Haciendo Big Data a League of Legends

Jorge de Andrés

16 de enero de 2019

Importamos los paquetes necesarios:

```
#install.packages("dplyr")
#install.packages("pryr")
#install.packages("corrplot")
#install.packages("rjson")
#install.packages("plyr")
#install.packages("wordcloud")
#install.packages("ggplot2")
#install.packages("hexbin")
#install.packages("RColorBrewer")
#install.packages("FactoMineR")
#devtools::install_github("kassambara/factoextra")
#install.packages("factoextra")

library("corrplot")
```

```
## corrplot 0.84 loaded
```

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("pryr")
```

```
## Warning: package 'pryr' was built under R version 3.5.2
```

```
library("rjson")
```

```
## Warning: package 'rjson' was built under R version 3.5.2
```

```
library("plyr")
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library("wordcloud")
```

```
## Warning: package 'wordcloud' was built under R version 3.5.2
```

```
## Loading required package: RColorBrewer
```

```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
library("hexbin")
library("RColorBrewer")
library("FactoMineR")
```

```
## Warning: package 'FactoMineR' was built under R version 3.5.2
```

```
library("factoextra")
```

```
## Warning: package 'factoextra' was built under R version 3.5.2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

Lo primero que voy a hacer es importar todos los datasets:

```
data.games <- read.csv("../data/games.csv")
data.champs <- read.csv("../data/champs.csv")
```

Hemos importado varios CSV que harán las tablas de la base de datos.

Importamos los json:

```
lista.championInfo <- fromJSON(file = "../data/champion_info.json")
lista.championInfo2 <- fromJSON(file="../data/champion_info_2.json")
lista.summonerSpell <- fromJSON(file="../data/summoner_spell_info.json")
```

Tamaño de mis datasets...

```
print("El tamaño de la tabla de games es de: ")
```

```
## [1] "El tamaño de la tabla de games es de: "
```

```
object_size(data.games) # Es un dataset de 433 MB
```

```
## 13 MB
```

Esto me devuelve las filas que tengan NA:

```
data.games[!complete.cases(data.games),]
```

```
##  [1] gameId             creationTime       gameDuration
##  [4] seasonId           winner             firstBlood
##  [7] firstTower         firstInhibitor     firstBaron
## [10] firstDragon        firstRiftHerald    t1_champ1id
## [13] t1_champ1_sum1     t1_champ1_sum2     t1_champ2id
## [16] t1_champ2_sum1     t1_champ2_sum2     t1_champ3id
## [19] t1_champ3_sum1     t1_champ3_sum2     t1_champ4id
## [22] t1_champ4_sum1     t1_champ4_sum2     t1_champ5id
## [25] t1_champ5_sum1     t1_champ5_sum2     t1_towerKills
```

```
## [28] t1_inhibitorKills  t1_baronKills      t1_dragonKills
## [31] t1_riftHeraldKills t1_ban1            t1_ban2
## [34] t1_ban3            t1_ban4            t1_ban5
## [37] t2_champ1id        t2_champ1_sum1     t2_champ1_sum2
## [40] t2_champ2id        t2_champ2_sum1     t2_champ2_sum2
## [43] t2_champ3id        t2_champ3_sum1     t2_champ3_sum2
## [46] t2_champ4id        t2_champ4_sum1     t2_champ4_sum2
## [49] t2_champ5id        t2_champ5_sum1     t2_champ5_sum2
## [52] t2_towerKills      t2_inhibitorKills  t2_baronKills
## [55] t2_dragonKills     t2_riftHeraldKills t2_ban1
## [58] t2_ban2            t2_ban3            t2_ban4
## [61] t2_ban5
## <0 rows> (or 0-length row.names)
```

Como podemos ver, no tenemos filas con NA.

Como voy a ir haciendo "pequeños proyectos" para ir sacando conocimiento por ahora no voy a fusionar ninguna tabla de primeras porque no se lo que voy a necesitar. Según se vayan planteando los interrogantes iré jugando con las tablas para conseguir los objetivos

Análisis Exploratorio

Empezamos por el análisis exploratorio de los datos. Para ello, vamos intentar ver conclusiones sobre los mismos:

```
print("Resumen de data.games")
```

```
## [1] "Resumen de data.games"
```

```
summary(data.games) # Las únicas columnas interesantes son la 5 y la 6
```

```
##      gameId           creationTime        gameDuration     seasonId
##  Min.   :3.215e+09   Min.   :1.497e+12   Min.   : 190   Min.   :9
##  1st Qu.:3.292e+09   1st Qu.:1.502e+12   1st Qu.:1531   1st Qu.:9
##  Median :3.320e+09   Median :1.504e+12   Median :1833   Median :9
##  Mean   :3.306e+09   Mean   :1.503e+12   Mean   :1832   Mean   :9
##  3rd Qu.:3.327e+09   3rd Qu.:1.504e+12   3rd Qu.:2148   3rd Qu.:9
##  Max.   :3.332e+09   Max.   :1.505e+12   Max.   :4728   Max.   :9
##      winner        firstBlood      firstTower    firstInhibitor
##  Min.   :1.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000
##  Median :1.000   Median :1.000   Median :1.000   Median :1.000
##  Mean   :1.494   Mean   :1.471   Mean   :1.451   Mean   :1.308
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :2.000
##    firstBaron      firstDragon    firstRiftHerald   t1_champ1id
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :  1.0
##  1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.: 35.0
##  Median :1.0000   Median :1.000   Median :0.0000   Median : 79.0
##  Mean   :0.9265   Mean   :1.443   Mean   :0.7317   Mean   :114.3
##  3rd Qu.:2.0000   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:136.0
##  Max.   :2.0000   Max.   :2.000   Max.   :2.0000   Max.   :516.0
##  t1_champ1_sum1   t1_champ1_sum2    t1_champ2id    t1_champ2_sum1
##  Min.   : 1.000   Min.   : 1.000   Min.   :  1.0   Min.   : 1.000
##  1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 35.0   1st Qu.: 4.000
##  Median : 4.000   Median : 4.000   Median : 79.0   Median : 4.000
```

```
##  Mean   : 6.602   Mean   : 7.334   Mean   :118.1   Mean   : 6.548
##  3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.:141.0   3rd Qu.:11.000
##  Max.   :21.000   Max.   :21.000   Max.   :516.0   Max.   :21.000
##  t1_champ2_sum2    t1_champ3id    t1_champ3_sum1   t1_champ3_sum2
##  Min.   : 1.000   Min.   :  1.0   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 4.000   1st Qu.: 35.0   1st Qu.: 4.000   1st Qu.: 4.000
##  Median : 4.000   Median : 78.0   Median : 4.000   Median : 4.000
##  Mean   : 7.198   Mean   :116.9   Mean   : 6.542   Mean   : 7.201
##  3rd Qu.:11.000   3rd Qu.:141.0   3rd Qu.:11.000   3rd Qu.:11.000
##  Max.   :21.000   Max.   :516.0   Max.   :21.000   Max.   :21.000
##   t1_champ4id    t1_champ4_sum1   t1_champ4_sum2    t1_champ5id
##  Min.   :  1.0   Min.   : 1.000   Min.   : 1.000   Min.   :  1.0
##  1st Qu.: 36.0   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 35.0
##  Median : 79.0   Median : 4.000   Median : 4.000   Median : 78.0
##  Mean   :117.7   Mean   : 6.531   Mean   : 7.221   Mean   :114.6
##  3rd Qu.:141.0   3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.:136.0
##  Max.   :516.0   Max.   :21.000   Max.   :21.000   Max.   :516.0
##  t1_champ5_sum1   t1_champ5_sum2   t1_towerKills    t1_inhibitorKills
##  Min.   : 1.000   Min.   : 1.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.: 0.000
##  Median : 4.000   Median : 4.000   Median : 6.000   Median : 1.000
##  Mean   : 6.622   Mean   : 7.261   Mean   : 5.699   Mean   : 1.018
##  3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.: 9.000   3rd Qu.: 2.000
##  Max.   :21.000   Max.   :21.000   Max.   :11.000   Max.   :10.000
##  t1_baronKills    t1_dragonKills  t1_riftHeraldKills    t1_ban1
##  Min.   :0.0000   Min.   :0.000   Min.   :0.0000     Min.   : -1.0
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000     1st Qu.: 38.0
##  Median :0.0000   Median :1.000   Median :0.0000     Median : 90.0
##  Mean   :0.3723   Mean   :1.387   Mean   :0.2515     Mean   :108.3
##  3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:1.0000     3rd Qu.:141.0
##  Max.   :5.0000   Max.   :6.000   Max.   :1.0000     Max.   :516.0
##     t1_ban2          t1_ban3          t1_ban4         t1_ban5
##  Min.   : -1.0   Min.   : -1.0   Min.   : -1.0   Min.   : -1
##  1st Qu.: 38.0   1st Qu.: 37.0   1st Qu.: 38.0   1st Qu.: 38
##  Median : 90.0   Median : 90.0   Median : 89.0   Median : 90
##  Mean   :108.8   Mean   :108.2   Mean   :107.6   Mean   :109
##  3rd Qu.:141.0   3rd Qu.:141.0   3rd Qu.:141.0   3rd Qu.:141
##  Max.   :516.0   Max.   :516.0   Max.   :516.0   Max.   :516
##   t2_champ1id    t2_champ1_sum1   t2_champ1_sum2    t2_champ2id
##  Min.   :  1.0   Min.   : 1.000   Min.   : 1.000   Min.   :  1.0
##  1st Qu.: 35.0   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 35.0
##  Median : 78.0   Median : 4.000   Median : 4.000   Median : 79.0
##  Mean   :115.9   Mean   : 6.595   Mean   : 7.305   Mean   :117.6
##  3rd Qu.:141.0   3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.:141.0
##  Max.   :516.0   Max.   :21.000   Max.   :21.000   Max.   :516.0
##  t2_champ2_sum1   t2_champ2_sum2    t2_champ3id    t2_champ3_sum1
##  Min.   : 1.000   Min.   : 1.000   Min.   :  1.0   Min.   : 1.000
##  1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 36.0   1st Qu.: 4.000
##  Median : 4.000   Median : 4.000   Median : 79.0   Median : 4.000
##  Mean   : 6.547   Mean   : 7.231   Mean   :117.5   Mean   : 6.522
##  3rd Qu.:11.000   3rd Qu.:11.000   3rd Qu.:141.0   3rd Qu.:11.000
##  Max.   :21.000   Max.   :21.000   Max.   :516.0   Max.   :21.000
##  t2_champ3_sum2    t2_champ4id    t2_champ4_sum1   t2_champ4_sum2
##  Min.   : 1.000   Min.   :  1.0   Min.   : 1.000   Min.   : 1.000
```

```
##  1st Qu.: 4.000   1st Qu.: 35.0   1st Qu.: 4.000   1st Qu.: 4.000
##  Median : 4.000   Median : 79.0   Median : 4.000   Median : 4.000
##  Mean   : 7.227   Mean   :118.2   Mean   : 6.535   Mean   : 7.201
##  3rd Qu.:11.000   3rd Qu.:141.0   3rd Qu.:11.000   3rd Qu.:11.000
##  Max.   :21.000   Max.   :516.0   Max.   :21.000   Max.   :21.000
##   t2_champ5id    t2_champ5_sum1   t2_champ5_sum2  t2_towerKills
##  Min.   :  1.0   Min.   : 1.000   Min.   : 1.00   Min.   : 0.000
##  1st Qu.: 35.0   1st Qu.: 4.000   1st Qu.: 4.00   1st Qu.: 2.000
##  Median : 78.0   Median : 4.000   Median : 4.00   Median : 6.000
##  Mean   :115.9   Mean   : 6.613   Mean   : 7.25   Mean   : 5.549
##  3rd Qu.:141.0   3rd Qu.:11.000   3rd Qu.:11.00   3rd Qu.: 9.000
##  Max.   :516.0   Max.   :21.000   Max.   :21.00   Max.   :11.000
##  t2_inhibitorKills t2_baronKills    t2_dragonKills  t2_riftHeraldKills
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.0000
##  Median : 0.0000   Median :0.0000   Median :1.000   Median :0.0000
##  Mean   : 0.9851   Mean   :0.4145   Mean   :1.404   Mean   :0.2401
##  3rd Qu.: 2.0000   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:0.0000
##  Max.   :10.0000   Max.   :4.0000   Max.   :6.000   Max.   :1.0000
##     t2_ban1          t2_ban2          t2_ban3          t2_ban4
##  Min.   : -1.0   Min.   : -1.0   Min.   : -1.0   Min.   : -1.0
##  1st Qu.: 38.0   1st Qu.: 37.0   1st Qu.: 38.0   1st Qu.: 38.0
##  Median : 90.0   Median : 90.0   Median : 90.0   Median : 90.0
##  Mean   :108.2   Mean   :107.9   Mean   :108.7   Mean   :108.6
##  3rd Qu.:141.0   3rd Qu.:141.0   3rd Qu.:141.0   3rd Qu.:141.0
##  Max.   :516.0   Max.   :516.0   Max.   :516.0   Max.   :516.0
##     t2_ban5
##  Min.   : -1.0
##  1st Qu.: 38.0
##  Median : 90.0
##  Mean   :108.1
##  3rd Qu.:141.0
##  Max.   :516.0
```

Una vez que tenemos los estadísticos básicos principales, vamos a limpiar algunos datasets de variables que no necesitamos para poder hacer análisis de variables con su correlación y similares...

```
# Ahora voy a hacer una tabla de games especial solo con las variables de las que calcularemos las corr
# Game ID es una variable que no necesito para nada, por lo que la voy a eliminar:

data.games.corr <- data.games[, c(-1, -4)]
#head(data.games.corr)
data.games.corr <- data.games.corr[, -10:-24]
#head(data.games.corr)
data.games.corr <- data.games.corr[, -15:-34]
#head(data.games.corr)
data.games.corr <- data.games.corr[, -20:-24]
head(data.games.corr)
```

```
##   creationTime gameDuration winner firstBlood firstTower firstInhibitor
## 1 1.504279e+12         1949      1          2          1              1
## 2 1.497849e+12         1851      1          1          1              1
## 3 1.504360e+12         1493      1          2          1              1
## 4 1.504349e+12         1758      1          1          1              1
## 5 1.504554e+12         2094      1          2          1              1
```

```
## 6 1.501668e+12         2059      1          2          2              1
##   firstBaron firstDragon firstRiftHerald t1_towerKills t1_inhibitorKills
## 1          1           1               2           11                 1
## 2          0           1               1           10                 4
## 3          1           2               0            8                 1
## 4          1           1               0            9                 2
## 5          1           1               0            9                 2
## 6          1           2               0            8                 1
##   t1_baronKills t1_dragonKills t1_riftHeraldKills t2_towerKills
## 1             2              3                  0             5
## 2             0              2                  1             2
## 3             1              1                  0             2
## 4             1              2                  0             0
## 5             1              3                  0             3
## 6             1              1                  0             6
##   t2_inhibitorKills t2_baronKills t2_dragonKills t2_riftHeraldKills
## 1                 0             0              1                  1
## 2                 0             0              0                  0
## 3                 0             0              1                  0
## 4                 0             0              0                  0
## 5                 0             0              1                  0
## 6                 0             0              3                  0
```

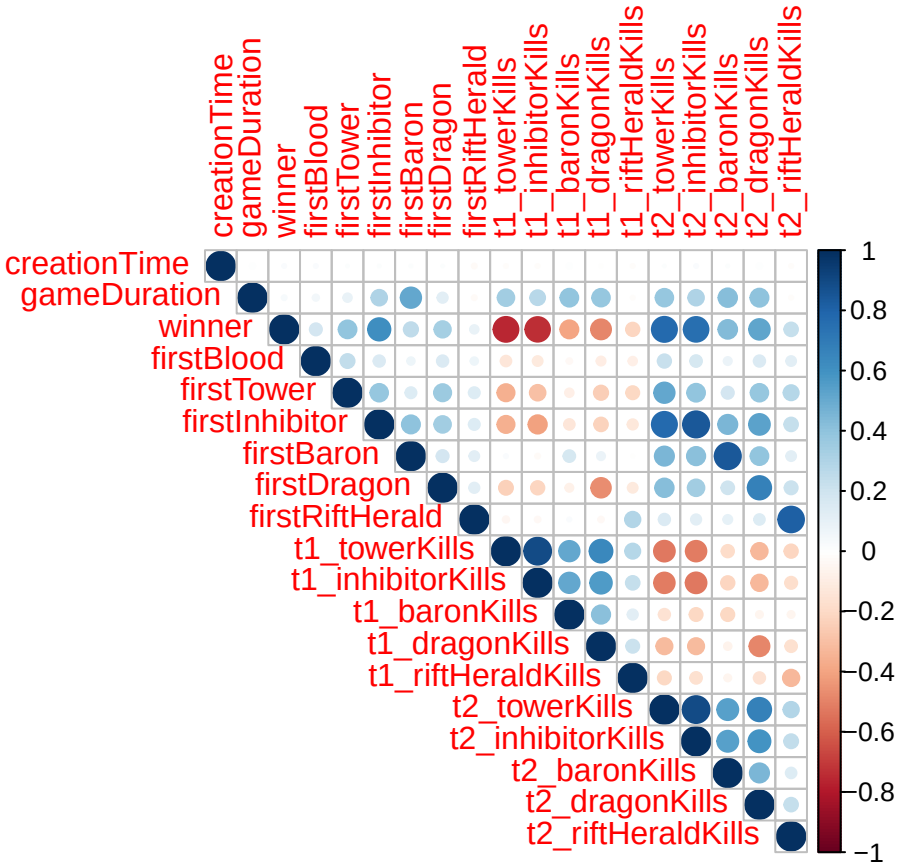
Correlation Plots

Lo primero que voy a hacer son los correlation plot de cada una de las tablas. Vamos a ver qué relaciones tenemos entre los datos, para poder ver si podemos ir sacando algunas conclusiones...

```
res.games <- cor(data.games.corr, method = "spearman")
options(width = 100)
res.games.round <- round(res.games, 2)
```

El primer gráfico que muestro es entre la duración de las partidas y la temporada en la que se está jugando:

```
corrplot(res.games.round, method="circle", type = "upper")
```



Podemos apreciar las siguientes correlaciones sobre destruir torres:

- Destruir más torres tiene una relación directa con destruir inhibidores
- Destruir más torres tiene una cierta relación directa con matar dragones
- Destruir más torres tiene una cierta relación inversa con que el equipo contrario destruya tus torres
- Destruir más torres tiene uan cierta relación con que el equipo contrario mate menos dragones

Respecto a ganar, obtenemos las siguientes correlaciones:

- Muchas veces, quien destruye el primer inhibidor es el equipo que acaba ganando la partida
- Un aumento en que el equipo 1 destruya más torres tiene que ver con que tengan más posibilidades de victoria. Esto se da con una relación inversa, ya que la victoria del equipo 1 se marca con un 1, la del 2 con un dos, y el aumento de towerkills de t1 implica una bajada en win (1 en vez de dos)
- Lo mismi pasa con la cantidad de inhibidores destruidos, donde es también una relación muy fuerte la que hay.
- Con el equipo 2 pasa lo mismo, lo que pasa es que, por lo explicado anteriormente, en este caso se muestra como relación direta, y de este modo podemos ver las mismas correlaciones de una parte que de otra.

Respecto a dragones, podemos ver:

- Es más importante para el equipo 2 hacerse el primer dragón de cara a hacerse más que el que el equipo 1 haga el mismo.

Finalmente, respecto a el momento de la creación de la partida, podemos observar:

- No tiene ninguna relación el paso del tiempo cno el aumento o disminución de ninguna de las variables.

Preguntas

Ahora que tenemos las correlaciones vistas, podemos pasar a hacernos preguntas.

¿Campeón más baneado por temporada?

Uno de los elementos que más se tienen en cuenta a la hora de analizar el League of Legends en los e-sports es el campeón más baneado por temporada o campeonato. Esto es porque desde Riot Games hacen cambios a las habilidades y fuerza de los personajes, de tal manera que algunas veces algunos campeones son demasiado fuertes y se pueden bloquear al inicio de la partida. Vamos a ver cuáles han sido:

Para analizar esto, contaré la cantidad de ocurrencias y veré, en estas más de 50.000 partidas, cual es el campeón más temido entre los jugadores, tras sustiruir por los nombres.

```
# Lo que voy a hacer es contar por columnas, y luego sumo todas las columnas que he contado. Así obteng

# TEAM 1

cont.bans.t1.1 <- ddply(data.games,.(t1_ban1),nrow)
cont.bans.t1.1 <- cont.bans.t1.1[order(cont.bans.t1.1$V1, decreasing = TRUE), ]
cont.bans.t1.2 <- ddply(data.games,.(t1_ban2),nrow)
cont.bans.t1.2 <- cont.bans.t1.2[order(cont.bans.t1.2$V1, decreasing = TRUE), ]
cont.bans.t1.3 <- ddply(data.games,.(t1_ban3),nrow)
cont.bans.t1.3 <- cont.bans.t1.3[order(cont.bans.t1.3$V1, decreasing = TRUE), ]
cont.bans.t1.4 <- ddply(data.games,.(t1_ban4),nrow)
cont.bans.t1.4 <- cont.bans.t1.4[order(cont.bans.t1.4$V1, decreasing = TRUE), ]
cont.bans.t1.5 <- ddply(data.games,.(t1_ban5),nrow)
cont.bans.t1.5 <- cont.bans.t1.5[order(cont.bans.t1.5$V1, decreasing = TRUE), ]

# TEAM 2

cont.bans.t2.1 <- ddply(data.games,.(t2_ban1),nrow)
cont.bans.t2.1 <- cont.bans.t2.1[order(cont.bans.t2.1$V1, decreasing = TRUE), ]
cont.bans.t2.2 <- ddply(data.games,.(t2_ban2),nrow)
cont.bans.t2.2 <- cont.bans.t2.2[order(cont.bans.t2.2$V1, decreasing = TRUE), ]
cont.bans.t2.3 <- ddply(data.games,.(t2_ban3),nrow)
cont.bans.t2.3 <- cont.bans.t2.3[order(cont.bans.t2.3$V1, decreasing = TRUE), ]
cont.bans.t2.4 <- ddply(data.games,.(t2_ban4),nrow)
cont.bans.t2.4 <- cont.bans.t2.4[order(cont.bans.t2.4$V1, decreasing = TRUE), ]
cont.bans.t2.5 <- ddply(data.games,.(t2_ban5),nrow)
cont.bans.t2.5 <- cont.bans.t2.5[order(cont.bans.t2.5$V1, decreasing = TRUE), ]

# Ahora lo que tengo que hacer es sumar todas estas columnas de V1 según el valor de name...

df.bans <- left_join(cont.bans.t1.1, cont.bans.t1.2, by =c("t1_ban1" = "t1_ban2"))
df.bans <- left_join(df.bans, cont.bans.t1.3, by =c("t1_ban1" = "t1_ban3"))
df.bans <- left_join(df.bans, cont.bans.t1.4, by =c("t1_ban1" = "t1_ban4"))
df.bans <- left_join(df.bans, cont.bans.t1.5, by =c("t1_ban1" = "t1_ban5"))

df.bans <- left_join(df.bans, cont.bans.t2.1, by =c("t1_ban1" = "t2_ban1"))
df.bans <- left_join(df.bans, cont.bans.t2.2, by =c("t1_ban1" = "t2_ban2"))
df.bans <- left_join(df.bans, cont.bans.t2.3, by =c("t1_ban1" = "t2_ban3"))
df.bans <- left_join(df.bans, cont.bans.t2.4, by =c("t1_ban1" = "t2_ban4"))
df.bans <- left_join(df.bans, cont.bans.t2.5, by =c("t1_ban1" = "t2_ban5"))

df.bans$total <- rowSums( df.bans[,2:11] )
```

```r
remove(cont.bans.t1.1)
remove(cont.bans.t1.2)
remove(cont.bans.t1.3)
remove(cont.bans.t1.4)
remove(cont.bans.t1.5)
remove(cont.bans.t2.1)
remove(cont.bans.t2.2)
remove(cont.bans.t2.3)
remove(cont.bans.t2.4)
remove(cont.bans.t2.5)

df.bans <- df.bans[order(df.bans$total, decreasing = TRUE), ]
df.bans <- df.bans[, -2:-11]
head(df.bans)
```

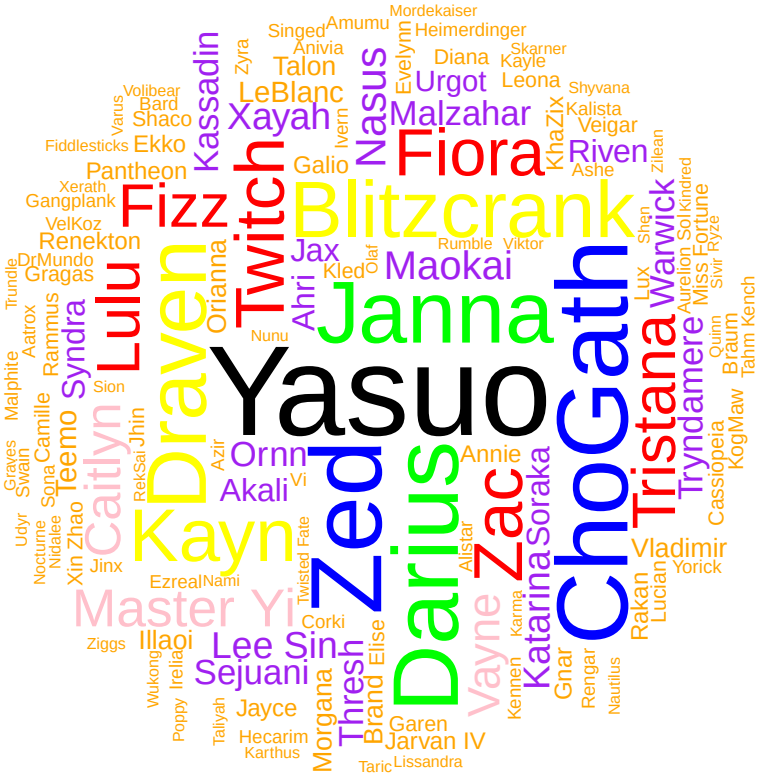
```
##   t1_ban1 total
## 1     157 33015
## 2     238 25393
## 3      31 25175
## 4     122 22870
## 5      40 21390
## 6     119 20262
```

```r
# Finalmente, junto con la tabla de data.champs para ponerles nombre...

df.bans <- left_join(df.bans, data.champs, by=c("t1_ban1" = "id"))
df.bans <- df.bans[, -1]
head(df.bans)
```

```
##   total    name
## 1 33015   Yasuo
## 2 25393     Zed
## 3 25175 ChoGath
## 4 22870  Darius
## 5 21390   Janna
## 6 20262  Draven
```

Ahora que tenemos todos los bans contados, es hora de hacer un wordcloud para poder verlo visualmente:

```r
set.seed(9999) # Para el mantenimiento del mismo patrón

wordcloud(words = df.bans$name, freq = df.bans$total, min.freq = 1, random.order=FALSE, rot.per=0.5, co
```




```
# Ratio del más baneado

print("El porcentaje de baneo a Yasuo es de: ")
```

```
## [1] "El porcentaje de baneo a Yasuo es de: "
```

```
ratio.ban.yasuo <- df.bans$total[1]/sum(df.bans$total)
print(ratio.ban.yasuo)
```

```
## [1] 0.06411925
```

Pregunta respondida.

¿Matar un mayor número de dragones aumenta las posibilidades de victoria?

Esto lo vamos a responder para el equipo 1 y para el equipo 2, de tal manera que podamos ver si tiene más influencia en uno u en otro.

Para esto, necesito la cantidad de dragones matados por el equipo que venció y por el equipo que perdió, y esos datos los tengo en el dataset de data.games.

```
data.dragons.win <- cbind(data.games$t1_dragonKills, data.games$t2_dragonKills, data.games$winner)
colnames(data.dragons.win) <- c("KillsT1", "KillsT2", "Win")
head(data.dragons.win)
```

```
##      KillsT1 KillsT2 Win
## [1,]       3       1   1
## [2,]       2       0   1
## [3,]       1       1   1
```

```
## [4,]       2       0   1
## [5,]       3       1   1
## [6,]       1       3   1
```

Primero vamos a estudiar la correlación. La hemos visto anteriormente, pero vamos a hacerlo ahora en especial de esta tabla para verlo de una manera más grande:

```
res.dragons.win <- cor(data.dragons.win, method = "spearman")
options(width = 100)
res.dragons.win.round <- round(res.dragons.win, 2)
```

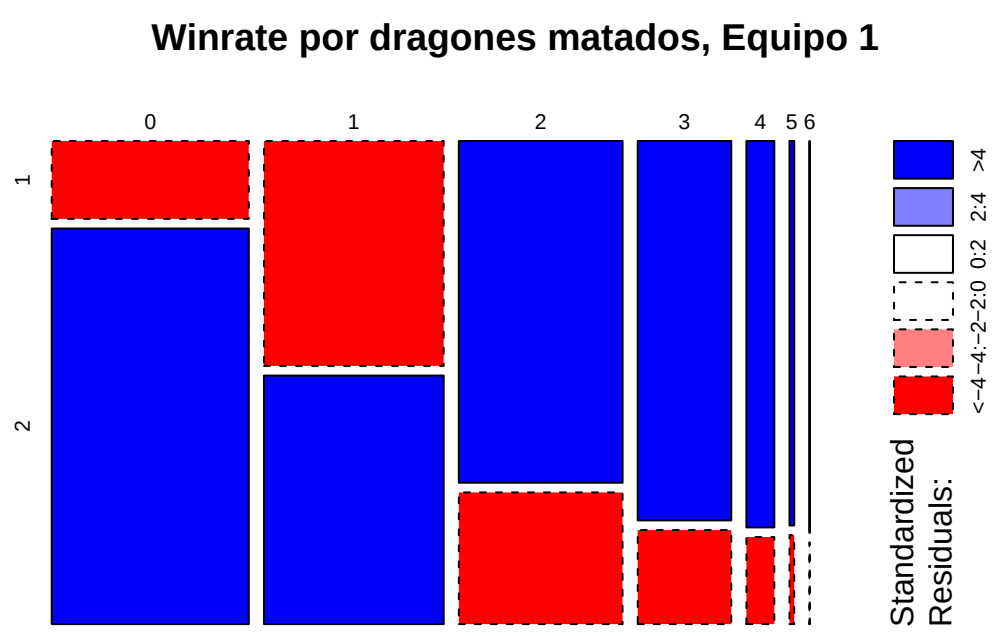
```
corrplot(res.dragons.win.round, method="circle", type = "upper", tl.srt = 0.7)
```

Como podemos ver, tenemos una correlación de aproximadamente el 50% entre el aumento del número de dragones matados y conseguir la victoria.

Equipo 1

Vamos a ver el número de dragones matados por el equipo 1 respecto a la victoria:

```
mosaicplot(table(data.dragons.win[, 1], data.dragons.win[, 3]), main='Winrate por dragones matados, Equ
```



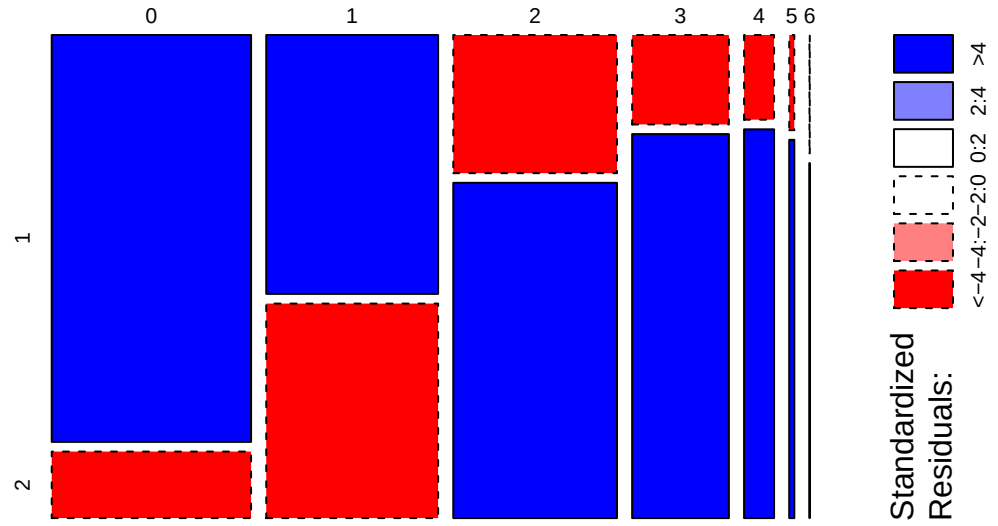
Como podemos ver, el equipo 1, si no mata a ningún dragón, tiene serias posibilidades de no ser el equipo que gane. Conforme va matando dragones, aumenta la posibilidad, especialmente con el paso de 1 a 2 dragones, donde se planta con una mayor parte de las posibilidades.

A partir de los 3 dragones matados, el hecho de matar más no tiene prácticamente influencia en el resultado final.

Equipo 2

Procedemos igual con el equipo 2:

```
mosaicplot(table(data.dragons.win[, 2], data.dragons.win[, 3]), main='Winrate por dragones matados, Equ
```



Como se puede observar, las distribuciones son prácticamente similares, solo que en espejo como es normal.

¿Cuales son los campeones más escogidos?

Sabemos que Yasuo es el campeón más baneado, con un 6% de banrate. Ahora me pregunto cual es el campeón más escogido para jugar.

Para ello, tenemos que seguir la misma metodología que en la parte de los baneos:

```
# Lo que voy a hacer es contar por columnas, y luego sumo todas las columnas que he contado. Así obteng

# TEAM 1

cont.picks.t1.1 <- ddply(data.games,.(t1_champ1id),nrow)
cont.picks.t1.1 <- cont.picks.t1.1[order(cont.picks.t1.1$V1, decreasing = TRUE), ]
cont.picks.t1.2 <- ddply(data.games,.(t1_champ2id),nrow)
cont.picks.t1.2 <- cont.picks.t1.2[order(cont.picks.t1.2$V1, decreasing = TRUE), ]
cont.picks.t1.3 <- ddply(data.games,.(t1_champ3id),nrow)
cont.picks.t1.3 <- cont.picks.t1.3[order(cont.picks.t1.3$V1, decreasing = TRUE), ]
cont.picks.t1.4 <- ddply(data.games,.(t1_champ4id),nrow)
cont.picks.t1.4 <- cont.picks.t1.4[order(cont.picks.t1.4$V1, decreasing = TRUE), ]
cont.picks.t1.5 <- ddply(data.games,.(t1_champ5id),nrow)
cont.picks.t1.5 <- cont.picks.t1.5[order(cont.picks.t1.5$V1, decreasing = TRUE), ]

# TEAM 2

cont.picks.t2.1 <- ddply(data.games,.(t2_champ1id),nrow)
```

```r
cont.picks.t2.1 <- cont.picks.t2.1[order(cont.picks.t2.1$V1, decreasing = TRUE), ]
cont.picks.t2.2 <- ddply(data.games,.(t2_champ2id),nrow)
cont.picks.t2.2 <- cont.picks.t2.2[order(cont.picks.t2.2$V1, decreasing = TRUE), ]
cont.picks.t2.3 <- ddply(data.games,.(t2_champ3id),nrow)
cont.picks.t2.3 <- cont.picks.t2.3[order(cont.picks.t2.3$V1, decreasing = TRUE), ]
cont.picks.t2.4 <- ddply(data.games,.(t2_champ4id),nrow)
cont.picks.t2.4 <- cont.picks.t2.4[order(cont.picks.t2.4$V1, decreasing = TRUE), ]
cont.picks.t2.5 <- ddply(data.games,.(t2_champ5id),nrow)
cont.picks.t2.5 <- cont.picks.t2.5[order(cont.picks.t2.5$V1, decreasing = TRUE), ]

# Ahora lo que tengo que hacer es sumar todas estas columnas de V1 según el valor de name...

df.picks <- left_join(cont.picks.t1.1, cont.picks.t1.2, by =c("t1_champ1id" = "t1_champ2id"))
df.picks <- left_join(df.picks, cont.picks.t1.3, by =c("t1_champ1id" = "t1_champ3id"))
df.picks <- left_join(df.picks, cont.picks.t1.4, by =c("t1_champ1id" = "t1_champ4id"))
df.picks <- left_join(df.picks, cont.picks.t1.5, by =c("t1_champ1id" = "t1_champ5id"))

df.picks <- left_join(df.picks, cont.picks.t2.1, by =c("t1_champ1id" = "t2_champ1id"))
df.picks <- left_join(df.picks, cont.picks.t2.2, by =c("t1_champ1id" = "t2_champ2id"))
df.picks <- left_join(df.picks, cont.picks.t2.3, by =c("t1_champ1id" = "t2_champ3id"))
df.picks <- left_join(df.picks, cont.picks.t2.4, by =c("t1_champ1id" = "t2_champ4id"))
df.picks <- left_join(df.picks, cont.picks.t2.5, by =c("t1_champ1id" = "t2_champ5id"))

df.picks$total <- rowSums( df.picks[,2:11] )

remove(cont.picks.t1.1)
remove(cont.picks.t1.2)
remove(cont.picks.t1.3)
remove(cont.picks.t1.4)
remove(cont.picks.t1.5)
remove(cont.picks.t2.1)
remove(cont.picks.t2.2)
remove(cont.picks.t2.3)
remove(cont.picks.t2.4)
remove(cont.picks.t2.5)

df.picks <- df.picks[order(df.picks$total, decreasing = TRUE), ]
df.picks <- df.picks[, -2:-11]
head(df.picks)
```

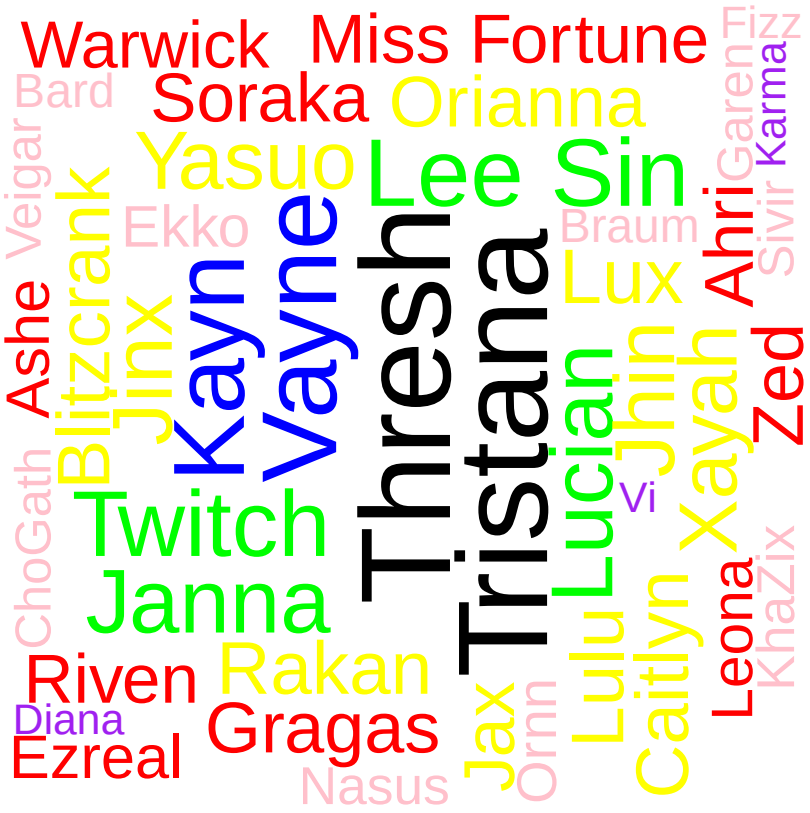
```
##   t1_champ1id total
## 2         412 13002
## 1          18 12983
## 3          67 10658
## 4         141  9853
## 5          64  9188
## 6          29  8838
```

```r
# Finalmente, junto con la tabla de data.champs para ponerles nombre...

df.picks <- left_join(df.picks, data.champs, by=c("t1_champ1id" = "id"))
df.picks <- df.picks[, -1]
head(df.picks)
```

```
##   total     name
## 1 13002   Thresh
## 2 12983 Tristana
## 3 10658    Vayne
## 4  9853     Kayn
## 5  9188  Lee Sin
## 6  8838   Twitch
```

Ahora que tenemos todos los picks contados, es hora de hacer un wordcloud para poder verlo visualmente:

```
set.seed(9998) # Para el mantenimiento del mismo patrón

wordcloud(words = df.picks$name, freq = df.picks$total, min.freq = 3000, random.order=FALSE, rot.per=0.5
```




```
# Ratio del más baneado

print("El porcentaje de pick de Thresh es de: ")
```

```
## [1] "El porcentaje de pick de Thresh es de: "
```

```
ratio.pick.thresh <- df.picks$total[1]/sum(df.picks$total)
print(ratio.pick.thresh)
```

```
## [1] 0.02525151
```

Está muy bien que Thresh y Tristana sean los más elegidos, pero... ¿esto es porque conllevan mayor victoria de partidas? Vamos a calcularlo:

¿Qué camepones conllevan un mayor winrate?

Para esto, lo primero que tenemos que hacer es crear un nuevo dataframe con los campeones que han ganado cada partida:

```
dataframe.winners.1 <- filter(data.games, winner == "1")

# En este dataframe, solo me interesan las columnas de los campeones elegidos por el equipo 1, por lo q

dataframe.winners.1 <- dataframe.winners.1[, c(12, 15, 18, 21, 24)]
colnames(dataframe.winners.1) <- c("Pick1", "Pick2", "Pick3", "Pick4", "Pick5")

# Hacemos lo mismo con los ganadores del equipo 2

dataframe.winners.2 <- filter(data.games, winner == "2")

# En este dataframe, solo me interesan las columnas de los campeones elegidos por el equipo 1, por lo q

dataframe.winners.2 <- dataframe.winners.2[, c(37, 40, 43, 46, 49)]
colnames(dataframe.winners.2) <- c("Pick1", "Pick2", "Pick3", "Pick4", "Pick5")

# Ahora los juntamos...

dataframe.winners <- rbind(dataframe.winners.1, dataframe.winners.2)

remove(dataframe.winners.1)
remove(dataframe.winners.2)
```

Ahora ya tenemos preparado el dataframe para poder jugar con él. Procedemos contando el número de ocurrencias que hay de cada campeón, y con esto veremos cual es el campeón que más se repite en este dataframe donde solo están las victorias:

```
# Ahora voy a llevar todas las columnas a una:

dataframe.winners <- data.frame(all = c(dataframe.winners[,"Pick1"], dataframe.winners[,"Pick2"], dataf

head(dataframe.winners)
```

```
##   all
## 1   8
## 2 119
## 3  18
## 4  57
## 5  19
## 6  40
```

```
dataframe.winners <- left_join(dataframe.winners, data.champs, by=c("all" = "id"))

dataframe.winners <- dataframe.winners[, -1]

head(dataframe.winners)
```

```
## [1] Vladimir Draven   Tristana Maokai   Warwick  Janna
## 138 Levels: Aatrox Ahri Akali Alistar Amumu Anivia Annie Ashe Aurelion Sol Azir Bard ... Zyra
```

```
# Ahora tenemos todo el dataframe en una única columna, donde solo nos queda ordenar y contar el número
```

```
dataframe.winners <- dataframe.winners[order(dataframe.winners)]
head(dataframe.winners)
```

```
## [1] Aatrox Aatrox Aatrox Aatrox Aatrox Aatrox
## 138 Levels: Aatrox Ahri Akali Alistar Amumu Anivia Annie Ashe Aurelion Sol Azir Bard ... Zyra
```

```
# Contamos

cantidad <- as.data.frame(table(dataframe.winners))

cantidad <- cantidad[order(cantidad$Freq, decreasing = TRUE), ]
colnames(cantidad) <- c("name", "freq")
head(cantidad)
```

```
##         name freq
## 112 Tristana 6713
## 111   Thresh 6143
## 120    Vayne 5498
## 42     Janna 4826
## 54      Kayn 4807
## 116   Twitch 4665
```

Como podemos ver, los campeones más elegidos son Tristana y Thresh, que casualmente, como acabamos de probar, son los que tienen más victorias en su haber. También, la tercera con más picks es Vayne, y también es la tercera en cantidad de victorias.

Vamos a calcular el ratio pick-victory:

```
# Para hacer esto, voy a juntar los dos dataframes en uno solo, juntando por el valor del campeón (nomb

# No puedo hacer un join normal porque todos mis elementos de texto son factors, por eso lo reconvierto

cantidad <- data.frame(cantidad, stringsAsFactors = FALSE)
colnames(cantidad) <- c("name", "freq_win")

df.picks <- data.frame(df.picks, stringsAsFactors = FALSE)
colnames(df.picks) <- c("total_picks", "name")

dataframe.pick.win <- inner_join(cantidad, df.picks, by="name")

# Ahora que tengo los datos a mano, podemos hacer las divisiones:

dataframe.pick.win <- transform(dataframe.pick.win, ratio = dataframe.pick.win[, 2] / dataframe.pick.wi

dataframe.pick.win <- dataframe.pick.win[order(dataframe.pick.win$ratio, decreasing = TRUE), ]

head(dataframe.pick.win)
```

```
##       name freq_win total_picks     ratio
## 4    Janna     4826        8691 0.5552871
## 29    Sona     2942        5429 0.5419046
## 120 Yorick      744        1378 0.5399129
## 66  Rammus     1614        2997 0.5385385
## 87  Anivia     1207        2252 0.5359680
## 118 Singed      762        1425 0.5347368
```

Es interesante ver que, a la vista de estos resultados, la gente banea lo que ve mucho, no lo que de verdad

gana partidas. Como podemos observar, Janna, Sona y Yorick son los 3 campeones que más porcentaje de partidas ganan. En cambio, a la hora de escogerlos los vemos en las posiciones 7, 30 y 124. Respecto a bans, si tantas partidas ganan la masa debería de banearlos, pero se encuentran en las posiciones 5, 91 y 96, por lo que la gente no se da cuenta del peligro de estos campeones en las manos adecuadas.

¿El lado rojo (equipo 2) pierde más partidas?

Una de las afirmaciones que corre por la comunidad de League of Legends es la creencia de que el lado rojo, correspondiente con el equipo 2 en nuestro dataset, es el que pierde un mayor número de partidas. Para visualizar esto, lo único que hay que hacer es obtener el winrate del equipo rojo respecto al total de partidas:

```
columna.wins <- data.games$winner
cantidad.wins <- as.data.frame(table(columna.wins))
colnames(cantidad.wins) <- c("winner", "frequency")
head(cantidad.wins)
```

```
##   winner frequency
## 1      1     26077
## 2      2     25413
```

```
# Si calculamos el ratio...

ratio.red <- cantidad.wins[2, 2] / (cantidad.wins[1, 2] + cantidad.wins[2, 2])
ratio.red
```

```
## [1] 0.4935521
```

```
ratio.blue <- cantidad.wins[1, 2] / (cantidad.wins[1, 2] + cantidad.wins[2, 2])
ratio.blue
```

```
## [1] 0.5064479
```

Es decir, el grupo que juega en el lado rojo gana el 49,35% de las partidas, mientras que el que juega en el lado azul gana el 50,64%.

Vamos a verlo en forma de gráfica:

```
ratios_win <- c(ratio.blue, ratio.red)
barplot(ratios_win, main="Victorias por equipo", ylab="Equipo", col = c("Blue", "Red"))
```

¿Quién consigue más primeros objetivos?

Los primeros objetivos son cruciales en la partida, ya que algunos dan extra de oro, pero todos empiezan a decantar la partida a tu favor. Debido a ello, el análisis de si existe alguna diferencia entre los equipos a la hora de conseguir los primeros objetivos se plantea crucial.

Para ello, contaré para cada columna de primeros objetivos qué equipo ha conseguido ser el mejor, y por qué porcentaje, y entonces determinaré si hay alguna ventaja en alguno de los objetivos.

PCA

Vamos a hacer ahora un análisis de las componentes principales: Para los cálculos, uso la matriz con el centrado y escalado ya hechos

```
resultado.pca <- PCA(data.games.corr, graph = FALSE)

#Con la siguiente línea podemos ver que podemos hacer con esto calculado
print(resultado.pca)
```

```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 51490 individuals, described by 19 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
```

```
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Nos interesa ver los eigenvalues, que son los que presentarán la cantidad de varianza que aportan las variables:

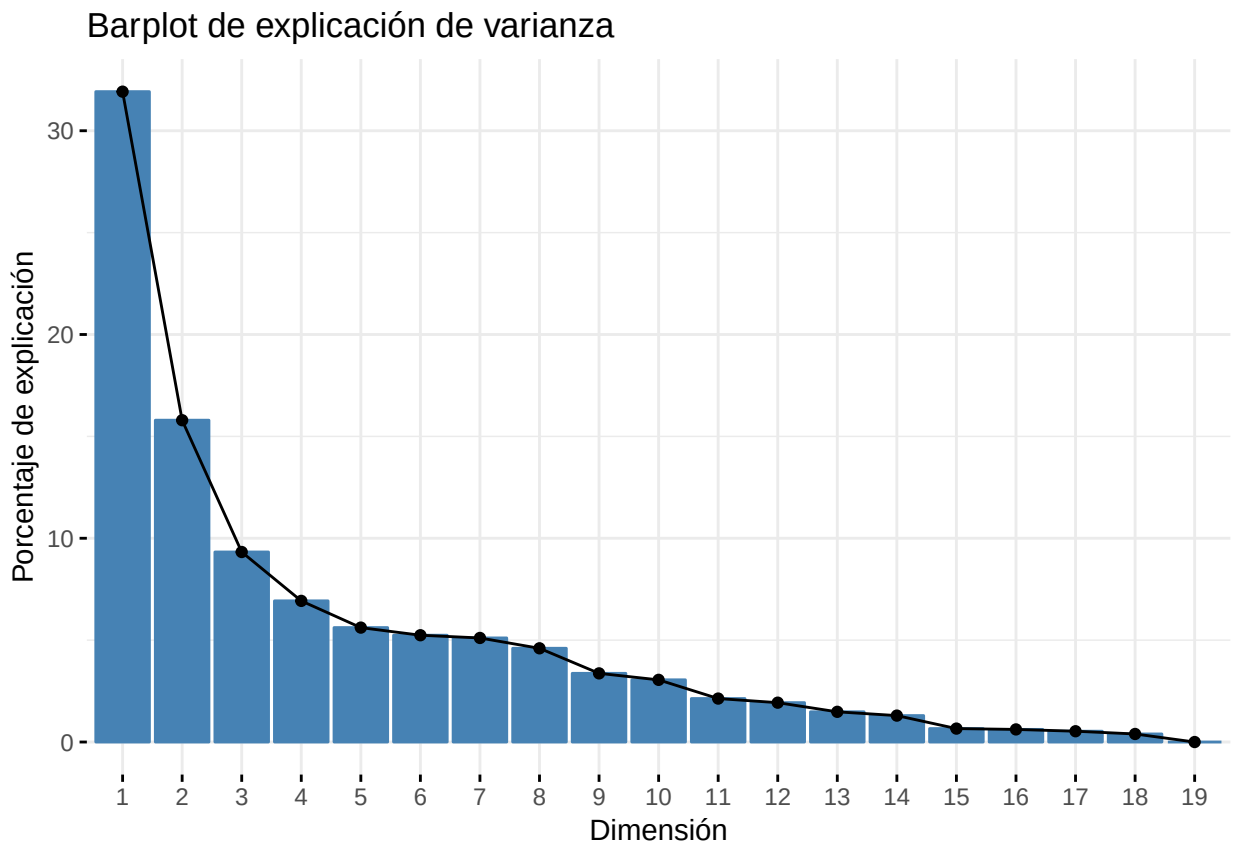
```
eigenvalues.PCA <- resultado.pca$eig
eigenvalues.PCA
```

```
##           eigenvalue percentage of variance cumulative percentage of variance
## comp 1  6.064459e+00           3.191820e+01                           31.91820
## comp 2  3.000738e+00           1.579336e+01                           47.71156
## comp 3  1.771844e+00           9.325495e+00                           57.03706
## comp 4  1.316075e+00           6.926710e+00                           63.96377
## comp 5  1.066890e+00           5.615210e+00                           69.57898
## comp 6  9.958756e-01           5.241450e+00                           74.82043
## comp 7  9.704525e-01           5.107645e+00                           79.92807
## comp 8  8.740157e-01           4.600083e+00                           84.52816
## comp 9  6.403057e-01           3.370030e+00                           87.89819
## comp 10 5.797901e-01           3.051527e+00                           90.94971
## comp 11 4.053622e-01           2.133485e+00                           93.08320
## comp 12 3.670873e-01           1.932039e+00                           95.01524
## comp 13 2.816983e-01           1.482623e+00                           96.49786
## comp 14 2.465127e-01           1.297435e+00                           97.79529
## comp 15 1.253587e-01           6.597825e-01                           98.45508
## comp 16 1.175431e-01           6.186477e-01                           99.07372
## comp 17 1.007805e-01           5.304239e-01                           99.60415
## comp 18 7.521185e-02           3.958518e-01                          100.00000
## comp 19 7.248996e-26           3.815261e-25                          100.00000
```

Como podemos ver, tenemos 19 componentes principales (una por cada dimensión), y vemos que solo con 10 variables ya tenemos un 90% de explicación. Además, de cara a la representación en dos dimensiones, podemos ver que con solo las dos variables con más varianza tenemos una explicación del 47,7%.

Ahora, para completar este apartado de PCA, lo que voy a hacer es sacar la gráfica de la varianza acumulada con los valores anteriores:

```
plotPCA <- fviz_screeplot(resultado.pca, ncp=19, main="Barplot de explicación de varianza", ylab="Porce
plot(plotPCA)
```



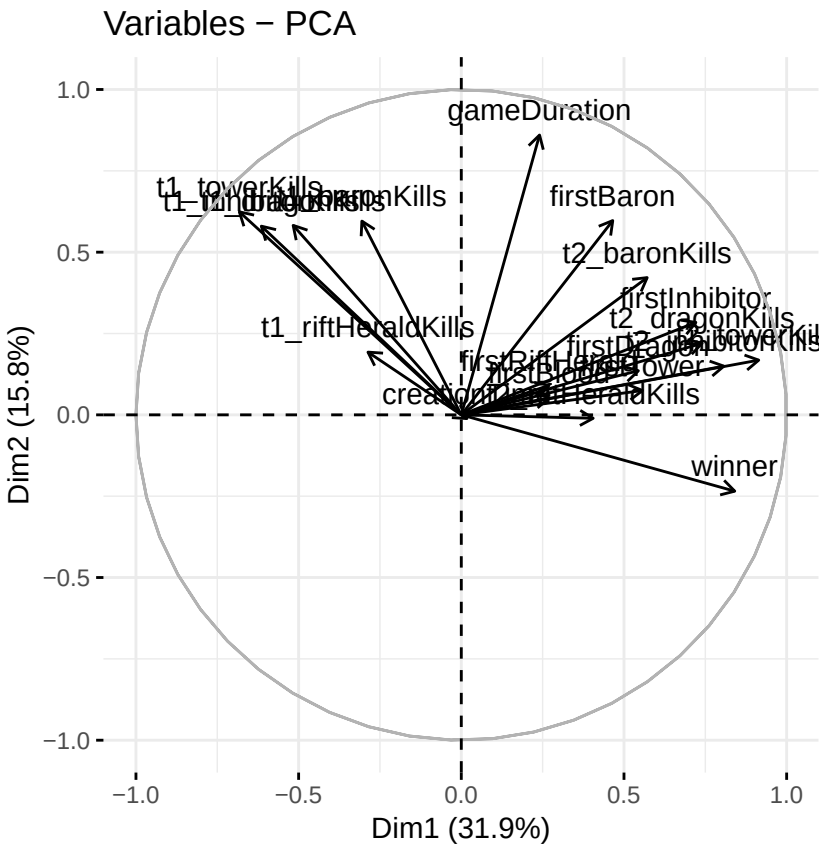
Ahora voy a sacar un "Factor Map" de las variables. Esto lo puedo hacer gracias a las coordenadas que me da una de las variables tras hacer el PCA. Así, voy primero a ver la tabla y luego voy a sacar el mapa:

```
head(resultado.pca$var$coord)
```

```
##                     Dim.1       Dim.2       Dim.3       Dim.4       Dim.5
## creationTime   0.01723817 -0.01149443 -0.06290260  0.04393011 -0.25111184
## gameDuration   0.24050089  0.86002372 -0.09989262  0.13764973 -0.07795331
## winner         0.84048823 -0.23457181 -0.14685233 -0.11282396 -0.04006251
## firstBlood     0.26968344  0.05182651  0.13357189  0.34780625 -0.23151329
## firstTower     0.55683600  0.07472362  0.19566560  0.37066228 -0.14932506
## firstInhibitor 0.72099264  0.28414084 -0.07367165 -0.02461892 -0.03806185
```

Como se puede ver, me está poniendo mis 24 variables en 5 dimensiones, con unas coordenadas concretas. Ahora, lo que voy a hacer, es representarlo. Con esta representación podré sacar algunas conclusiones:

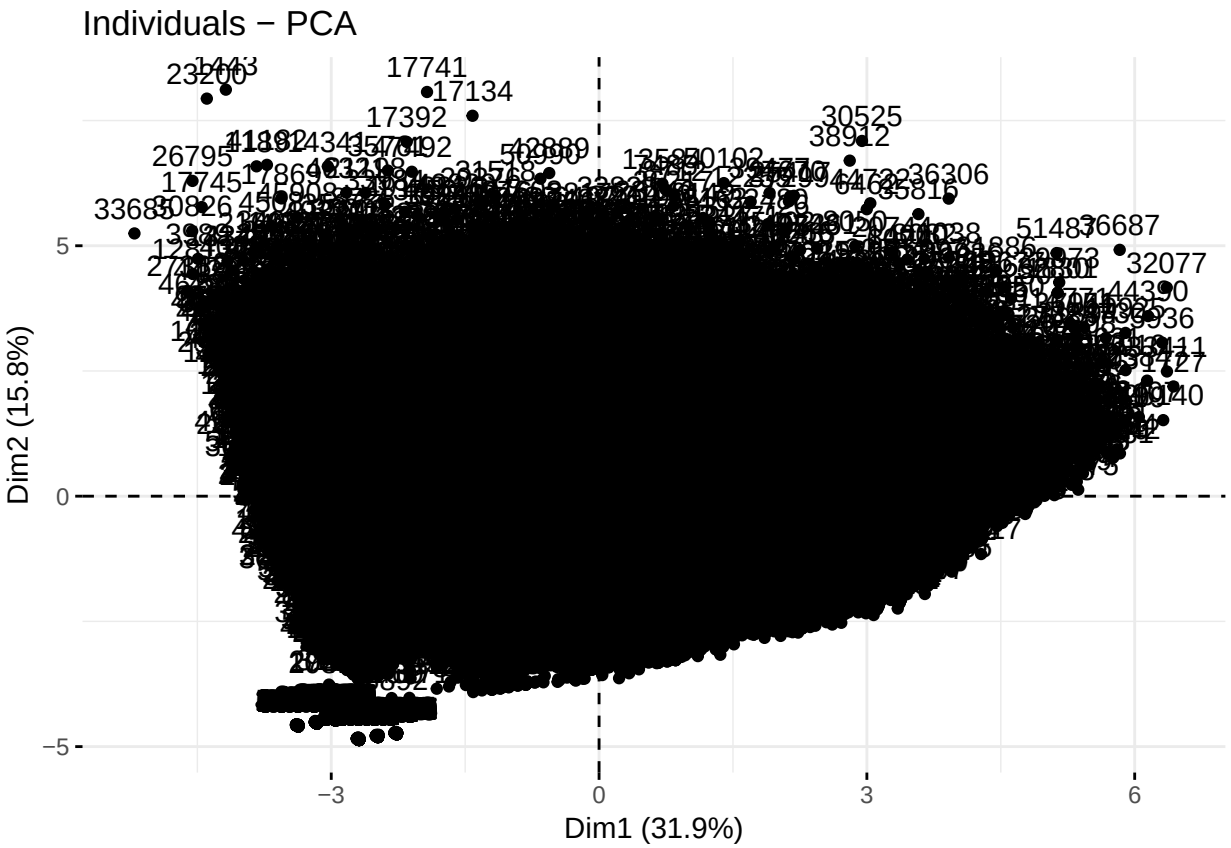
```
fviz_pca_var(resultado.pca)
```



Es interesante ver como no hay nada que vaya en la misma dirección que la victoria. T1 tiene que ser interpretado de una manera en espejo respecto al eje de las Y, y vemos como muchísimas variables como el heraldo, first blood, número de dragones... todas tienen más o menos la misma importancia de cara a conseguir la victoria.

Vemos como la duración del juego forma un ángulo de 90 grados con la victoria, lo cual significa que no tiene nada que ver.

```
fviz_pca_ind(resultado.pca)
```



Como se puede apreciar en este gráfico también, prácticamente todas las partidas forman un gran cluster (representado sobre las dos componentes con más variabilidad), y esto nos demuestra que no hay demasiada varianza entre unas partidas y otras. En otras palabras, no podemos distinguir fácilmente varios tipos de partidas, todas son parecidas y no hay diferencias suficientes para clasificar en grupos.