Mad Lions EC Colombia - Scrims CS Report

Tet

20 - 01 - 2018

Script para obtener los datos deseados

```
library(googlesheets)
library(tidyverse)
library(lubridate)
library(jsonlite)

key <- readLines("API/API key.txt", warn=F)
servers <- c("https://la1.api.riotgames.com","https://na1.api.riotgames.com")
names(servers) =c("LAN","NA")
by_match <- "/lol/match/v4/timelines/by-match/"
mes <- "enero"
```

```
#This line may ask to authenticate using a browser
gs_ls()
```

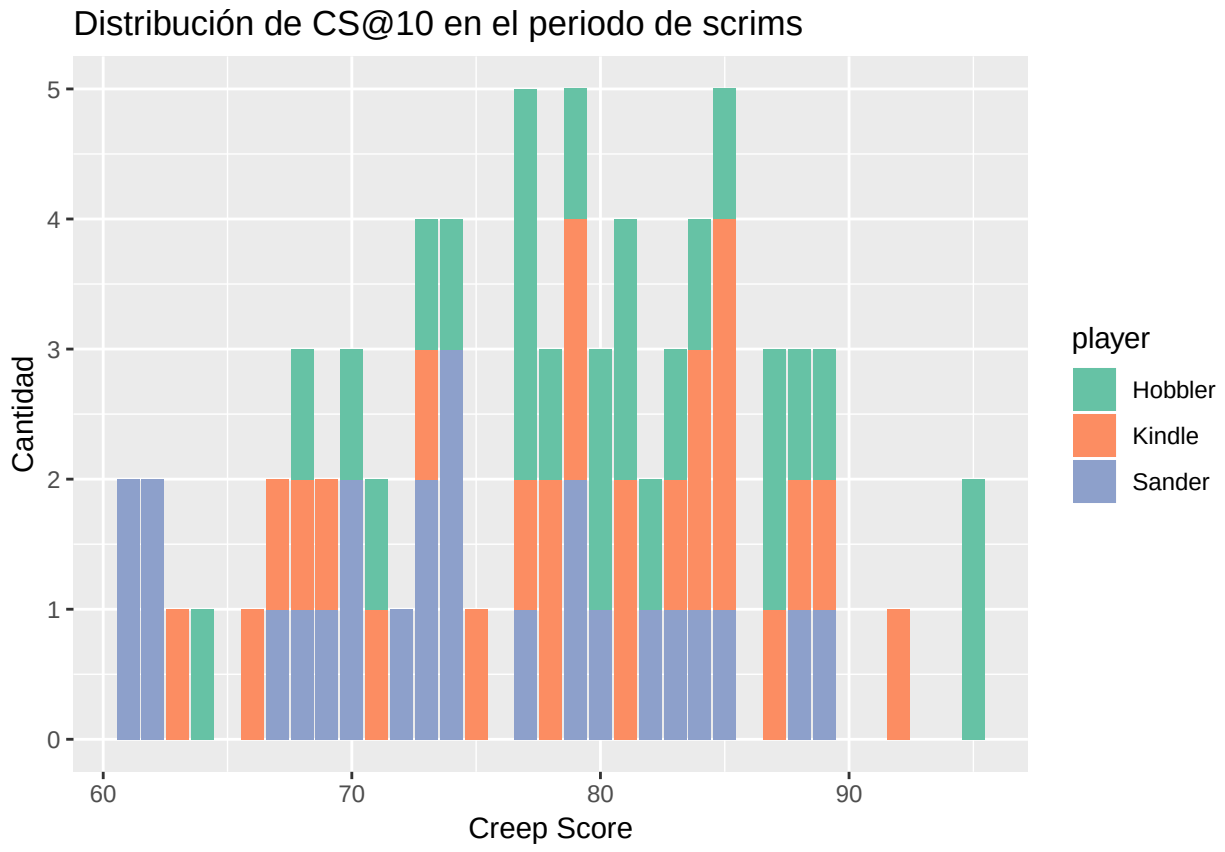
```
#get the match history
mh <- gs_url("https://docs.google.com/spreadsheets/d/1WAXDVqF0Bm2QvADV76uPd78n03oLLl_NJVFVfItHKHE/")

# get the raw match_history sheet
training_history <- gs_read(ss=mh, ws = "match_history", range = "B4:BA9")

#fixing for the lack of last character with an NA
fix <- training_history[length(training_history)-1]
names(fix) = paste0("X",as.integer(str_replace(names(fix),"X","")) + 2)
training_history <- cbind(training_history,fix)

#Cleaning the links and dates
historial <- training_history %>%
  gather(Date, link) %>%
  mutate(Date = dmy(Date)) %>%
  filter(!is.na(Date))
#Extracting which side we played on
lados <- training_history %>%
  select(num_range("X",seq(2,length(training_history), by = 2)))
colnames(lados) <- unique(historial$Date)
lados <- lados %>% gather(Date, lado) %>%
  mutate(Date = ymd(Date))

#Joining the tables and cleaning up the link
scrims <- as.tbl(cbind(historial, lados$lado), stringsAsFactors = FALSE) %>%
  mutate(server = str_extract(link, "LA1|NA1")) %>%
  mutate(link =
           str_replace(link,
                       "https://matchhistory.lan.leagueoflegends.com/es/#match-details/LA1/", "")) %>%
  mutate(link =
           str_replace(link,
```

```
                        "https://matchhistory.na.leagueoflegends.com/en/#match-details/NA1/", "")) %>%
  mutate(link = strtrim(link, 10)) %>%
  mutate(link = str_remove(link, "/")) %>%
  mutate(`lados$lado` = as.character(`lados$lado`))
names(scrims) = c("date","match_id","lado","server")

#This function extracts the general info of the matches
#includes: position, currentgold, totalgold, level, xp, minionsKilled, jungleMinionsKilled
extract_match_data <- function(match_id, server) {
  if (!is.na(match_id)) {
    if (server == "LA1") {
      fromJSON(paste0(servers[1],by_match,match_id,"?api_key=",key))[[1]][[1]]
    } else if (server == "NA1") {
      fromJSON(paste0(servers[2],by_match,match_id,"?api_key=",key))[[1]][[1]]
    }
  } else {
    NA
  }
}

#filtering games on the blue side and adding general info
blue_side <- scrims %>% filter(lado == "azul")
blue_side_games <-  map2(.x = blue_side$match_id, .y = blue_side$server, .f = extract_match_data) %>%
  map(~ .[1:5])

#filtering games on the red side and adding general info
red_side <- scrims %>% filter(lado == "rojo")
red_side_games <-  map2(red_side$match_id, red_side$server, extract_match_data) %>%
  map(~ .[6:10])

#mixing all the games regardless of the participant id, all these are our games
our_games <- purrr::flatten(list(blue_side_games, red_side_games))
names(our_games) <- paste("game",1:length(our_games))


#Flattened the positions data frame inside each game
for (i in seq_along(our_games)) {
  our_games[[i]] <- our_games[[i]]  %>%  map(jsonlite::flatten)
}

#This function selects only the useful information for the given player
info_selector <- function(filtered_games, player_number) {
  map(filtered_games,player_number) %>%
    map(~select(., -teamScore, -dominionScore))
}

#These are the matches separated by player with only the useful info selected
top <- info_selector(our_games, 1)
jungle <- info_selector(our_games, 2)
mid <- info_selector(our_games, 3)
adc <- info_selector(our_games, 4)
support <- info_selector(our_games, 5)
```

```r
#A function to get the relevant lane cs for each player
get_lane_cs <- function(games, player) {
  games %>% map(~map(.x,"minionsKilled")) %>%
    map(player) %>%
    map(possibly(~data.frame(min5 = .x[[6]], min10 = .x[[11]], min15 = .x[[16]],
                             min20 = .x[[21]], lastminute = last(.x)),
                 otherwise = data.frame(min5 = NA, min10 = NA, min15 = NA, min20 = NA, lastminute = NA)))
}
#Subsetting the cs for each player to study
top_cs <- get_lane_cs(our_games, 1)
mid_cs <- get_lane_cs(our_games, 3)
adc_cs <- get_lane_cs(our_games, 4)

#Building the data frame to use with ggplot2
cs_10 <-  tibble(games = names(top_cs), Sander = map_dbl(top_cs, "min10"),
                 Hobbler = map_dbl(mid_cs, "min10"), Kindle = map_dbl(adc_cs, "min10")) %>%
  gather(player, cs, -games)
```

Gráficas

Cada barra de longitud 1 en `cantidad` representa un juego en el periodo de scrims en el que alcanzó el cs al minuto 10 correspondiente en `CreepScore`. El color de las barras representa los jugadores de interés.



Guardar los scrims de enero

Utilizando el siguiente código se guarda el *timeline* en los scrims del mes enero en formato `JSON`

```
toJSON(our_games) %>% write("enero_timeline.json")
```