Cómo ganar al League of Legends

May 17th, 2023

Análisis en R de las variables más importantes para ganar una partida

# Introducción

El League of Legends ha arruinado mi vida y va a arruinar los siguientes 20 minutos que dediques a leerte esto:

# Descripción del problema:

# Análisis en R

## Librerías y semilla:

A continuación descargamos las librerías y establecemos una semilla para garantizar que se puede repetir

## Carga de datos:

library(readr)  
df <- read\_csv("data/high\_diamond\_ranked\_10min.csv")

## Rows: 9879 Columns: 40  
## ── Column specification ────────────────────────────────────  
## Delimiter: ","  
## dbl (40): gameId, blueWins, blueWardsPlaced, blueWardsDe...  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

# Exploración y limpieza de los datos

## Primer contacto

head(df)

## # A tibble: 6 × 40  
## gameId blueW…¹ blueW…² blueW…³ blueF…⁴ blueK…⁵ blueD…⁶  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 4519157822 0 28 2 1 9 6  
## 2 4523371949 0 12 1 0 5 5  
## 3 4521474530 0 15 0 0 7 11  
## 4 4524384067 0 43 1 0 4 5  
## 5 4436033771 0 75 4 0 6 6  
## 6 4475365709 1 18 0 0 5 3  
## # … with 33 more variables: blueAssists <dbl>,  
## # blueEliteMonsters <dbl>, blueDragons <dbl>,  
## # blueHeralds <dbl>, blueTowersDestroyed <dbl>,  
## # blueTotalGold <dbl>, blueAvgLevel <dbl>,  
## # blueTotalExperience <dbl>,  
## # blueTotalMinionsKilled <dbl>,  
## # blueTotalJungleMinionsKilled <dbl>, …

Miramos los vacíos

colSums(is.na(df))

## gameId blueWins   
## 0 0   
## blueWardsPlaced blueWardsDestroyed   
## 0 0   
## blueFirstBlood blueKills   
## 0 0   
## blueDeaths blueAssists   
## 0 0   
## blueEliteMonsters blueDragons   
## 0 0   
## blueHeralds blueTowersDestroyed   
## 0 0   
## blueTotalGold blueAvgLevel   
## 0 0   
## blueTotalExperience blueTotalMinionsKilled   
## 0 0   
## blueTotalJungleMinionsKilled blueGoldDiff   
## 0 0   
## blueExperienceDiff blueCSPerMin   
## 0 0   
## blueGoldPerMin redWardsPlaced   
## 0 0   
## redWardsDestroyed redFirstBlood   
## 0 0   
## redKills redDeaths   
## 0 0   
## redAssists redEliteMonsters   
## 0 0   
## redDragons redHeralds   
## 0 0   
## redTowersDestroyed redTotalGold   
## 0 0   
## redAvgLevel redTotalExperience   
## 0 0   
## redTotalMinionsKilled redTotalJungleMinionsKilled   
## 0 0   
## redGoldDiff redExperienceDiff   
## 0 0   
## redCSPerMin redGoldPerMin   
## 0 0

str(data)

## function (..., list = character(), package = NULL,   
## lib.loc = NULL, verbose = getOption("verbose"),   
## envir = .GlobalEnv, overwrite = TRUE)

Miramos propiedades generales de las variables con *describe()* del paquete [psych](https://www.rdocumentation.org/packages/psych/versions/2.3.3/topics/describe).

describe(df)

## vars n mean  
## gameId 1 9879 4500084044.85  
## blueWins 2 9879 0.50  
## blueWardsPlaced 3 9879 22.29  
## blueWardsDestroyed 4 9879 2.82  
## blueFirstBlood 5 9879 0.50  
## blueKills 6 9879 6.18  
## blueDeaths 7 9879 6.14  
## blueAssists 8 9879 6.65  
## blueEliteMonsters 9 9879 0.55  
## blueDragons 10 9879 0.36  
## blueHeralds 11 9879 0.19  
## blueTowersDestroyed 12 9879 0.05  
## blueTotalGold 13 9879 16503.46  
## blueAvgLevel 14 9879 6.92  
## blueTotalExperience 15 9879 17928.11  
## blueTotalMinionsKilled 16 9879 216.70  
## blueTotalJungleMinionsKilled 17 9879 50.51  
## blueGoldDiff 18 9879 14.41  
## blueExperienceDiff 19 9879 -33.62  
## blueCSPerMin 20 9879 21.67  
## blueGoldPerMin 21 9879 1650.35  
## redWardsPlaced 22 9879 22.37  
## redWardsDestroyed 23 9879 2.72  
## redFirstBlood 24 9879 0.50  
## redKills 25 9879 6.14  
## redDeaths 26 9879 6.18  
## redAssists 27 9879 6.66  
## redEliteMonsters 28 9879 0.57  
## redDragons 29 9879 0.41  
## redHeralds 30 9879 0.16  
## redTowersDestroyed 31 9879 0.04  
## redTotalGold 32 9879 16489.04  
## redAvgLevel 33 9879 6.93  
## redTotalExperience 34 9879 17961.73  
## redTotalMinionsKilled 35 9879 217.35  
## redTotalJungleMinionsKilled 36 9879 51.31  
## redGoldDiff 37 9879 -14.41  
## redExperienceDiff 38 9879 33.62  
## redCSPerMin 39 9879 21.73  
## redGoldPerMin 40 9879 1648.90  
## sd median  
## gameId 27573278.49 4510920346.0  
## blueWins 0.50 0.0  
## blueWardsPlaced 18.02 16.0  
## blueWardsDestroyed 2.17 3.0  
## blueFirstBlood 0.50 1.0  
## blueKills 3.01 6.0  
## blueDeaths 2.93 6.0  
## blueAssists 4.06 6.0  
## blueEliteMonsters 0.63 0.0  
## blueDragons 0.48 0.0  
## blueHeralds 0.39 0.0  
## blueTowersDestroyed 0.24 0.0  
## blueTotalGold 1535.45 16398.0  
## blueAvgLevel 0.31 7.0  
## blueTotalExperience 1200.52 17951.0  
## blueTotalMinionsKilled 21.86 218.0  
## blueTotalJungleMinionsKilled 9.90 50.0  
## blueGoldDiff 2453.35 14.0  
## blueExperienceDiff 1920.37 -28.0  
## blueCSPerMin 2.19 21.8  
## blueGoldPerMin 153.54 1639.8  
## redWardsPlaced 18.46 16.0  
## redWardsDestroyed 2.14 2.0  
## redFirstBlood 0.50 0.0  
## redKills 2.93 6.0  
## redDeaths 3.01 6.0  
## redAssists 4.06 6.0  
## redEliteMonsters 0.63 0.0  
## redDragons 0.49 0.0  
## redHeralds 0.37 0.0  
## redTowersDestroyed 0.22 0.0  
## redTotalGold 1490.89 16378.0  
## redAvgLevel 0.31 7.0  
## redTotalExperience 1198.58 17974.0  
## redTotalMinionsKilled 21.91 218.0  
## redTotalJungleMinionsKilled 10.03 51.0  
## redGoldDiff 2453.35 -14.0  
## redExperienceDiff 1920.37 28.0  
## redCSPerMin 2.19 21.8  
## redGoldPerMin 149.09 1637.8  
## trimmed mad  
## gameId 4504103897.37 19856848.76  
## blueWins 0.50 0.00  
## blueWardsPlaced 18.29 2.97  
## blueWardsDestroyed 2.61 1.48  
## blueFirstBlood 0.51 0.00  
## blueKills 6.04 2.97  
## blueDeaths 6.00 2.97  
## blueAssists 6.31 4.45  
## blueEliteMonsters 0.47 0.00  
## blueDragons 0.33 0.00  
## blueHeralds 0.11 0.00  
## blueTowersDestroyed 0.00 0.00  
## blueTotalGold 16439.35 1506.32  
## blueAvgLevel 6.92 0.30  
## blueTotalExperience 17946.56 1154.95  
## blueTotalMinionsKilled 217.21 22.24  
## blueTotalJungleMinionsKilled 50.36 8.90  
## blueGoldDiff 11.84 2360.30  
## blueExperienceDiff -35.52 1856.22  
## blueCSPerMin 21.72 2.22  
## blueGoldPerMin 1643.93 150.63  
## redWardsPlaced 18.37 2.97  
## redWardsDestroyed 2.51 1.48  
## redFirstBlood 0.49 0.00  
## redKills 6.00 2.97  
## redDeaths 6.04 2.97  
## redAssists 6.33 4.45  
## redEliteMonsters 0.50 0.00  
## redDragons 0.39 0.00  
## redHeralds 0.08 0.00  
## redTowersDestroyed 0.00 0.00  
## redTotalGold 16428.95 1466.29  
## redAvgLevel 6.93 0.30  
## redTotalExperience 17982.25 1150.50  
## redTotalMinionsKilled 217.89 22.24  
## redTotalJungleMinionsKilled 51.06 10.38  
## redGoldDiff -11.84 2360.30  
## redExperienceDiff 35.52 1856.22  
## redCSPerMin 21.79 2.22  
## redGoldPerMin 1642.90 146.63  
## min max  
## gameId 4295358071.0 4527990640.0  
## blueWins 0.0 1.0  
## blueWardsPlaced 5.0 250.0  
## blueWardsDestroyed 0.0 27.0  
## blueFirstBlood 0.0 1.0  
## blueKills 0.0 22.0  
## blueDeaths 0.0 22.0  
## blueAssists 0.0 29.0  
## blueEliteMonsters 0.0 2.0  
## blueDragons 0.0 1.0  
## blueHeralds 0.0 1.0  
## blueTowersDestroyed 0.0 4.0  
## blueTotalGold 10730.0 23701.0  
## blueAvgLevel 4.6 8.0  
## blueTotalExperience 10098.0 22224.0  
## blueTotalMinionsKilled 90.0 283.0  
## blueTotalJungleMinionsKilled 0.0 92.0  
## blueGoldDiff -10830.0 11467.0  
## blueExperienceDiff -9333.0 8348.0  
## blueCSPerMin 9.0 28.3  
## blueGoldPerMin 1073.0 2370.1  
## redWardsPlaced 6.0 276.0  
## redWardsDestroyed 0.0 24.0  
## redFirstBlood 0.0 1.0  
## redKills 0.0 22.0  
## redDeaths 0.0 22.0  
## redAssists 0.0 28.0  
## redEliteMonsters 0.0 2.0  
## redDragons 0.0 1.0  
## redHeralds 0.0 1.0  
## redTowersDestroyed 0.0 2.0  
## redTotalGold 11212.0 22732.0  
## redAvgLevel 4.8 8.2  
## redTotalExperience 10465.0 22269.0  
## redTotalMinionsKilled 107.0 289.0  
## redTotalJungleMinionsKilled 4.0 92.0  
## redGoldDiff -11467.0 10830.0  
## redExperienceDiff -8348.0 9333.0  
## redCSPerMin 10.7 28.9  
## redGoldPerMin 1121.2 2273.2  
## range skew kurtosis  
## gameId 232632569.0 -1.46 3.33  
## blueWins 1.0 0.00 -2.00  
## blueWardsPlaced 245.0 4.14 23.42  
## blueWardsDestroyed 27.0 2.85 17.18  
## blueFirstBlood 1.0 -0.02 -2.00  
## blueKills 22.0 0.54 0.26  
## blueDeaths 22.0 0.51 0.21  
## blueAssists 29.0 0.89 1.16  
## blueEliteMonsters 2.0 0.69 -0.50  
## blueDragons 1.0 0.57 -1.67  
## blueHeralds 1.0 1.60 0.55  
## blueTowersDestroyed 4.0 5.59 39.83  
## blueTotalGold 12971.0 0.47 0.48  
## blueAvgLevel 3.4 -0.34 1.11  
## blueTotalExperience 12126.0 -0.25 0.68  
## blueTotalMinionsKilled 193.0 -0.27 0.17  
## blueTotalJungleMinionsKilled 92.0 0.12 0.38  
## blueGoldDiff 22297.0 0.03 0.30  
## blueExperienceDiff 17681.0 0.02 0.36  
## blueCSPerMin 19.3 -0.27 0.17  
## blueGoldPerMin 1297.1 0.47 0.48  
## redWardsPlaced 270.0 4.56 30.45  
## redWardsDestroyed 24.0 2.95 18.22  
## redFirstBlood 1.0 0.02 -2.00  
## redKills 22.0 0.51 0.21  
## redDeaths 22.0 0.54 0.26  
## redAssists 28.0 0.82 0.78  
## redEliteMonsters 2.0 0.62 -0.57  
## redDragons 1.0 0.35 -1.88  
## redHeralds 1.0 1.85 1.44  
## redTowersDestroyed 2.0 5.34 30.53  
## redTotalGold 11520.0 0.41 0.22  
## redAvgLevel 3.4 -0.40 1.23  
## redTotalExperience 11804.0 -0.28 0.82  
## redTotalMinionsKilled 182.0 -0.29 0.23  
## redTotalJungleMinionsKilled 88.0 0.23 0.41  
## redGoldDiff 22297.0 -0.03 0.30  
## redExperienceDiff 17681.0 -0.02 0.36  
## redCSPerMin 18.2 -0.29 0.23  
## redGoldPerMin 1152.0 0.41 0.22  
## se  
## gameId 277416.26  
## blueWins 0.01  
## blueWardsPlaced 0.18  
## blueWardsDestroyed 0.02  
## blueFirstBlood 0.01  
## blueKills 0.03  
## blueDeaths 0.03  
## blueAssists 0.04  
## blueEliteMonsters 0.01  
## blueDragons 0.00  
## blueHeralds 0.00  
## blueTowersDestroyed 0.00  
## blueTotalGold 15.45  
## blueAvgLevel 0.00  
## blueTotalExperience 12.08  
## blueTotalMinionsKilled 0.22  
## blueTotalJungleMinionsKilled 0.10  
## blueGoldDiff 24.68  
## blueExperienceDiff 19.32  
## blueCSPerMin 0.02  
## blueGoldPerMin 1.54  
## redWardsPlaced 0.19  
## redWardsDestroyed 0.02  
## redFirstBlood 0.01  
## redKills 0.03  
## redDeaths 0.03  
## redAssists 0.04  
## redEliteMonsters 0.01  
## redDragons 0.00  
## redHeralds 0.00  
## redTowersDestroyed 0.00  
## redTotalGold 15.00  
## redAvgLevel 0.00  
## redTotalExperience 12.06  
## redTotalMinionsKilled 0.22  
## redTotalJungleMinionsKilled 0.10  
## redGoldDiff 24.68  
## redExperienceDiff 19.32  
## redCSPerMin 0.02  
## redGoldPerMin 1.50

## Filas lógicamente correlacionadas en abse a conocimientos del juego

La columna *gameId* no nos aporta información puesto que no guarda informaciíon mas allá que para cuantificar las partidas en sus servidores, la podemos quitar:

df <- df[,-1]

Por cómo funciona el juego, en este dataset hay muchas columnas con información redundante, puesto que si un equipo se hace una baja (*blueKill*), se corresponde con que el equipo rojo obtiene una muerte (*redDeaths*), por lo que esas columnas siempre van a contener información complementaria.

df <- df[,!names(df) %in% c("blueDeaths", "redDeaths")]

De forma análoga, como sabemos que el dataset está capado en los primeros 10 minutos de la partida, también podemos limpiar columnas *PerMin* puesto que van a estar directamente correladas con columnas de su misma métrica. También las columnas que muestran un diferencial de oro o experiencia se pueden suprimir en valor de los valores totales (*blueTotalGold*, y *redTotalGold*).

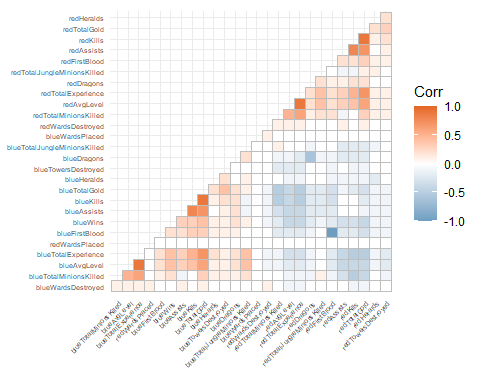
df <- df[,!names(df) %in% c("blueCSPerMin", "redCSPerMin",  
 "blueGoldPerMin", "redGoldPerMin",  
 "blueGoldDiff", "redGoldDiff",  
 "blueExperienceDiff","redExperienceDiff"  
 )]

Por último, este dataset agrupaba los [Monstruos Épicos](https://leagueoflegends.fandom.com/wiki/Category:Epic_monsters) en una misma columna, he decidido quedarme con los datos por separado (*blueDragons, blueHeralds, redDragons, redHeralds*) para estudiar si preocuparse más de un objetivo u otro es determinanteb a la hora de ganar o perder.

df <- df[,!names(df) %in% c( "blueEliteMonsters", "redEliteMonsters")]

Buscamos correlaciones entre las variables que nos quedan:

corr <- round(cor(df), 1)  
ggcorrplot( corr,  
 hc.order = TRUE,  
 type = "lower",  
 outline.col = "gray",  
 tl.cex = 5,  
 colors = c("#6D9EC1", "white", "#E46726")  
 )



Podemos observar que *blueAVGlevel* tiene una correlación muy alta con *blueTotalExperience*, y sus contrapartidas del equipo rojo *redAVGLevel* y *redTotalExperience*, por lo que elegimos una de las 2 para representar este dato, en esta caso vamos a elegir los totales.

df <- df[,!names(df) %in% c( "blueAVGLevel", "redAVGLevel")]

La columna *redFirstBlood* es la contraparte de *blueFirstBlood*, quitamos la variable del equipo rojo porque así coincide con *blueWins*, que es nuestra variable a predecir (no tenemos *redWins*, porque precisamente sería la contraparte).

df <- df[,!names(df) %in% c("redFirstBlood")]

Analizamos la distribución, separando entre equipo rojo y equipo azul por claridad a la hora de mostrar los gráficos

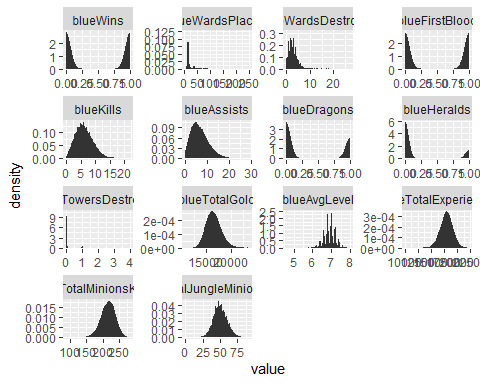
blue\_df <- melt(df[1:14])

## No id variables; using all as measure variables

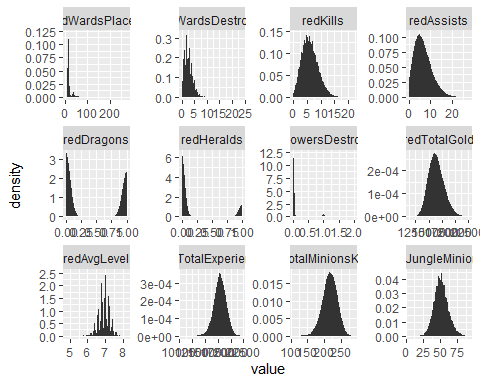
red\_df <- melt(df[15:26])

## No id variables; using all as measure variables

ggplot(data=blue\_df, aes(x=value))+  
stat\_density()+  
facet\_wrap(~variable, scales="free")



ggplot(data=red\_df, aes(x=value))+  
stat\_density()+  
facet\_wrap(~variable, scales="free")

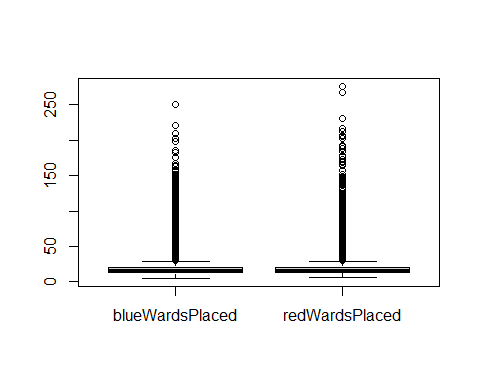


Podemos ver que las variables tienen una distribución normal cuando se trata de variables no dicotómicas, mientras las dicotómicas (*blueWins*, *blueFirstBlood*, *blueHerlads*, *blueDragons*) cumplen con su definición.

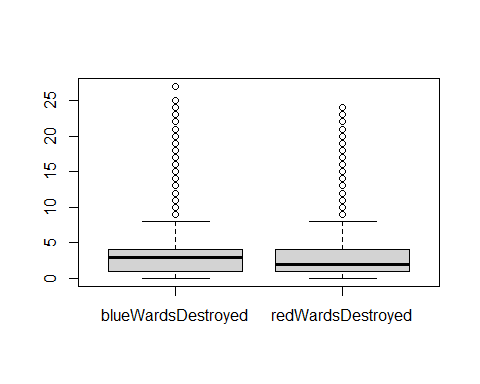
Los Dragones y Heraldos son dicotómicas por las características del *dataset*, puesto que no pueden más de 1 vez antes de la marca temporal de los 10min.

Vamos a hacer boxplot de los casos anómalos para encontrar *outliers* y decidir si los debemos quitar.

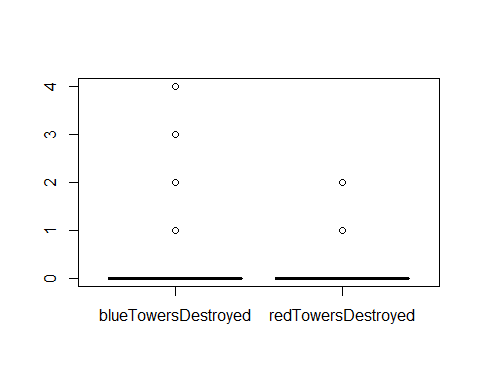
boxplot(df[c("blueWardsPlaced","redWardsPlaced")])



boxplot(df[c("blueWardsDestroyed","redWardsDestroyed")])



boxplot(df[c("blueTowersDestroyed","redTowersDestroyed")])



Se comprueba que son datos plausibles dentro de cómo funciona *League of Legends*, se podría valorar quitarlos si posteriormente perjudicasen al modelo. El caso más preocupante serían las partidas donde se tiran más de 2 torres al minuto 10 pero de momento no se van a sacar del modelo.

## Análisis de componentes principales PCA

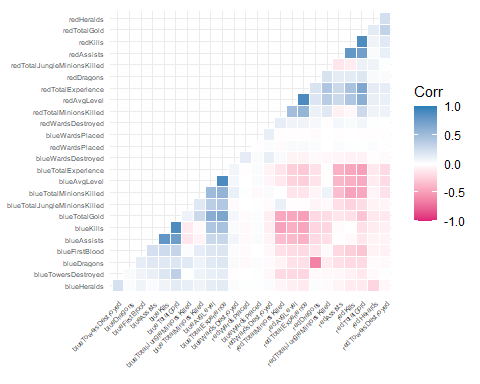
Quitamos la variable a predecir (*blueWins*)

df\_nowins <- df[,-1]  
df\_wins <-df$blueWins

df\_normalized <- scale(df\_nowins)  
head(df\_normalized)

## blueWardsPlaced blueWardsDestroyed blueFirstBlood  
## [1,] 0.3169796 -0.3792559 0.9903793  
## [2,] -0.5709633 -0.8390264 -1.0096119  
## [3,] -0.4044740 -1.2987969 -1.0096119  
## [4,] 1.1494261 -0.8390264 -1.0096119  
## [5,] 2.9253119 0.5402850 -1.0096119  
## [6,] -0.2379847 -1.2987969 -1.0096119  
## blueKills blueAssists blueDragons blueHeralds  
## [1,] 0.93525352 1.0714412 -0.7531875 -0.4811081  
## [2,] -0.39319645 -0.4047479 -0.7531875 -0.4811081  
## [3,] 0.27102854 -0.6507794 1.3275562 -0.4811081  
## [4,] -0.72530894 -0.4047479 -0.7531875 2.0783247  
## [5,] -0.06108396 -0.1587164 -0.7531875 -0.4811081  
## [6,] -0.39319645 -0.1587164 1.3275562 -0.4811081  
## blueTowersDestroyed blueTotalGold blueAvgLevel  
## [1,] -0.2104284 0.46015568 -1.0355824  
## [2,] -0.2104284 -1.16673251 -1.0355824  
## [3,] -0.2104284 -0.25429442 -1.6910067  
## [4,] -0.2104284 -0.87691456 0.2752663  
## [5,] -0.2104284 -0.06737812 0.2752663  
## [6,] -0.2104284 -0.39366755 0.2752663  
## blueTotalExperience blueTotalMinionsKilled  
## [1,] -0.74060186 -0.9927317  
## [2,] -1.38532046 -1.9534592  
## [3,] -1.42197113 -1.4044721  
## [4,] 0.02156548 -0.7182382  
## [5,] 0.51218467 -0.3064979  
## [6,] 0.19399022 0.3797360  
## blueTotalJungleMinionsKilled redWardsPlaced  
## [1,] -1.4658773 -0.3991863  
## [2,] -0.7586839 -0.5617225  
## [3,] -0.4556010 -0.3991863  
## [4,] 0.4536477 -0.3991863  
## [5,] 0.6557030 -0.2908289  
## [6,] -0.8597115 0.7385671  
## redWardsDestroyed redKills redAssists redDragons  
## [1,] 1.5324154 -0.04692376 0.3294795 -0.8389232  
## [2,] -0.8058293 -0.38777657 -1.1481301 1.1918836  
## [3,] 0.1294686 1.65734027 1.8070891 -0.8389232  
## [4,] -0.3381804 -0.38777657 0.8220160 -0.8389232  
## [5,] -0.3381804 -0.04692376 0.0832112 1.1918836  
## [6,] 1.0647665 -1.06948218 -1.1481301 -0.8389232  
## redHeralds redTowersDestroyed redTotalGold redAvgLevel  
## [1,] -0.4364729 -0.1983428 0.05229003 -0.4104541  
## [2,] 2.2908613 4.4120780 0.75858032 -0.4104541  
## [3,] -0.4364729 -0.1983428 0.53388208 -0.4104541  
## [4,] -0.4364729 -0.1983428 -0.00740592 0.2446147  
## [5,] -0.4364729 -0.1983428 0.61370026 0.2446147  
## [6,] -0.4364729 -0.1983428 -0.86394219 0.2446147  
## redTotalExperience redTotalMinionsKilled  
## [1,] -0.7631759689 -0.9286936  
## [2,] -0.4369576740 1.0337312  
## [3,] -0.5904721657 -0.6548669  
## [4,] -0.0006094177 0.8055423  
## [5,] 0.2930704794 0.3491644  
## [6,] 0.0819880533 0.1666133  
## redTotalJungleMinionsKilled  
## [1,] 0.36766593  
## [2,] 0.06850015  
## [3,] -2.32482609  
## [4,] -0.43010948  
## [5,] 1.56432905  
## [6,] 0.76655364

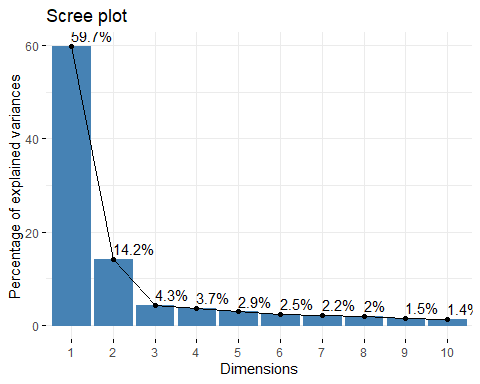
corr\_norm <- cor(df\_normalized)  
ggcorrplot( corr\_norm,  
 hc.order = TRUE,  
 type = "lower",  
 outline.col = "white",  
 tl.cex = 5,  
 colors = c("#dd1c77", "white", "#2c7fb8")  
 )



df\_PCA <- prcomp(corr\_norm)  
summary(df\_PCA)

## Importance of components:  
## PC1 PC2 PC3 PC4  
## Standard deviation 1.1977 0.5837 0.31998 0.29919  
## Proportion of Variance 0.5974 0.1419 0.04264 0.03728  
## Cumulative Proportion 0.5974 0.7393 0.78197 0.81925  
## PC5 PC6 PC7 PC8  
## Standard deviation 0.26446 0.24332 0.22751 0.21725  
## Proportion of Variance 0.02913 0.02466 0.02156 0.01966  
## Cumulative Proportion 0.84838 0.87304 0.89460 0.91426  
## PC9 PC10 PC11 PC12  
## Standard deviation 0.19026 0.18008 0.17039 0.15808  
## Proportion of Variance 0.01508 0.01351 0.01209 0.01041  
## Cumulative Proportion 0.92933 0.94284 0.95493 0.96534  
## PC13 PC14 PC15 PC16  
## Standard deviation 0.14433 0.14011 0.13282 0.11525  
## Proportion of Variance 0.00868 0.00818 0.00735 0.00553  
## Cumulative Proportion 0.97402 0.98219 0.98954 0.99507  
## PC17 PC18 PC19 PC20  
## Standard deviation 0.07044 0.05892 0.03885 0.02909  
## Proportion of Variance 0.00207 0.00145 0.00063 0.00035  
## Cumulative Proportion 0.99714 0.99859 0.99921 0.99957  
## PC21 PC22 PC23 PC24  
## Standard deviation 0.02416 0.01509 0.01336 0.007177  
## Proportion of Variance 0.00024 0.00009 0.00007 0.000020  
## Cumulative Proportion 0.99981 0.99990 0.99998 1.000000  
## PC25  
## Standard deviation 1.212e-18  
## Proportion of Variance 0.000e+00  
## Cumulative Proportion 1.000e+00

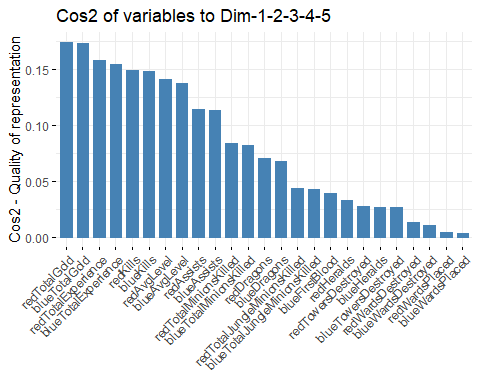
fviz\_eig(df\_PCA, addlabels = TRUE)



Con las 2 primeras variables se podría explicar casi todo el dataset, pero que elegimos las primeras 5 para tener un alto porcentaje de explicación de la variabilidad.

Con el gráfico a continuación mostramos lo que aportan cada variable a las 5 dimensiones que hemos elegido

fviz\_cos2(df\_PCA, choice = "var", axes = 1:5)



## Separación de los datos:

Se escoge repartir los datos de forma aleatoria en 2 subsets, uno de entrenamiento y otro para comprobar posteriormente.

mark1 <- sample(c ( TRUE , FALSE ), nrow (df), replace = TRUE , prob = c (0.8, 0.2))  
train <- df[mark1, ]  
test <- df[!mark1, ]  
  
mark2 <- sample(c ( TRUE , FALSE ), nrow (df\_nowins), replace = TRUE , prob = c (0.8, 0.2))  
train\_nowins <- df\_nowins[mark2, ]  
test\_nowins <- df\_nowins[!mark2, ]

# Regresión lineal múltiple (con el dataframe original)

model\_glm <- glm( blueWins ~ ., data = train)  
  
summary(model\_glm)

##   
## Call:  
## glm(formula = blueWins ~ ., data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.08810 -0.36100 -0.00721 0.36152 1.16895   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 3.443e-01 2.421e-01 1.422  
## blueWardsPlaced -1.494e-04 2.678e-04 -0.558  
## blueWardsDestroyed 2.494e-03 2.224e-03 1.121  
## blueFirstBlood 1.855e-02 1.081e-02 1.716  
## blueKills -6.826e-03 5.969e-03 -1.144  
## blueAssists -2.443e-03 2.281e-03 -1.071  
## blueDragons 6.963e-02 1.325e-02 5.254  
## blueHeralds 6.238e-03 1.314e-02 0.475  
## blueTowersDestroyed -7.310e-02 2.539e-02 -2.879  
## blueTotalGold 9.481e-05 1.363e-05 6.958  
## blueAvgLevel 7.105e-03 3.660e-02 0.194  
## blueTotalExperience 3.817e-05 1.198e-05 3.187  
## blueTotalMinionsKilled -7.799e-04 3.844e-04 -2.029  
## blueTotalJungleMinionsKilled 6.825e-04 6.620e-04 1.031  
## redWardsPlaced -1.297e-04 2.593e-04 -0.500  
## redWardsDestroyed -2.096e-03 2.298e-03 -0.912  
## redKills 9.141e-03 6.008e-03 1.521  
## redAssists 9.262e-04 2.220e-03 0.417  
## redDragons -5.961e-02 1.300e-02 -4.586  
## redHeralds -2.699e-02 1.393e-02 -1.937  
## redTowersDestroyed 1.126e-01 2.737e-02 4.114  
## redTotalGold -9.003e-05 1.357e-05 -6.633  
## redAvgLevel 7.001e-03 3.650e-02 0.192  
## redTotalExperience -5.053e-05 1.207e-05 -4.187  
## redTotalMinionsKilled 1.208e-03 3.832e-04 3.152  
## redTotalJungleMinionsKilled 1.215e-03 6.587e-04 1.845  
## Pr(>|t|)   
## (Intercept) 0.15501   
## blueWardsPlaced 0.57700   
## blueWardsDestroyed 0.26218   
## blueFirstBlood 0.08617 .   
## blueKills 0.25286   
## blueAssists 0.28407   
## blueDragons 1.53e-07 \*\*\*  
## blueHeralds 0.63486   
## blueTowersDestroyed 0.00400 \*\*   
## blueTotalGold 3.72e-12 \*\*\*  
## blueAvgLevel 0.84607   
## blueTotalExperience 0.00144 \*\*   
## blueTotalMinionsKilled 0.04251 \*   
## blueTotalJungleMinionsKilled 0.30256   
## redWardsPlaced 0.61709   
## redWardsDestroyed 0.36174   
## redKills 0.12820   
## redAssists 0.67656   
## redDragons 4.59e-06 \*\*\*  
## redHeralds 0.05275 .   
## redTowersDestroyed 3.93e-05 \*\*\*  
## redTotalGold 3.52e-11 \*\*\*  
## redAvgLevel 0.84791   
## redTotalExperience 2.86e-05 \*\*\*  
## redTotalMinionsKilled 0.00163 \*\*   
## redTotalJungleMinionsKilled 0.06508 .   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.1803773)  
##   
## Null deviance: 1964.9 on 7859 degrees of freedom  
## Residual deviance: 1413.1 on 7834 degrees of freedom  
## AIC: 8871.8  
##   
## Number of Fisher Scoring iterations: 2

Cosas importantes:

El p-valor es estádisticamente significativo Se ven las variables que son significativa (las que tienen asteriscos)

Pseudo R cuadrado

pscl :: pR2(model\_glm)["McFadden"]

## fitting null model for pseudo-r2

## McFadden   
## 0.2271144

Como esté entre 2.0 y 0.4 esta ok

caret::varImp(model\_glm)

## Overall  
## blueWardsPlaced 0.5578013  
## blueWardsDestroyed 1.1213297  
## blueFirstBlood 1.7161867  
## blueKills 1.1435228  
## blueAssists 1.0713012  
## blueDragons 5.2535951  
## blueHeralds 0.4749169  
## blueTowersDestroyed 2.8789203  
## blueTotalGold 6.9582188  
## blueAvgLevel 0.1941427  
## blueTotalExperience 3.1870435  
## blueTotalMinionsKilled 2.0288329  
## blueTotalJungleMinionsKilled 1.0310341  
## redWardsPlaced 0.5000047  
## redWardsDestroyed 0.9121023  
## redKills 1.5214047  
## redAssists 0.4171816  
## redDragons 4.5860943  
## redHeralds 1.9372734  
## redTowersDestroyed 4.1136347  
## redTotalGold 6.6325882  
## redAvgLevel 0.1917970  
## redTotalExperience 4.1865593  
## redTotalMinionsKilled 3.1519495  
## redTotalJungleMinionsKilled 1.8449632

imp <- as.data.frame(caret::varImp(model\_glm))  
imp <- data.frame(VarImp = imp$Overall,  
 names = rownames(imp))  
imp[order(imp$VarImp,decreasing = T),]

## VarImp names  
## 9 6.9582188 blueTotalGold  
## 21 6.6325882 redTotalGold  
## 6 5.2535951 blueDragons  
## 18 4.5860943 redDragons  
## 23 4.1865593 redTotalExperience  
## 20 4.1136347 redTowersDestroyed  
## 11 3.1870435 blueTotalExperience  
## 24 3.1519495 redTotalMinionsKilled  
## 8 2.8789203 blueTowersDestroyed  
## 12 2.0288329 blueTotalMinionsKilled  
## 19 1.9372734 redHeralds  
## 25 1.8449632 redTotalJungleMinionsKilled  
## 3 1.7161867 blueFirstBlood  
## 16 1.5214047 redKills  
## 4 1.1435228 blueKills  
## 2 1.1213297 blueWardsDestroyed  
## 5 1.0713012 blueAssists  
## 13 1.0310341 blueTotalJungleMinionsKilled  
## 15 0.9121023 redWardsDestroyed  
## 1 0.5578013 blueWardsPlaced  
## 14 0.5000047 redWardsPlaced  
## 7 0.4749169 blueHeralds  
## 17 0.4171816 redAssists  
## 10 0.1941427 blueAvgLevel  
## 22 0.1917970 redAvgLevel

car :: vif(model\_glm)

## blueWardsPlaced blueWardsDestroyed   
## 1.026253 1.069683   
## blueFirstBlood blueKills   
## 1.272326 14.129545   
## blueAssists blueDragons   
## 3.755034 1.760789   
## blueHeralds blueTowersDestroyed   
## 1.132392 1.613627   
## blueTotalGold blueAvgLevel   
## 19.126658 5.393140   
## blueTotalExperience blueTotalMinionsKilled   
## 9.002275 3.085994   
## blueTotalJungleMinionsKilled redWardsPlaced   
## 1.856752 1.033224   
## redWardsDestroyed redKills   
## 1.062944 13.522058   
## redAssists redDragons   
## 3.516989 1.787090   
## redHeralds redTowersDestroyed   
## 1.132103 1.574569   
## redTotalGold redAvgLevel   
## 17.886442 5.440496   
## redTotalExperience redTotalMinionsKilled   
## 9.144101 3.077227   
## redTotalJungleMinionsKilled   
## 1.882147

Como regla general, los valores de VIF por encima de 5 indican una multicolinealidad severa. Dado que hay varias que superan el 10, vamos a hacer más trabajo sobre los datos.

# Naive - Bayes

train\_NB <- train  
train\_NB$blueWins <- as.factor(train\_NB$blueWins)  
  
  
test\_NB <- test  
#test\_NB$blueWins <- as.factor(test\_NB$blueWins)  
  
x\_train\_NB <- train\_NB[,-1]  
y\_train\_NB <- train\_NB$blueWins

model\_NB = caret::train(x\_train\_NB,y\_train\_NB,'naive\_bayes',trControl=trainControl(method='cv',number=10))  
# cv es cross validation  
   
model\_NB

## Naive Bayes   
##   
## 7860 samples  
## 25 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 7074, 7074, 7074, 7074, 7074, 7074, ...   
## Resampling results across tuning parameters:  
##   
## usekernel Accuracy Kappa   
## FALSE 0.7237913 0.4474615  
## TRUE 0.7253181 0.4507835  
##   
## Tuning parameter 'laplace' was held constant at a value  
## of 0  
## Tuning parameter 'adjust' was held constant at  
## a value of 1  
## Accuracy was used to select the optimal model using  
## the largest value.  
## The final values used for the model were laplace =  
## 0, usekernel = TRUE and adjust = 1.

predictions\_NB <- predict(model\_NB, newdata = test\_NB )  
real\_NB <- as.factor(test\_NB$blueWins)  
caret::confusionMatrix(predictions\_NB, real\_NB)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 709 271  
## 1 279 760  
##   
## Accuracy : 0.7276   
## 95% CI : (0.7076, 0.7469)  
## No Information Rate : 0.5106   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.4548   
##   
## Mcnemar's Test P-Value : 0.7653   
##   
## Sensitivity : 0.7176   
## Specificity : 0.7371   
## Pos Pred Value : 0.7235   
## Neg Pred Value : 0.7315   
## Prevalence : 0.4894   
## Detection Rate : 0.3512   
## Detection Prevalence : 0.4854   
## Balanced Accuracy : 0.7274   
##   
## 'Positive' Class : 0   
##

Esto dice que un 73 casi de aciertos así que pinta bien,

caret::filterVarImp(model\_NB$trainingData,y\_train\_NB,nonpara = TRUE)

## X0 X1  
## blueWardsPlaced 0.5280854 0.5280854  
## blueWardsDestroyed 0.5379639 0.5379639  
## blueFirstBlood 0.6008027 0.6008027  
## blueKills 0.6910926 0.6910926  
## blueAssists 0.6597608 0.6597608  
## blueDragons 0.6058901 0.6058901  
## blueHeralds 0.5353016 0.5353016  
## blueTowersDestroyed 0.5239858 0.5239858  
## blueTotalGold 0.7412715 0.7412715  
## blueAvgLevel 0.7018509 0.7018509  
## blueTotalExperience 0.7285169 0.7285169  
## blueTotalMinionsKilled 0.6265670 0.6265670  
## blueTotalJungleMinionsKilled 0.5759234 0.5759234  
## redWardsPlaced 0.5261101 0.5261101  
## redWardsDestroyed 0.5431864 0.5431864  
## redKills 0.6926278 0.6926278  
## redAssists 0.6569627 0.6569627  
## redDragons 0.6064042 0.6064042  
## redHeralds 0.5360241 0.5360241  
## redTowersDestroyed 0.5192735 0.5192735  
## redTotalGold 0.7367584 0.7367584  
## redAvgLevel 0.7000287 0.7000287  
## redTotalExperience 0.7234052 0.7234052  
## redTotalMinionsKilled 0.6204614 0.6204614  
## redTotalJungleMinionsKilled 0.5612379 0.5612379  
## .outcome 1.0000000 1.0000000

imp\_1 <- as.data.frame(caret::filterVarImp(model\_NB$trainingData,y\_train\_NB,nonpara = TRUE))  
imp\_1 <- data.frame(VarImp = imp\_1$X0,  
 names = rownames(imp\_1))  
imp\_1<-head(imp\_1, -1)  
imp\_1[order(imp\_1$VarImp, decreasing = T),]

## VarImp names  
## 9 0.7412715 blueTotalGold  
## 21 0.7367584 redTotalGold  
## 11 0.7285169 blueTotalExperience  
## 23 0.7234052 redTotalExperience  
## 10 0.7018509 blueAvgLevel  
## 22 0.7000287 redAvgLevel  
## 16 0.6926278 redKills  
## 4 0.6910926 blueKills  
## 5 0.6597608 blueAssists  
## 17 0.6569627 redAssists  
## 12 0.6265670 blueTotalMinionsKilled  
## 24 0.6204614 redTotalMinionsKilled  
## 18 0.6064042 redDragons  
## 6 0.6058901 blueDragons  
## 3 0.6008027 blueFirstBlood  
## 13 0.5759234 blueTotalJungleMinionsKilled  
## 25 0.5612379 redTotalJungleMinionsKilled  
## 15 0.5431864 redWardsDestroyed  
## 2 0.5379639 blueWardsDestroyed  
## 19 0.5360241 redHeralds  
## 7 0.5353016 blueHeralds  
## 1 0.5280854 blueWardsPlaced  
## 14 0.5261101 redWardsPlaced  
## 8 0.5239858 blueTowersDestroyed  
## 20 0.5192735 redTowersDestroyed