"Rozgrywki w popularnej grze typu auto chess: Teamfight Tactics”

Członkowie projektu

- Borek Łukasz
- Domaradzki Stefan
- Dziedzic Paweł

Źródło danych

https://www.kaggle.com/datasets/gyejr95/tft-match-data

Wprowadzenie

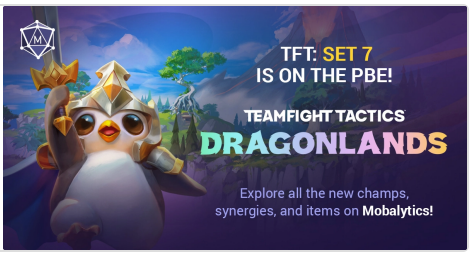
Teamfight Tactics to wywodząca się z uniwersum *”League of Legends”* gra z gatunku auto chess, w której 8 graczy rywalizuje poprzez odpowiednie dobieranie w każdej turze postaci (analogią do szachów są figury) które następnie zależnie od swojego rodzaju oraz miejsca na które zostały ustawione walczą ze sobą. Główną ingerencją gracza jest możliwość ich przestawiania lub (po zebraniu 3 kopii danej figury “promowania” na wyższy poziom). Figury zdobywa się poprzez wykupienie za zdobyte w pojedynczych partiach złoto celem uformowania możliwie najsilniejszej kombinacji. Każdy z 8 graczy zaczyna z liczbą zdrowia wynoszącą 100. Ostatni gracz który zachowa zdrowie wygrywa.

Z racji, że gra zmienia się sezonowo i pewne mechaniki mogą być nieaktualne należy zaznaczyć, że dane z których korzystamy odnoszą się do 3. sezonu noszącego nazwę: *”Galaxies”,* kompleksowe informacje na temat statystyk poszczególnych “figur” można znaleźć np. na stronie:

TFT Set 3 Champions: All Teamfight Tactics Champs

TFT Set 3 champions. Find out all the info, recommended items, existing class and origin synergies and much more for TFT Set 3.

https://app.mobalytics.gg/tft/set3/champions



Przykladowa plansza we wczesnej fazie rozgrywki

Sklep z którego dobierane są “figury”

Organizacja pracy

W celu płynnego działania grupy nad kodem i plikami wchodzącymi w skład projektu, grupa aktywnie korzystała z popularnego serwisu hostującego repozytoria: GitHub. Poszczególne elementy pracy, zadania które muszą być wykonane zostały rozdzielone według umiejętności i preferencji członków zespołu, co usprawniło pracę i oszczędziło konfliktów.

Struktura folderu projektu, przed wykonaniem wykresów (06..06.2022)

Import danych

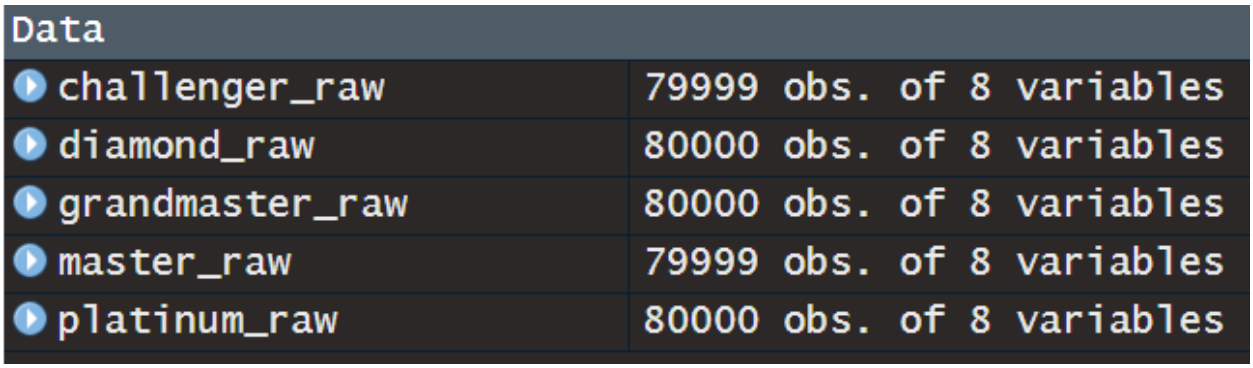
Projekt składa się z pięciu ramek danych, każda odpowiada jednej z rang określając poziom na którym znajdowali się zawodnicy.

Kolejność rang (malejąco):

1. Pretendent (ang. Challenger)
2. Wielki Mistrz (ang. Grandmaster**)**
3. Mistrz (ang. Master)
4. Diament (ang. Diamond)
5. Platyna (ang. Platinum)

W skład ramek danych wchodzą:

- "gameId" - numer identyfukujący grę,
- "gameDuration" - czas trwania całej gry tj. do wyznaczenia zwycięzcy,
- "level" - poziom który osiągnął gracz, w dużym uproszczeniu oznacza on liczbę zebranych figur,
- "lastRound" - liczba rund które zawodnik ukończył,
- "Ranked" - pozycja w rankingu w momencie ukończenia gry
- "ingameDuration" - czas jaki indywidualny gracz spędził w grze
- "combination" - w dużym uproszczeniu oznacza strategię obraną przez gracza (jest to liczba bierek które dobrze ze sobą współgrają),
- "champion" - lista figur uzbieranych w trakcie rozgrywki.



Ramki danych bezpośrednio po zaimportowaniu

Oczyszczenie danych

Ponieważ dane zostały wygenerowane na podstawie historii gier w większości nie posiadają w pewnym rozumieniu braków danych. W przypadku niepołączenia się gracza, lub jego przedwczesnej rezygnacji, jest on uznany za AFK (ang. away from keyboard) i mimo braku informacji o statystykach rozgrywki takiego gracza, jest to ciekawe zjawisko które również może zostać poddane analizie.

Jednakże w trakcie pracy nad projektem (ze względu na rozmiar plików nie udało nam się ręcznie trafić na tę wiersze, dopiero błąd programu jednego z algorytmów je wskazał) odnaleźliśmy bardzo ciekawą anomalię.

gameId	gameDuration	level	lastRound	Ranked	ingameDuration	combination
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}
KR_4229542485	127.6975	3	4	0	127.6975	{}

Odkryta w trakcie pracy programu “anomalia”

W grze o numerze identyfikacyjnym “KR_4229542485” wszyscy gracze zostali rozłączeni w tym samym czasie, bez możliwości wyboru nawet po jednej postaci (pusta kolumna *combination*), oraz wszyscy ukończyli rozgrywkę z wynikiem końcowym równym 0.

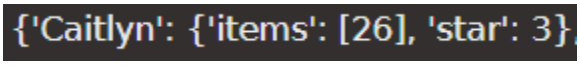
Niestety niemożliwe jest odszukanie tej rozgrywki, lecz najprawdopodobniej doszło do awarii serwera, który rozłączył wszystkich. Po odnalezieniu tego ewenementu postanowiliśmy się jeszcze raz przyjrzeć naszym ramkom danych i stworzyć fragment skryptu który pozbędzie się tej *“anomalii”*, i wszelkim jej podobnych, oraz zapobiegnie błędom programu w trakcie jego czasu pracy.

Statystyka poziomy postaci

Skompletowanie trzech kopii danej figury powoduje jej promocję na wyższy poziom (max. 3). Jednak im silniesza dana figura tym trudniej zebrać jej komplet. Gracze stają więc przed wyborem, kompletować słabe bierki i je promować czy wstrzymać się z

wydatkiem złota (ceny figur różnią się zależnie od siły) i spróbować zdobyć silniejszą postać która nawet bez promocji może znacząco wpłynąć na przebieg rozgrywki.

W celu wyznaczenia tych parametrów tworzymy kolumnę do której będą wrzucane wartości liczbowe oznaczające poziomy postaci. Ze względu na strukturę wyjściowej ramki (poza informacją o poziomie kolumna zawiera informacje o przedmiotach) operację wybierania wyłącznie wartości poziomów:



Jedna z wartości zawierająca informację o figurach gracza

By uniknąć wybierania wartości liczby odnoszącej się do przedmiotów postaci zastosowujemy funkcję wyrażeń regularnych o następujących argumentach:

```
stri_extract_all_regex(platinum_champion_stars$champion,    ".star...")
stri_extract_all_regex(platinum_champion_stars$star_list,    "[0-9]")
```

Następnie funkcją *aggregate* wyznaczamy średnią dla każdego z dostępnych poziomów postaci (tj. od 1 do 3) względem osiągniętego wyniku końcowego, całość powtarzamy dla wszystkich 5 możliwych rang.

Statystyka zwycięstwa wg. poziomu

W miarę postępu gry zawodnicy zdobywają poziomy co bezpośrednio przekłada się na maksymalną możliwą liczbę figur na planszy danego zawodnika. Statystyka zwycięstw wg. poziomu bezpośrednio wskazuję jak liczba dostępnych figur przekładała się na zwycięstwa względem umiejętności zawodnika (przekrój platyna - pretendent)

```r
F_level_ranked <- function(rank_level_ranked){
rank_level_ranked_list <- list(1,2,3,4)
for (i in 1:4) {
  for (j in 6:9) {
    rank_level_ranked_list[[i]][j-5] <- length(rank_level_ranked$level[rank_level_ranked$level == j
                                    & rank_level_ranked$Ranked == i])
  }
}
rank_level_ranked_frame <- as.data.frame(rank_level_ranked_list,row.names = c("Szósty","Siódmy","Ósmy","Dziewiąty")
                                         ,col.names = (c("Pierwsze","Drugie","Trzecie","Czwarte")))
return(rank_level_ranked_frame)
}
```

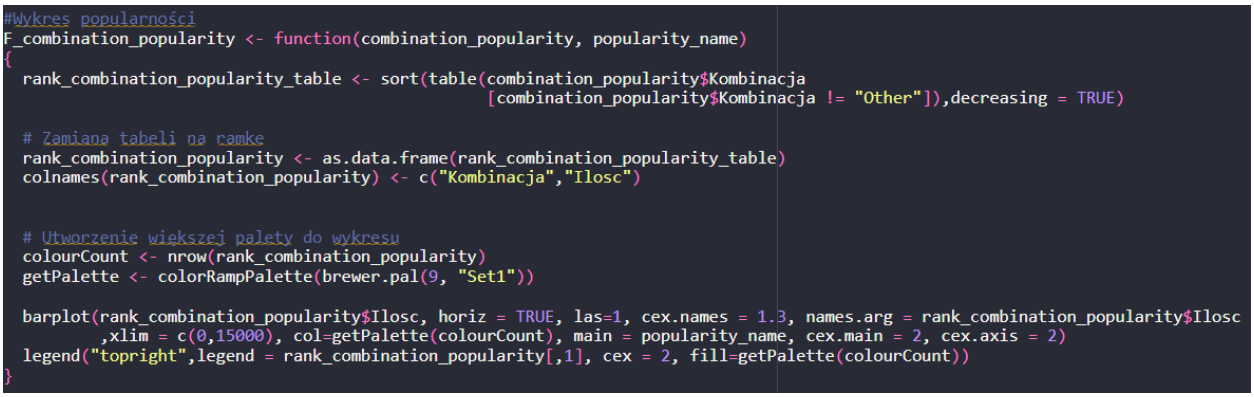
Wybór zwycięstw na podstawie osiągniętego poziomu

Statystyki kombinacji

Wszystkie figury mają przypisane pewne charakteryzujące je aspekty, w najlepszym uproszczeniu, figury o takich samych cechach będą ze sobą współgrały, można powiedzieć “tworzyły formację” która jest znacząco bardziej efektywna niż, pojedyncze postaci o różnych cechach. Dla przykładu 3 postacie o dalekim zasięgu będą stawiały większy opór niż dwie o dalekim zasięgu i jedna o średnim. Oczywistym jest również, że pewne strategie, pojedyncze kombinacje lub ich wymieszania będą bardziej skuteczne od innych co będzie się przekładało na ich odsetek zwycięstw. Obie statystyki są wizualizowane w trakcie pracy skryptu. (niestety nie zmieściły się wszystkie na plansze).

```r
for (i in 1:nrow(rank_combination)) {
  for(j in 1:(length(rank_combination)-3)){
    #Blasters
    if(rank_combination[i,1] >= 5){
      rank_combination_all[i,] <- "Blaster"
      break
    }
    #Blademaster + celestial
    else if(rank_combination[i,12] >= 3 && rank_combination[i,4] >= 4){
      rank_combination_all[i,] <- "Blademaster_Celestial"
      break
    }
    #infiltrators
    else if(rank_combination[i,15] >= 6){
      rank_combination_all[i,] <- "Infiltrators"
      break
    }
    #Chrono
    else if(rank_combination[i,2] >= 6){
      rank_combination_all[i,] <- "Chrono"
      break
    }
```

Część skryptu odpowiedzialnego za wyszukiwanie i zliczanie kombinacji zebranych przez graczy



```r
#Wykres popularności
F_combination_popularity <- function(combination_popularity, popularity_name)
{
  rank_combination_popularity_table <- sort(table(combination_popularity$Kombinacja
                                                  [combination_popularity$Kombinacja != "Other"]),decreasing = TRUE)

  # Zamiana tabeli na ramke
  rank_combination_popularity <- as.data.frame(rank_combination_popularity_table)
  colnames(rank_combination_popularity) <- c("Kombinacja","Ilosc")


  # Utworzenie większej palety do wykresu
  colourCount <- nrow(rank_combination_popularity)
  getPalette <- colorRampPalette(brewer.pal(9, "Set1"))

  barplot(rank_combination_popularity$Ilosc, horiz = TRUE, las=1, cex.names = 1.3, names.arg = rank_combination_popularity$Ilosc
          ,xlim = c(0,15000), col=getPalette(colourCount), main = popularity_name, cex.main = 2, cex.axis = 2)
  legend("topright",legend = rank_combination_popularity[,1], cex = 2, fill=getPalette(colourCount))
}
```

Skrypt odpowiedzialny za wykresy kombinacji

Statystyka odsetek AFK

W przypadku braku aktywności gracza w meczu zostaje on uznany zgodnie z żargonem gry AFK (ang. away from keyboard). AFK jest uznawane za niesportowe zachowanie a gracze opuszczający rozgrywki są pozbawiani nagród a w szczególnych przypadkach możliwa jest również surowsza kara. Zjawisko jest jednak nieuniknione, wiadomo gracze są zwykłymi ludźmi i różne rzeczy mogą się wydarzyć, jednak im wyższa ranga zawodnika tym bardziej jest skłonny do poświęceń i stara się unikać opuszczania rozgrywek. Odsetek graczy AFK jest realizowany prostą funkcją, i wyznacza wartość względem wszystkich pięciu dostępnym do analizy rangom.

Statystyka popularność postaci

Pewne postaci ze względy na swoje unikalne cechy i zdolności bojowe są znacznie potężniejsze od innych. Ze względu na to pewne figury są częściej wybierane, a informacje o tym które postaci są obecnie najpopularniejsze może bezpośrednio wpłynąć na liczbę zwycięstw odnoszonych przez gracza posiadającego takie statystyki.

Wnioski

Praca nad projektem grupowym rozwinęła nasze zdolności programowania w języku R. Wspólnymi siłami staraliśmy się rozwiązywać problemy programistyczne napotkane na drodze do ukończenia projektu. Znacząco poprawiły się również nasze umiejętności pracy w zespole oraz zarządzania czasem, a także rozwinęliśmy umiejętności

współpracy ze środowiskiem GIT które jest obecnie obowiązkowym narzędziem każdego szanującego się programisty.