-as.factor('adc')
todos_kdas<-rbind(todos_kdas, variable)

variable<-pierde['kda_sup_rojo']
colnames(variable)<-'kda'
variable$posicion<-as.factor('sup')
todos_kdas<-rbind(todos_kdas, variable)
```

Una vez se ha generado el dataset, con una variable que tiene el kda y otra la posición para los jugadores de los equipos que ganan. Vamos a estudiar la normalidad y la homocedasticidad de las variables.

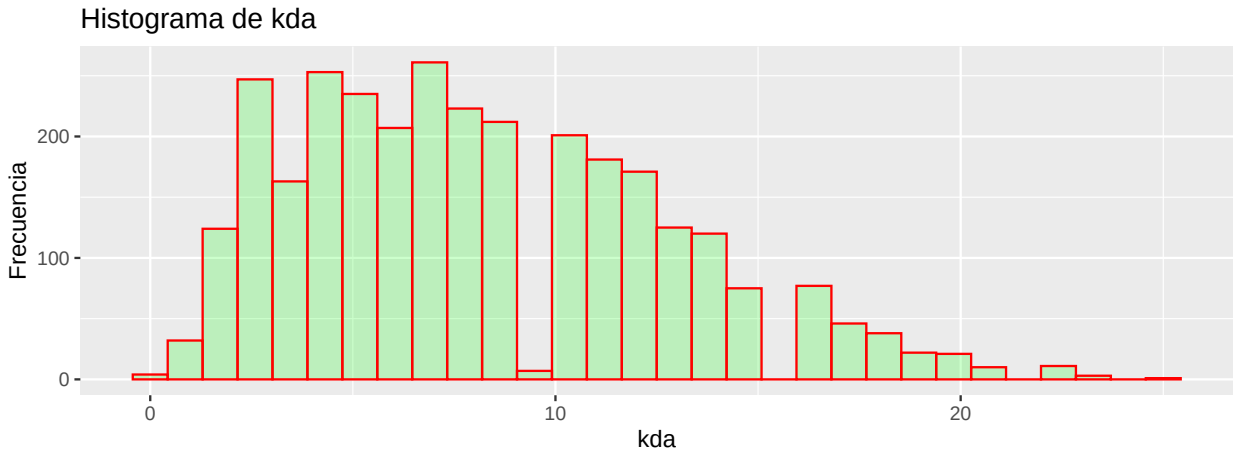
```
shapiro.test(todos_kdas$kda)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  todos_kdas$kda
## W = 0.95847, p-value < 2.2e-16
```

Vemos que el shapiro test indica que el p-valor es muy pequeño, y por tanto, que la distribución de la variable no es normal. Si estudiamos la distribución mediante un histograma:

```
plot_hist_2<-function(x)
{
  print(ggplot(data=todos_kdas, aes(x=unlist(todos_kdas[x]))) +
  geom_histogram(
                col="red",
                fill="green",
                alpha = .2) +
  labs(title=paste0("Histograma de ",x)) +
  labs(x=x, y="Frecuencia"))
```

```
}

plot_hist_2('kda')
```



Vemos que claramente, la distribución no es normal ya que podría ser una distribución desviada hacía los valores menores. Donde evidentemente, vemos como los valores que más aparacen para los kda de los equipos ganadores están entre 3 y 9.

A continuación estudiamos la varianza mediante un test de fligner para estudiar si hay homocedasticidad de las varianzas.

```
fligner.test(kda ~ posicion, data = todos_kdas)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  kda by posicion
## Fligner-Killeen:med chi-squared = 21.297, df = 4, p-value = 0.0002765
```

Vemos que el p-valor es menor de 0.05, por tanto podemos concluir que no hay homocedasticidad entre las varianzas de kda en función de la posición de cada jugador.

Debido a estos resultados, vamos a estudiar si la distribución del kda a lo largo de las diferentes posiciones es la misma para los equipos que ganan. Y lo realizamos con un kruskal.test, que es un test no paramétrico.

El test de Kruskal-Wallis, es un test no paramétrico con las siguientes hipotesis: H_0 : Las 5 distribuciones vienen de la misma población. H_1 : Las 5 distribuciones NO vienen de la misma población.

```
kruskal.test(kda ~ posicion, data = todos_kdas)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  kda by posicion
## Kruskal-Wallis chi-squared = 125.96, df = 4, p-value < 2.2e-16
```

Al obtener un p-valor tan inferior a 0.05 podemos concluir con un 95% de confianza que rechazamos la hipótesis nula y se acepta la hipótesis alternativa. **Por tanto, las 5 distribuciones de KDA en función**

de la posición del jugador no vienen de la misma población. Es decir, que las distribuciones son diferentes.

Una vez hemos obtenido esto, puede ser interesante estudiar las diferencias en función de los grupos de KDA por posición, para encontrar que distribución de KDA es significativamente diferente respecto a las otras.

Para ello se puede realizar un pairwise wilcoxon test, donde nos proporciona un p-valor para cada combinación de grupo de KDA en función de la posición.

```
pairwise.wilcox.test(x = todos_kdas$kda, g = todos_kdas$posicion )
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  todos_kdas$kda and todos_kdas$posicion
##
##     top     jng     mid   adc
## jng 1.2e-09 -       -     -
## mid 1.8e-14 0.208   -     -
## adc < 2e-16 5.8e-05 0.030 -
## sup 1.6e-05 0.208   0.009 1.2e-07
##
## P value adjustment method: holm
```

Vemos como el kda del top, es significativamente diferente respecto a los otros, igual que ocurre para el adc (utilizando una significancia del 95%). Por otro lado, el jng no es significativamente diferente ni a mid ni a sup, pero entre mid y sup si que son significativamente diferentes. De hecho, si vemos por el p-valor, el mid está tiene un p-valor mayor para igualdad con el adc que con el sup.

Por tanto, aplicando la lógica, el kda del top es significativamente inferior que el resto, el del adc significativamente superior. El del jungla no se diferencia significativamente del mid ni del support, pero el mid sí que del support, pareciendose más al del adc. Por tanto, un orden lógico derivado de estos p-valores es: top<sup<=jng<=mid<adc

4.3.3 Regresión logística para predecir que equipo gana.

Ahora que ya hemos realizado varios contrastes de hipótesis y conocemos un poco más sobre el dataset y sus comparaciones, **podemos desarrollar un modelo de regresión logística para predecir el ganador de la partida.**

Vamos a hacer por tanto un modelo supervisado, con su conjunto de entrenamiento y test, donde la variable objetivo sea gana_azul, y el resto de variables sean variables independientes en el modelo.

Primero hay que subdividir el dataset en entrenamiento y test:

```
require(caret)
require(questionr)
library(sjPlot)
library(sjlabelled)
library(sjmisc)
require(MASS)

lol_imputed_onehot$gana_azul <- factor(lol_imputed_onehot$gana_azul, levels=c("no", "si"))
```

```r
set.seed(42)
trainIndex <- createDataPartition(lol_imputed_onehot$gana_azul, p = .66,
                                  list = FALSE,
                                  times = 1)

Train <- lol_imputed_onehot[ trainIndex,]
Test  <- lol_imputed_onehot[-trainIndex,]
```

A continuación se entrena el modelo y estudiamos su summary.

```r
glm_train<- glm(gana_azul ~ ., data=Train, family = "binomial")

summary(glm_train)
```

```
##
## Call:
## glm(formula = gana_azul ~ ., family = "binomial", data = Train)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.013e-05  -2.100e-08   2.100e-08   2.100e-08   5.763e-05
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -1.056e+02  1.534e+06   0.000    1.000
## num_torres_rojo       -1.197e+01  1.819e+04  -0.001    0.999
## num_torres_azul        2.019e+01  1.370e+04   0.001    0.999
## DamageObjectives_rojo -3.507e-02  1.048e+02   0.000    1.000
## DamageObjectives_azul -4.814e-02  5.524e+01  -0.001    0.999
## inhibs_rojo           -5.718e+00  4.465e+04   0.000    1.000
## kda_adc_rojo           2.081e+00  6.018e+03   0.000    1.000
## num_asesinatos_rojo   -3.877e+00  1.912e+04   0.000    1.000
## inhibs_azul           -1.128e+01  4.871e+04   0.000    1.000
## kda_mid_rojo          -6.448e+00  6.566e+03  -0.001    0.999
## kda_adc_azul           3.110e+00  9.403e+03   0.000    1.000
## num_nashors_rojo       2.434e+01  5.307e+04   0.000    1.000
## kda_jng_rojo           2.263e+00  6.872e+03   0.000    1.000
## kda_mid_azul          -2.402e+00  1.126e+04   0.000    1.000
## kda_jng_azul          -1.548e+00  1.618e+04   0.000    1.000
## kda_sup_azul          -2.032e+00  5.494e+03   0.000    1.000
## kda_top_rojo          -1.984e+00  1.178e+04   0.000    1.000
## kda_sup_rojo           3.033e+00  1.032e+04   0.000    1.000
## num_asesinatos_azul    4.831e+00  2.267e+04   0.000    1.000
## num_dragones_rojo     -3.751e-01  2.607e+04   0.000    1.000
## kda_top_azul          -1.418e+00  5.601e+03   0.000    1.000
## num_dragones_azul      9.121e+00  2.820e+04   0.000    1.000
## num_nashors_azul       7.735e+00  2.466e+04   0.000    1.000
## double_k_rojo         -1.118e+01  1.739e+04  -0.001    0.999
## diff_oro_al_15         1.410e-02  6.537e+01   0.000    1.000
## oro_al_15_rojo         8.964e-03  6.062e+01   0.000    1.000
## diferencia_exp        -1.708e-02  1.622e+01  -0.001    0.999
## oro_al_15_azul        -1.100e-03  7.711e+01   0.000    1.000
## double_k_azul         -2.375e+00  4.558e+04   0.000    1.000
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5.6204e+02  on 405  degrees of freedom
## Residual deviance: 3.7064e-08  on 377  degrees of freedom
## AIC: 58
##
## Number of Fisher Scoring iterations: 25
```

Al ver el summary, vemos que la métrica AIC es muy pequeña, por lo tanto el ajuste del modelo a la variable objetivo es bastante bueno y además, que la desviación de residuales es ínfima. Eso sí, también se ha de comentar que se han realizado 25 iteraciones de Fisher, lo que indica junto con una desviación residual tan pequeña pero un AIC pequeño pero no de 0, que la complejidad del modelo es bastante grande respecto a la bondad del ajuste.

Esto también se puede apreciar en el p-valor de las variables. Y es que todos los p-valores son 1 o 0.999. Es decir, que el modelo puede hacer una predición buena, pero el coeficiente de regresión de las variables no es nada seguro. En definitiva, esto se debe a que tenemos varias variables con una correación muy alta, algunas incluso casi separan perfectamente el ganador.

Realizo las predicciones del modelo para estudiar su desempeño, no estudio sus odds ratio porque los p-valores son todos de 1. Además obtengo su curva roc.

```
predicciones<- predict(glm_train,Test, type="response")
predicciones <-as.factor(ifelse(predicciones>0.5,'si','no'))

confmatrix<- confusionMatrix(predicciones,Test$gana_azul, positive = 'si' )
confmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no  si
##         no  96   2
##         si   3 107
##
##                Accuracy : 0.976
##                  95% CI : (0.9448, 0.9921)
##     No Information Rate : 0.524
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9518
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9817
##             Specificity : 0.9697
##          Pos Pred Value : 0.9727
##          Neg Pred Value : 0.9796
##              Prevalence : 0.5240
##          Detection Rate : 0.5144
##    Detection Prevalence : 0.5288
##       Balanced Accuracy : 0.9757
##
```

```
##        'Positive' Class : si
##
```

Al estudiar los resultados del modelo y su matriz de confianza, vemos como el acierto es altísimo en el conjunto de test, de un 97.6% de acierto, con una sensitividad del 98.17 para predecir si el equipo azul gana la partida. En total se equivoca solo en 5 partidas de las más de 200 del conjunto de test.

Por tanto, hemos obtenido un modelo muy bueno para predecir, pero que no podemos estudiar sus odds ratio, y por tanto no podemos estudiar las variables que influyen en esta predicción.

4.3.3.1 Regresión logística para identificar los factores que influyen en la victoria de un equipo.

Para poder estudiar las variables que influyen en la predicción, voy de nuevo a realizar una regresión logística donde en vez de predecir si gana el equipo azul o rojo, se predice si un equipo gana o pierde, convirtiendo el dataset a un dataset más sencillo, con menos variables, donde cada instancia son los valores de un equipo.

Generamos este dataset y hago la partición en train y test:

```
azul<-lol_imputed_onehot %>% dplyr::select(contains("azul")| contains("exp")| contains("diff"))
colnames(azul)<-str_replace(colnames(azul),'_azul','')
rojo<-lol_imputed_onehot %>% dplyr::select(contains("rojo")| contains("gana") | contains("exp")| contai
colnames(rojo)<-str_replace(colnames(rojo),'_rojo','')
colnames(rojo)<-str_replace(colnames(rojo),'_azul','')
rojo$gana<- ifelse(rojo$gana == 'si', 'no', 'si')
rojo$diferencia_exp<- -(rojo$diferencia_exp)
rojo$diff_oro_al_15<- -(rojo$diff_oro_al_15)


equipos <-rbind(azul, rojo)

set.seed(42)
trainIndex <- createDataPartition(equipos$gana, p = .66,
                                  list = FALSE,
                                  times = 1)

Train2 <- equipos[ trainIndex,]
Test2  <- equipos[-trainIndex,]
```

Ahora genero el modelo y estudio su summary.

```
glm_train2<- glm(gana ~ ., data=Train2, family = "binomial")

summary(glm_train2)
```

```
##
## Call:
## glm(formula = gana ~ ., family = "binomial", data = Train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2173  -0.0057  -0.0001   0.0199   2.4341
##
## Coefficients:
```

```
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.056e+01  9.641e+00  -2.133  0.03292 *
## num_torres        1.517e+00  3.243e-01   4.677 2.91e-06 ***
## DamageObjectives  1.008e-03  8.584e-04   1.174  0.24038
## inhibs           -5.631e-01  4.438e-01  -1.269  0.20452
## kda_adc           2.413e-01  8.975e-02   2.689  0.00717 **
## kda_mid           1.706e-01  1.272e-01   1.341  0.17980
## kda_jng           4.508e-01  1.606e-01   2.807  0.00501 **
## kda_sup           3.542e-01  1.168e-01   3.033  0.00242 **
## num_asesinatos   -8.411e-02  7.126e-02  -1.180  0.23783
## kda_top           1.804e-01  1.174e-01   1.537  0.12426
## num_dragones      1.745e-01  3.435e-01   0.508  0.61136
## num_nashors      -4.541e-01  4.451e-01  -1.020  0.30753
## oro_al_15         9.864e-05  3.731e-04   0.264  0.79151
## double_k          6.058e-01  3.959e-01   1.530  0.12598
## diferencia_exp    1.270e-05  2.637e-04   0.048  0.96158
## diff_oro_al_15    9.383e-05  2.692e-04   0.349  0.72743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1125.671  on 811  degrees of freedom
## Residual deviance:   81.247  on 796  degrees of freedom
## AIC: 113.25
##
## Number of Fisher Scoring iterations: 10
```

Ahora vemos que aunque la bondad del ajuste no es tan buena como antes, ya que el AIC y la desviación de residuales son mayores, la diferencia entre ellos no es tán grande y las iteraciones de Fisher son solo 10. Por lo que ahora la bondad ha disminuido, pero la complejidad del modelo ha disminuido mucho, hasta el punto de que ahora es interpretable.

En el summary podemos ver como los p-valores ya no son de 1, y que hay varias variables que salen significativas, estas variables son el número de torres, el kda del adc, el kda del jng, el kda del sup y el kda del top. Eso significa que el número de torres conseguido y el kda de adc, jng, sup y top es muy influyente en la partida, y por tanto para asegurarse una victoria tendremos que centrarnos en estas variables.

De esta manera, estas variables son las que sin duda influyen en la predicción de manera significativa. Pero aun así, podemos estudiar los odds ratio de todas las variables. Esto lo realizamos en el punto 5. Ahora, realizo las predicciones del modelo para evaluar si ha disminuido mucho la capacidad de acierto sobre el conjunto de test.

```
predicciones<- predict(glm_train2,Test2, type="response")
predicciones <-as.factor(ifelse(predicciones>0.5,'si','no'))

confmatrix<- confusionMatrix(predicciones,Test2$gana , positive = 'si' )
confmatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no  si
##         no 203   8
```

```
##         si   5 200
##
##                Accuracy : 0.9688
##                  95% CI : (0.9472, 0.9833)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9375
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.9615
##             Specificity : 0.9760
##          Pos Pred Value : 0.9756
##          Neg Pred Value : 0.9621
##              Prevalence : 0.5000
##          Detection Rate : 0.4808
##    Detection Prevalence : 0.4928
##       Balanced Accuracy : 0.9688
##
##        'Positive' Class : si
##
```

Vemos como el acierto del modelo es de 96.88%, menos de un punto de diferencia con el modelo anterior, pero en este caso con este modelo hemos podido identificar que variables son las más influyentes en la predicción del modelo. El modelo por tanto es muy bueno en identificar si un equipo va a ganar independientemente de lo que ha conseguido el equipo contrario.

5. Representación de los resultados a partir de tablas y gráficas.

Ya hemos realizado muchos análisis estadísticos, pero ahora es importante representar los resultados obtenidos en gráficas y tablas.

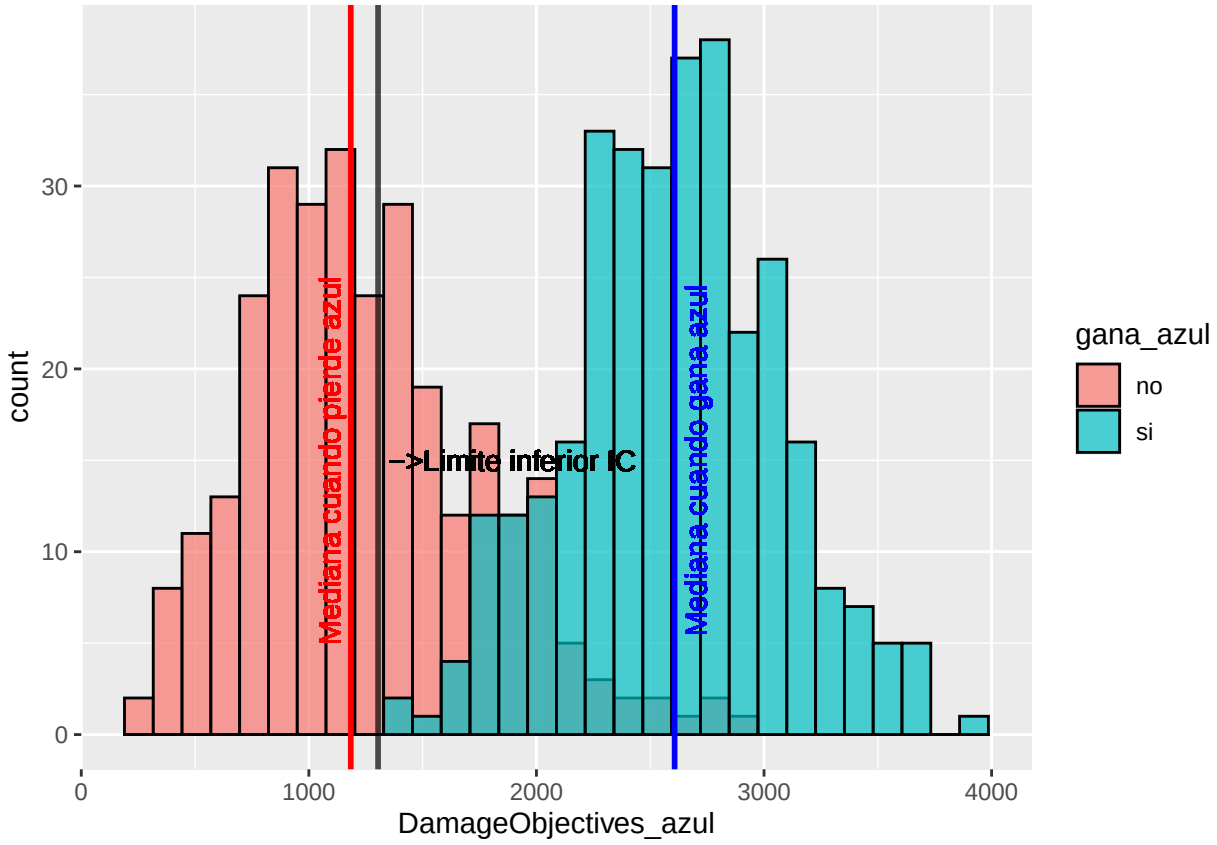
Es importante destacar que los gráficos para las variables del dataset final, que hacen referencia a la distribución de la variables y que son útiles para el análisis estadístico descriptivo ya se han realizado en el punto 4. A continuación desarrollo los gráficos para el resto de puntos.

Por otra parte, en este punto se incluyen los gráficos referentes a los contrastes de hipótesis realizados, y a los odds ratio y curvas ROC de la regresiones logísticas.

5.1 Compración distribución de la variable DamageObjectivos_azul en función del grupo gana_azul

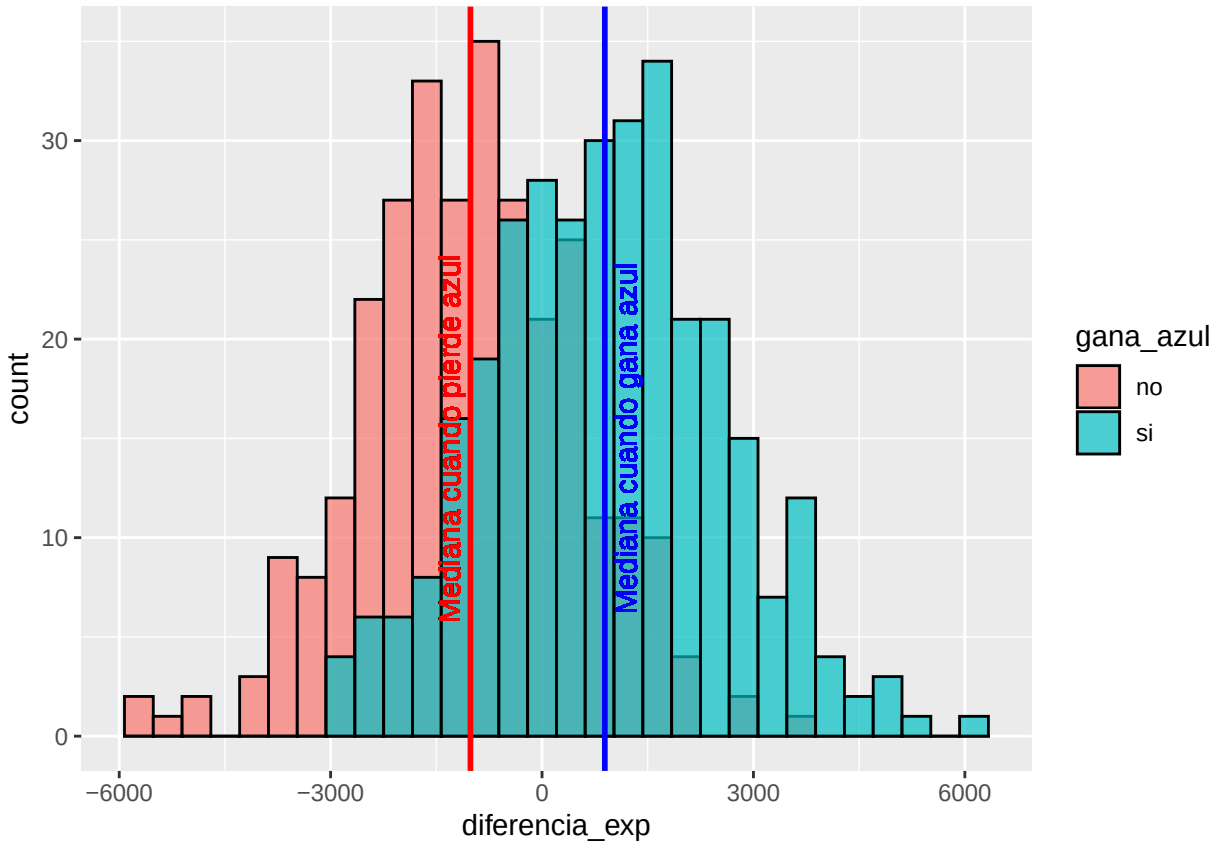
```
#
# lol_imputed_onehot$gana_azul<-factor(lol_imputed_onehot$gana_azul, levels = rev(levels(lol_imputed_on

lol_imputed_onehot %>%
  ggplot(aes(x=DamageObjectives_azul, fill =gana_azul )) +
  geom_histogram(color="black", alpha=0.7, position = 'identity')+
  geom_vline(xintercept = median(gana$DamageObjectives_azul), color='blue', size=1)+
  geom_text(aes(x=median(gana$DamageObjectives_azul), label="\nMediana cuando gana azul", y=15), colour=
```

```
  geom_vline(xintercept = median(pierde$DamageObjectives_azul), color='red', size=1)+
  geom_text(aes(x=median(pierde$DamageObjectives_azul), label="Mediana cuando pierde azul\n", y=15), col
  geom_vline(xintercept = intervalo, color='black', size=1, alpha = 0.7)+
  geom_text(aes(x=intervalo, label="                                    ->Limite inferior IC ", y=15), col
```



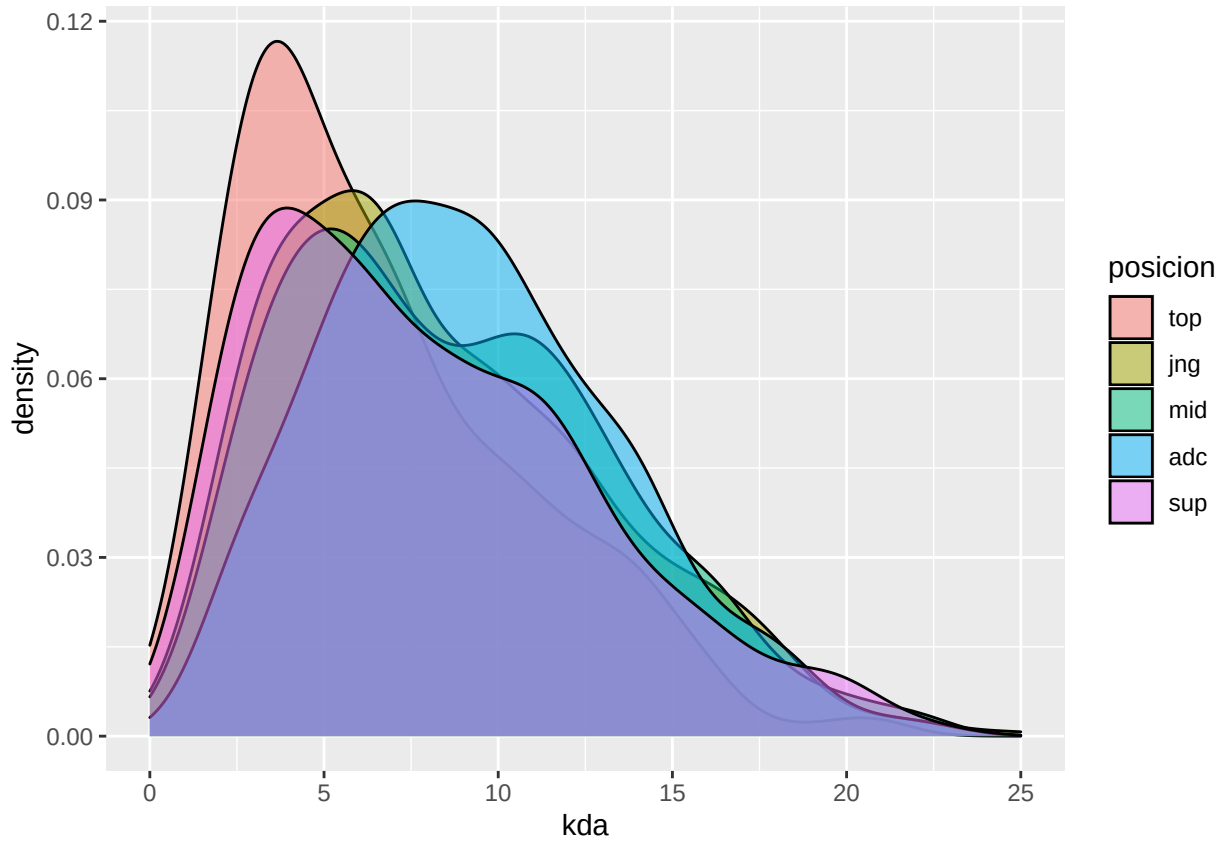
5.2 Compración distribución de la variable diferencia_exp en función del grupo gana_azul

```
lol_imputed_onehot %>%
  ggplot(aes(x=diferencia_exp, fill =gana_azul )) +
  geom_histogram(color="black", alpha=0.7, position = 'identity')+
  geom_vline(xintercept = median(gana$diferencia_exp), color='blue', size=1)+
  geom_text(aes(x=median(gana$diferencia_exp), label="\nMediana cuando gana azul", y=15), colour="blue"
  geom_vline(xintercept = median(pierde$diferencia_exp), color='red', size=1)+
  geom_text(aes(x=median(pierde$diferencia_exp), label="Mediana cuando pierde azul\n", y=15), colour="r
```



5.3 Compración distribución de la variable kda en función de la posición

```
todos_kdas %>%
  ggplot(aes(x=kda, fill =posicion )) +
  geom_density(color="black", alpha=0.5, position = 'identity')
```



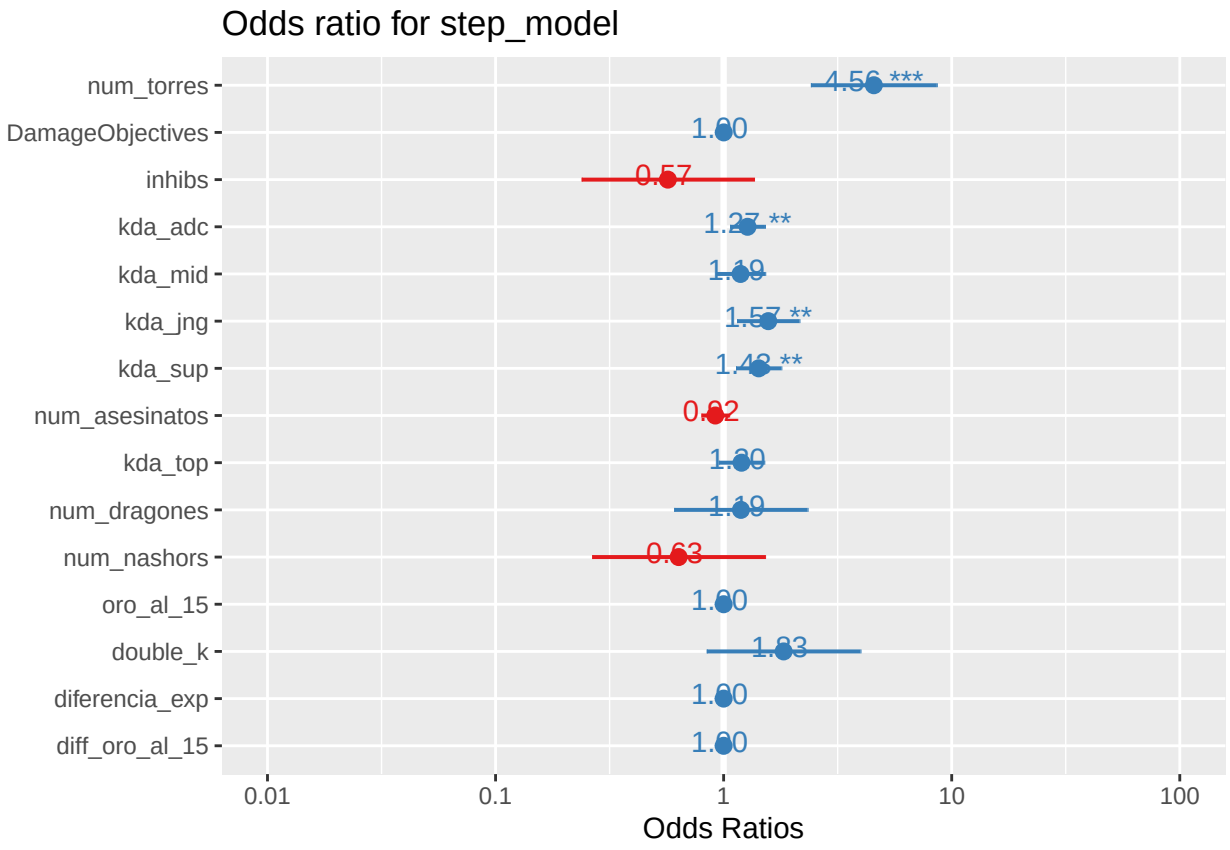
5.4 Curva ROC modelo de regresión para predecir que equipo gana (primera regresión)

```
require(pROC)
require(cvAUC)

# Se obtiene la predicción de los bebés sobre si tienen bajo peso o no.
prob <- predict(glm_train,Test, type="response")

# Se obtiene un objeti de predicción que tiene información sobre los tp, tn, fp y fn.
ROCRpred <- prediction(prob, Test$gana_azul)

# Se obtiene un objeto performance con el tpr y fpr
ROCRperf <- performance(ROCRpred, 'tpr', 'fpr')

# A partir del objeto prediction anterior, obtenemos la auc
auc <- performance(ROCRpred, measure = "auc")

auc <- auc@y.values[[1]]

print(paste0("El area bajo la curva ROC es de ", auc))
```

```
## [1] "El area bajo la curva ROC es de 0.989157631359466"
```

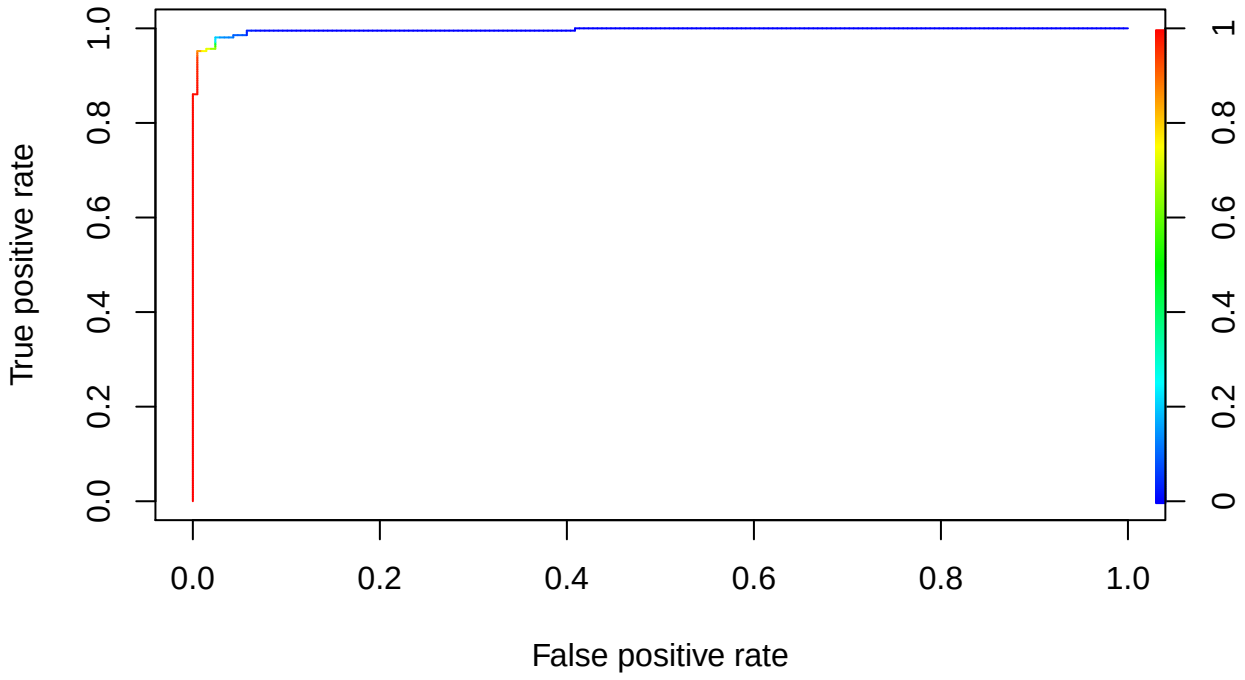
5.5 Odds ratio para la regresión para predecir si un equipo gana (segunda regresión)

```
plot_model(glm_train2, show.values = TRUE, value.offset = .1,  title = "Odds ratio for step_model")
```



5.6 Curva ROC modelo de regresión para predecir si un equipo gana (segunda regresión)

```
# Se obtiene la predicción de los bebés sobre si tienen bajo peso o no.
prob <- predict(glm_train2,Test2, type="response")

# Se obtiene un objeti de predicción que tiene información sobre los tp, tn, fp y fn.
ROCRpred <- prediction(prob, Test2$gana)

# Se obtiene un objeto performance con el tpr y fpr
ROCRperf <- performance(ROCRpred, 'tpr', 'fpr')

# Se dibuja la curva roc
plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2, 1.7))
```




```r
# A partir del objeto prediction anterior, obtenemos la auc
auc <- performance(ROCRpred, measure = "auc")

auc <- auc@y.values[[1]]

print(paste0("El area bajo la curva ROC es de ", auc))
```

```
## [1] "El area bajo la curva ROC es de 0.996186205621302"
```