Trabralho AV1

Erick Brito e Vinícius Mitsuo

Projeto

Os Esportes Eletrônicos ou apenas eSports, como é mundialmente conhecido, vêm dominando o mundo gamer e tecnológico. Numa definição simples, os eSports giram em torno de competições de jogos eletrônicos em que atletas profissionais disputam partidas, seja presencial ou on-line. Na maioria delas, os espectadores estão presentes no local do evento ou em plataformas de streaming.

Dado o crescimento dos Esportes Eletrônicos, assim como o crescimento desse mercado, decidimos então realizar uma análise exploratória de dados e visualizações em uma base de dados de partidas oficiais de um jogo específico, chamado Counter Strike: Global Offensive, ou CS:GO para os mais íntimos.

Motivação

A sensação dos Esportes Eletrônicos fez com que cada vez mais empresas/organizações se interessassem e investissem nesse mercado. No caso do CS:GO, isso não é diferente.

Hoje em dia existem organizações gigantes participando do cenário competitivo de CS:GO, como por exemplo, Team Liquid, FaZe Clan, TSM, entre outras mais, que são muito valiosas e possuem times profissionais excelentes. Essas organizações estão constantemente alterando o elenco de seus times e procurando um meio de se sobressair no cenário competitivo.

Alterar um time, buscar uma melhora técnica em jogo, criar táticas, tudo isso depende de informações, depende de estar seguindo o meta – Meta pode ser usado como um acrônimo para "most effective tactics available", em português "táticas mais eficazes disponíveis", e chamar algo de "meta" significa que é uma maneira eficaz de atingir o objetivo do jogo, seja vencer outros jogadores ou vencer o próprio jogo.

Portanto, uma análise de dados trazendo informações sobre o jogo(parte tática e mecânica) e sobre os jogadores se faz necessária e é de interesse de organizações/empresas que estão nesse meio. Daí, nossa motivação para começar esse projeto.

Base de Dados

A base de dados que escolhemos está disponível em CS:GO Professional Matches.

Nessa base temos as seguintes tabelas:

- **results.csv**: com as pontuações do mapa e classificações da equipe
- **picks.csv**: com a ordem de escolha de mapas e vetos no processo de seleção de mapas.
- **economy.csv**: com o valor do equipamento inicial da rodada para todas as rodadas jogadas em um partida
- **players.csv**: com as performances individuais dos jogadores em cada mapa.

Lendo as Bases

Usando a função read.csv2() do R Base, conseguimos ler e salvar em variáveis as nossas tabelas.

```r
### Usando uma função do R Base para ler os csvs e assim os armazenamos nas seguintes variáveis
setwd("/home/erick/Documents/MAP/5prd/Introdução ao R aplicado em Ciência de Dados/A1")
players <- read.csv2("players.csv", sep=",")
results <- read.csv2("results.csv", sep=",")
picks <- read.csv2("picks.csv", sep=",")
economy <- read.csv2("economy.csv", sep=",")

### Transformando em tibble
players <- as_tibble(players)
results <- as_tibble(results)
picks <- as_tibble(picks)
economy <- as_tibble(economy)
```

Dando uma olhada

```r
head(players[1:4],1)
```

```
## # A tibble: 1 x 4
##   date       player_name team          opponent
##   <chr>      <chr>       <chr>         <chr>
## 1 2020-02-26 Brehze      Evil Geniuses Liquid
```

Limpeza dos dados

Analisando os dados, percebemos que haviam alguns vazios, apenas como " ". Então, decidimos verificar quantas linhas com vazio tem cada coluna:

- **players__empty__strings**: tibble que contém a soma das linhas com " " na tabela **players.cs** por coluna.
- **results__empty__strings**: tibble que contém a soma das linhas com " " na tabela **results.csv** por coluna.
- **picks__empty__strings**: tibble que contém a soma das linhas com " " na tabela **picks.csv** por coluna.
- **economy__empty__strings**: tibble que contém a soma das linhas com " " na tabela **economy.csv** por coluna.

```r
### Verificamos a soma de dados que estão da forma "" para cada coluna

### (Obs: a função funs() foi descontinuada, mas usando as funções similares
### recomendadas não conseguimos o mesmo resultado, algumas até geraram erros.)

players_empty_strings <- players %>%
  summarise_all(funs(sum(. == "")))

results_empty_strings <- results %>%
  summarise_all(funs(sum(. == "")))

picks_empty_strings <- picks %>%
  summarise_all(funs(sum(. == "")))

economy_empty_strings <- economy %>%
  summarise_all(funs(sum(. == "")))
```

Com isso, conseguimos ter uma noção de quantas observações(linhas) podemos remover para que assim alcancemos a base limpa.

Observando **results__empty__strings** e **picks__empty__strings** percebemos que elas já estão limpas e prontas para uso. Mas ao observar as **players__empty__strings**(o mesmo ocorre para **economy__empty__strings**) temos, em grandes quantidades até, dados faltando em algumas colunas:

```
players_empty_strings[10:15]
```

```
## # A tibble: 1 x 6
##   best_of map_1  map_2  map_3 kills assists
##     <int> <int>  <int>  <int> <int>   <int>
## 1       0     0 186794 314639     0       0
```

Como por exemplo as colunas *map_2* e *map_3*, que mostram quais foram os segundos e terceiros mapas jogados em uma partida. Isso só mostra que existe uma grande quantidade de partidas que não possui segundo e terceiro mapa na nossa base de dados, mas com isso já temos um direcionamento para alcançar a base limpa.

Com essas informações e usando também do nosso conhecimento sobre algumas coisas do jogo, uma limpeza que pode ser feita de já, é remover algumas colunas que já sabemos que não vão ser necessárias para a análise,

```
### Aqui estamos removendo colunas que estão com grande quantidade de dados faltando
### e que não virão a ser tão utilizadas.
players <- players %>%
  select(-flash_assists, -kast, -adr,-c(m1_kills:m3_rating_t)) %>%
  subset(player_name!="")

### Removendo colunas que não importarão para nossa análise, decidimos analisar
### só certos rounds, que acontecem de certo em uma partida. No CS:GO uma partida
### tem até 30 rounds no "tempo regulamentar", podendo ter muito mais caso essa
### partida vá pro overtime. Mas de certo, 16 rounds ocorrem em qualquer partida
### então decidimos analisar apenas esses.
economy <- economy %>%
  select(-c(X17_t1:X30_t1), -c(X17_t2:X30_t2), -c(X17_winner:X30_winner))
```

Análises

Jogadores profissionais por país (2018-2020)

Vamos dar uma olhada em como estão distribuidos os jogadores profissionais de CSGO pelo mundo:

```
### Transformando
players$date <- ymd(players$date)
economy$date <- ymd(economy$date)
picks$date <- ymd(picks$date)
results$date <- ymd(results$date)

### Escolhendo o ano a ser considerado
num_jog_pais <- players %>% filter(year(date) >= 2018)

### Selecionando valores únicos e contando
num_jog_pais <-distinct(num_jog_pais, player_name, .keep_all = TRUE)
num_jog_pais <- num_jog_pais %>% count(country)


### Separando os intervalos
breaks <- seq(floor(min(num_jog_pais$n)),
              ceiling(max(num_jog_pais$n)),
              by = 150)
```

```
### Escolhendo a cor da palheta
pal <- brewer.pal(n = length(breaks), name = "Blues")

### Dicionário para os intervalos de classe e cores associadas.
cl <- sprintf("{values:[%s],\n colors:[%s]}",
              paste0(sprintf("'%0.1f'", breaks), collapse = ", "),
              paste0(sprintf("'%s'", pal), collapse = ", "))

### Mapa do número de jogadores por pais (2018-2020)
map <- gvisGeoChart(
  data = num_jog_pais,
  locationvar = "country",
  colorvar = "n",
  options = list(title = "Número de jogadores por país / 2018 - 2020",
                 region = "world",
                 displayMode = "regions",
                 resolution = "countries",
                 colorAxis = cl,
                 width = 640,
                 height = 480))
#plot(map)
```

Dinheiro em premiação recebido por uma das maiores organizações no CS

Só para que tenhamos uma noção do tamanho do mercado de esportes eletrônicos, vamos analisar quanto dinheiro(em dólares) a FaZe Clan, uma das organizações mais valiosas de eSports no mundo, já ganhou em campeonatos de CS:GO, desde 2016 até 2022.

Para isso, vamos raspar dados da Liquipedia, uma wiki pertencente à TeamLiquid, sendo um dos recursos mais usados em todos os esports.

```
### Mostrando o tamanho do mercado
### Pagina fonte
html <- read_html("https://liquipedia.net/counterstrike/FaZe_Clan/Results")

### Pegando a tabela de resultados
faze_results <- html %>%
  html_nodes(".table-responsive") %>%
  html_table() %>% .[[1]]


### Limpando o data frame

# Escolhendo apenas as colunas relevantes
faze_results <- select(faze_results, c(1,2,3,4,7,8,10))

# Removendo linhas inúteis da tabela dinâmica
faze_results <- faze_results %>%
  filter(!row_number() %in% c(1, 8, 30, 49, 74, 98, 122))

# Transformando as datas em datas do lubridate
faze_results$Date <- ymd(faze_results$Date)

# Transformando string de prêmio em dólares em double
```

```
faze_results$Prize <- parse_number(faze_results$Prize)

# Transformando os valores N.A's referentes à não premiação em 0
faze_results[is.na(faze_results)] <- 0

# Resumo de quanto a organização ganhou em torneios jogados de 2016-2020
resumo_faze <- summary(faze_results$Prize)
resumo_faze["Sum"] <- sum(faze_results$Prize)
resumo_faze["Tournaments"] <- nrow(faze_results)

resumo_faze
```

```
##        Min.     1st Qu.      Median        Mean     3rd Qu.        Max.
##           0        1527       10000       35551       30000      500000
##         Sum Tournaments
##     4977197         140
```

Top first killers

First kill é, como o nome diz, a primeira kill que acontece no round. Geralmente, a porcentagem de conversão de rounds 5x4 é alta, assim, começar o round com um jogador a mais é uma grande vantagem.

Uma análise relevante do ponto de vista de um apostador, de uma organização, ou de um jogador, seria sobre os jogadores com maior média de first kills por partida. Para um time, começar um round com vantagem numérica é importante, então, se uma organização está procurando por um bom jogador para essa posição, saber quem são os melhores é interessante. Para um apostador, apostar em first kills é uma boa pedida, então, tendo a informação sobre jogadores que se destacam pegando first kills faz com que um apostador tenha mais chance de ganhar apostas do tipo. Já para um jogador que joga buscando opening kills, saber quem são os melhores jogadores que jogam dessa forma já dá um bom direcionamento para o material de estudo desse jogador.

```
### Filtrando pelo ano de 2018, criando coluna com a soma e uma coluna da média
topfkdiff <- players %>%
  filter(year(date) == 2018) %>%
  group_by(player_name) %>%
  summarise(media_fkdiff = mean(fkdiff), soma_fkdiff = sum(fkdiff))

### Adicionando uma coluna com o número de partidas jogadas
npartidas <- players %>% filter(year(date) == 2018) %>% count(player_name)
topfkdiff <- topfkdiff %>% inner_join(npartidas, by = "player_name")

### Dando uma olhada
topfkdiff
```

```
## # A tibble: 5,413 x 4
##    player_name media_fkdiff soma_fkdiff     n
##    <chr>              <dbl>       <int> <int>
##  1 "-AtomiK"         -0.5            -1     2
##  2 "-ian"             2               2     1
##  3 "-stew-"          -0.2            -1     5
##  4 "-will"            1               1     1
##  5 ".K9 "             0               0     4
##  6 "$h0cK3r"          2               2     1
##  7 "0°c"              2               4     2
##  8 "0i"               0.172           5    29
```

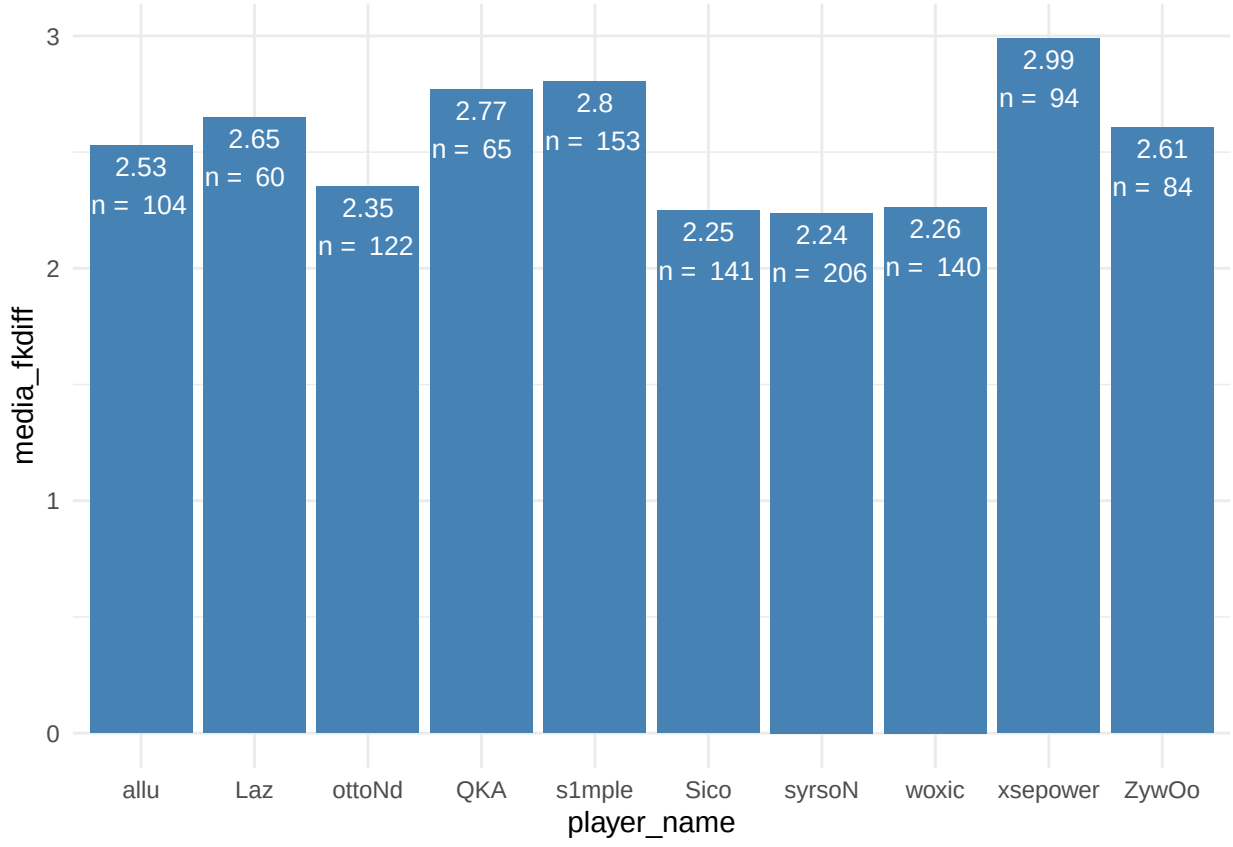
```
##  9 "0ng0wa"          -0.667          -2     3
## 10 "1000000"         -0.5            -2     4
## # ... with 5,403 more rows
```

```r
colnames(topfkdiff)[4] = "n_partidas"

### Escolhendo somente jogadores com certo número de partidas
### para que jogadores com poucas partidas não entrem no top
top50 <- filter(topfkdiff, topfkdiff$n_partidas>= 50)

### Pegando apenas os 10 melhores
topfk_media <- head(arrange(top50,desc(top50$media_fkdiff)),10)
```

Temos os seguintes significados e variáveis:

- **fkdiff**: È a diferença do número de first kill que o jogador fez, pelo número de first kills que o time adversário pegou no jogador.
- **media_fkdiff**: Média do fkdiff de um jogador.
- **player_name**: Nome do jogador.

```r
### Plot 1 - fkmedia
ggplot(data=topfk_media, aes(x=player_name, y=media_fkdiff)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=round(media_fkdiff,2)), vjust=1.6, color="white", size=3.5)+
  geom_text(aes(label = "n = "), hjust = 1.1, vjust = 3.6, color = "white", size = 3.5)+
  geom_text(aes(label = n_partidas), hjust = -0.03, vjust = 3.6, color = "white", size = 3.5)+
  theme_minimal()
```



o n, abaixo da média de cada jogador, é o número de partidas que esse jogador jogou.

Mapa CT ou TR?

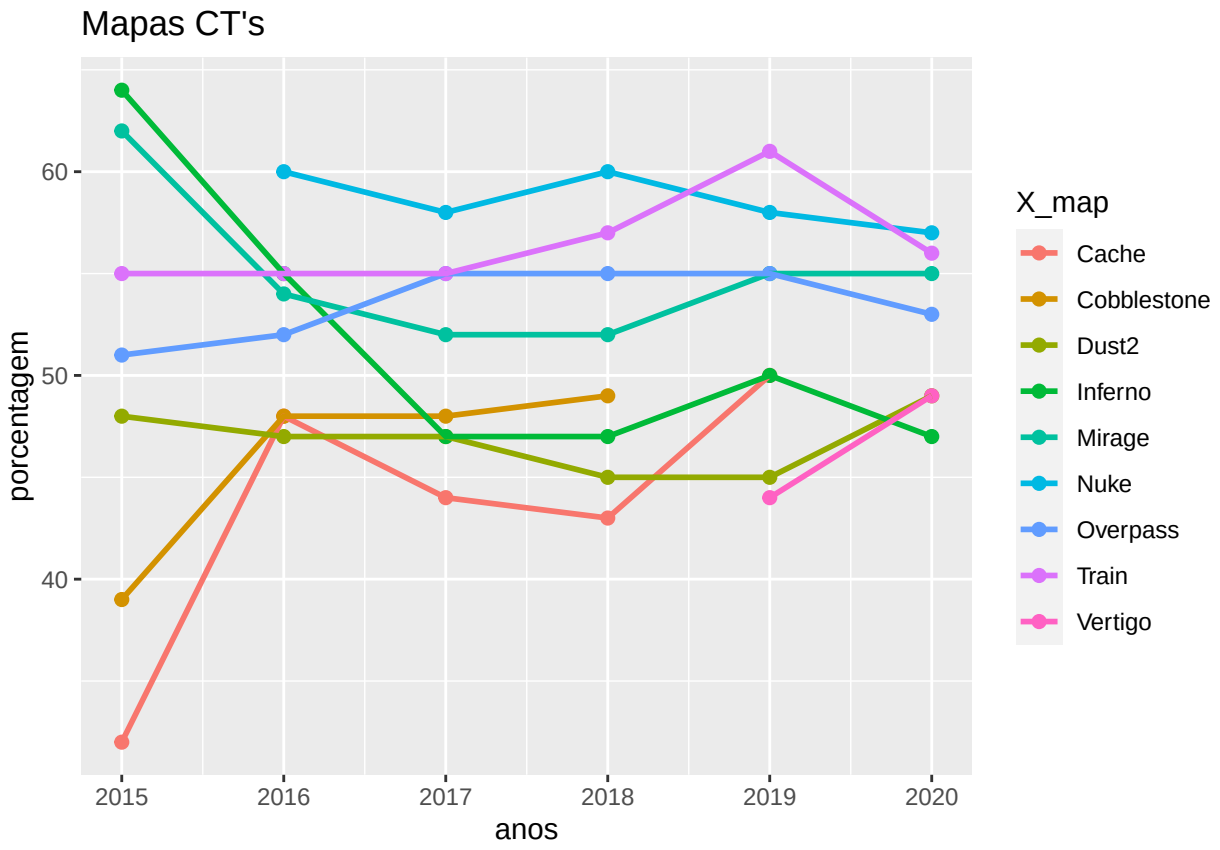
Uma discussão bastante recorrente no cenário do CS:GO, é a dúvida sobre qual mapa é mais CT-sided ou T-sided. Ou seja, se o mapa em questão é mais fácil de atacar para os terroristas, ou se é mais fácil defender para os contra-terroristas. Essa questão é bastante essencial no cenário competitivo, já que ao ter o conhecimento sobre isso, o time que tem a opção de escolher o mapa irá na maioria da vezes optar por escolher o lado mais fácil, para além de começar pontuando, fazer o time ganhar mais confiança.

Assim como primeira análise propusermos a identificar quais mapa são mais CT's a partir da tabela results.csv, que contém os resultados de cada lado das partidas incluindo quem começou de ct. Vale ressaltar, que como a Valve (desenvolvedora do jogo) está constantemente atualizando o jogo, assim o deixando mais dinâmico, então a tendência é que pelo menos a cada ano os mapas também tendem a mudar o lado mais fácil de se jogar, portanto realizamos uma análise anual.

```
### Adicionando o ano em que as partidas foram realizadas no data frame results
results$year <- with(results, year(date))

### Variável que contém a soma dos jogos em cada mapa por ano
total_per_map_year <- results %>%
  group_by(X_map) %>%
  count(year)

attach(results)

### Adicionando uma nova coluna binária, onde 1 representa se o lado ct ganhou e
### 0 se perdeu.
for(i in 1:nrow(results)) {
  if (starting_ct[i] == 2) {
    if (ct_2[i] > t_1[i]) {
      results$ct_winner[i] <- 1
    } else {
      results$ct_winner[i] <- 0
    }
  } else {
    if (ct_1[i] > t_2[i]) {
      results$ct_winner[i] <- 1
    } else {
      results$ct_winner[i] <- 0
    }
  }
}

### Novo dataframe realizado a partir do agrupamento dos maps e anos, com a
### quantidade de vitórias do lado ct.
maps <- results %>%
  group_by(X_map, year) %>%
  summarize(total_ct_winner = sum(ct_winner, na.rm = TRUE))

### Adicionando a coluna de uma aproximação da porcentagem vitória do lado ct.
for(i in 1:nrow(total_per_map_year)) {
  maps$porcentagem[i] <- as.integer(maps[i, 3]*100/total_per_map_year[i, 3])
}
```

```
### Removendo as linhas em que o Mapa escolhido era dado como "Default"
maps <- maps[-c(10:14), ]
```

Com as modificações realizadas obtemos o seguinte gráfico



Times com os Melhores Pistols {#plotly}

No CS:GO, a sequência de abertura de cada jogo e lado de mapa é o que é conhecido como pistol round ou pistol. É bastante importante a vitória já que além do ponto ele determina qual time terá um começo com vantagem econômica.

Embora os pistols não sejam o fator decisivo em séries individuais, as equipes podem mostrar uma quantidade moderada de melhorias em sua estratégia e habilidade para melhorar seu sucesso nessa fase.

Assim é de bastante interesse para os times conseguirem saber o quão bem os pitols adversários estão, além de ser uma informação bastante útil para possíveis apostadores, mesmo que ganhar pistols de forma consistente possa ser uma tarefa bem difícil.

```
### Dataframe com o nº do vencedor do pistol e o nº do time que venceu.
df <- economy %>%
  gather(key="number_team", value="team", team_1, team_2) %>%
  transmute(team = team,
            year=year(date),
            X1_winner = as.numeric(X1_winner),
            number_team=as.numeric(gsub('team_','',number_team))) %>%
  filter(year!=2020)

attach(df)
```

```
### Adicionando uma nova coluna binária, onde 1 representa se o time ganhou o
### pistol e 0 se perdeu.
for(i in 1:nrow(df)) {
  if (X1_winner[i] == number_team[i]) {
    df$result[i] <- 1
  } else {
    df$result[i] <- 0
  }
}

### Dataframe com a quantidade de jogos e sua taxa de vitória por ano e time
df <- df %>%
  group_by(team, year) %>%
  summarize(media = mean(result),
            n = n())

### Dataframe com a quantidade de nacionalidade dos jogadores por time e ano.
country <- players %>%
  transmute(team = team,
            year=year(date),
            country=country) %>%
  group_by(team, year, country) %>%
  summarise(n = n())

### Dataframe para encontrar a nacionalidade do time.
group <- as.data.table(country)
group <- group[group[, .I[which.max(n)], by=team]$V1] %>%
  select(team, country)

custom = c("Argentina" = "South America",
           "Brazil" = "South America",
           "Furious" = "South America",
           "Colombia" = "South America",
           "United States" = "North America",
           "Canada" = "North America")

### Realizando a coluna dos países ao dataframe df e adicionando uma coluna do
### continente do país
df <- df[df$n > 50,] %>%
  merge(group, by="team",all.x=TRUE) %>%
  mutate(continent = countrycode(sourcevar = country,
                                 origin = "country.name",
                                 destination = "continent",
                                 custom_match = custom))
```

Mapas mais escolhidos em partidas MD1

Nas partidas melhores de 1 (MD1), as equipes devem vetar os mapas de maneira alternada até que reste um mapa a ser jogado, onde o lado de ínicio é definido por um round faca.

Ter o conhecimento sobre os mapas mais escolhidos é bastante essencial para qualquer time do competitivo, já que possivelmente você deverá treiná-los ou então veta-los. Sabendo disso, decidimos fazer uma análise para descobrir quais são os mapas mais jogados em cada ano.

```r
### Adicionando uma coluna com o ano dos jogos e excluindo as outras
### variáveis, com excessão da best_of e o left_over.
picks <- picks %>%
  transmute(year = year(date),
            best_of = best_of,
            left_over = left_over)

### Vetor com os mapas do jogo
map <- c("Cache","Cobblestone","Dust2","Inferno","Mirage","Nuke","Overpass", "Train", "Vertigo")

### Dataframe com apenas as partidas md1, agrupadas pelo ano e mapa escolhido
picks_md1 <- picks %>%
  filter(best_of == 1,
         left_over %in% map) %>%
  group_by(year, left_over) %>%
  summarise(n = n())
```

Referências

[1] Mercado de eSports: faturamento, audiência e o cenário no Brasil

[2] Top 10 organizações mais valiosas no eSports - 13/04/2021

[3] Resultados - Faze Clan (2016-2022)

[4] Documentação do pacote Plotly

[5] Pacote countrycode