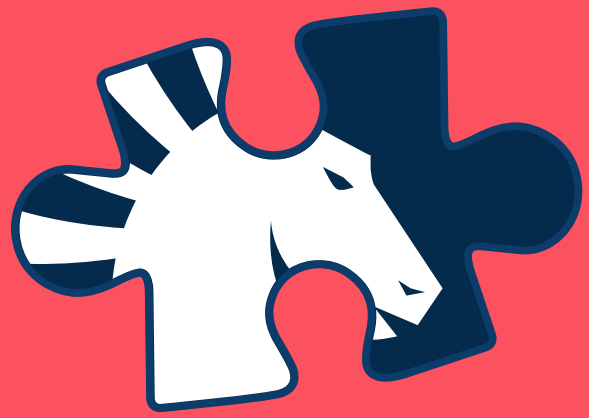
Data aggregation and rating calculations for VCT

Joshua Kim

Project Scope

- Intake data and organize into a readable format
- Allow users to see player stats per over various periods
- Formulate new rating system for players

Existing solutions



VCT (Valorant Champions tour)

- Esports league for Valorant
- Split into 4 regional league and 2/3 levels of competition
- Each game is a best of 3 maps

Americas

Pacific

EMEA

China

Dataset

- Dataset was obtained from Ryan Luong on Kaggle [1]
  - Scraped from VLR.gg (data from official api)
- 25 Total data categories across 11250 data points
- Specific dataset used is 2023 player stats
- Contains data from every official match played in 2023
  - Each row represents a different match/map

Important notes

- Some data points are blank due to China's different system
- No individual map data, this is in a different data set
- Player names are used for identification rather than id's

Data structures

Master_Stat	Player Name -> string Teams -> vector<string> Tournaments played -> vector<tournaments>
Tournaments	Tournament name -> string Tournaments stages -> vector<stages>
Stages	Stage name -> string Stage matches-> vector<mtypes>
Mtypes	Stage name -> string Stage matches-> vector<stats>
Stats	Various stats (21 total)

Data intake functions

File Intake, O(n)

Input: Vector to store raw file data
Output: 0(error) or 1(success)

Open file
Check for successful file opening, return 0 for failure
For each row in the file
    Split row by commas and store in temporary vector
    While values in vector position 6 are not numbers
        Combine them into vector position 5
    Erase the combined values
    Push this split row into raw data vector
Close file
Erase first row of data -> these are the data headings
Return 1

File trim, O(n log n)

Input: Vector of raw data, vector for player data

Sorts raw data alphabetically by player names -> nlogn (heap sort)
For each element in the raw data vector
    Checks for duplicate player in player data
    Checks for duplicate tournaments in player data
    Checks for duplicate stages in player data
    Checks for duplicate matches in player data
        Checks if duplicate match has more rounds played
    Assigns necessary data to a temporary vector
    Push the temp vector to the player data vector

Search and UI functions

Find_tournament O(n)

Input: master_strat for the respective player
Output : tournaments struct for the selected value

Print tournaments the player has played
Call function num_check() to ask user for their selection
Return tournaments struct for the selected tournament

Find_stage O(n)

Input: tournaments struct for the respective tournament
Output : stage struct for the stage

Print stages of the tournament
Call function num_check() to ask user for their selection
Return stage struct for the selected stage

Find_match O(n)

Input: tournaments struct for the respective player
Output : mtypes struct for the match

Print matches of the stage
Call function num_check() to ask user for their selection
Return mtypes struct for the selected match

Find_player O(n)

Input: sorted player data vector
Output : master_stat struct for the player

Ask user what letter/number the players name starts with
Print players who start with that letter
Call function num_check() to ask user for their selection
Return master_stat struct for the selected match

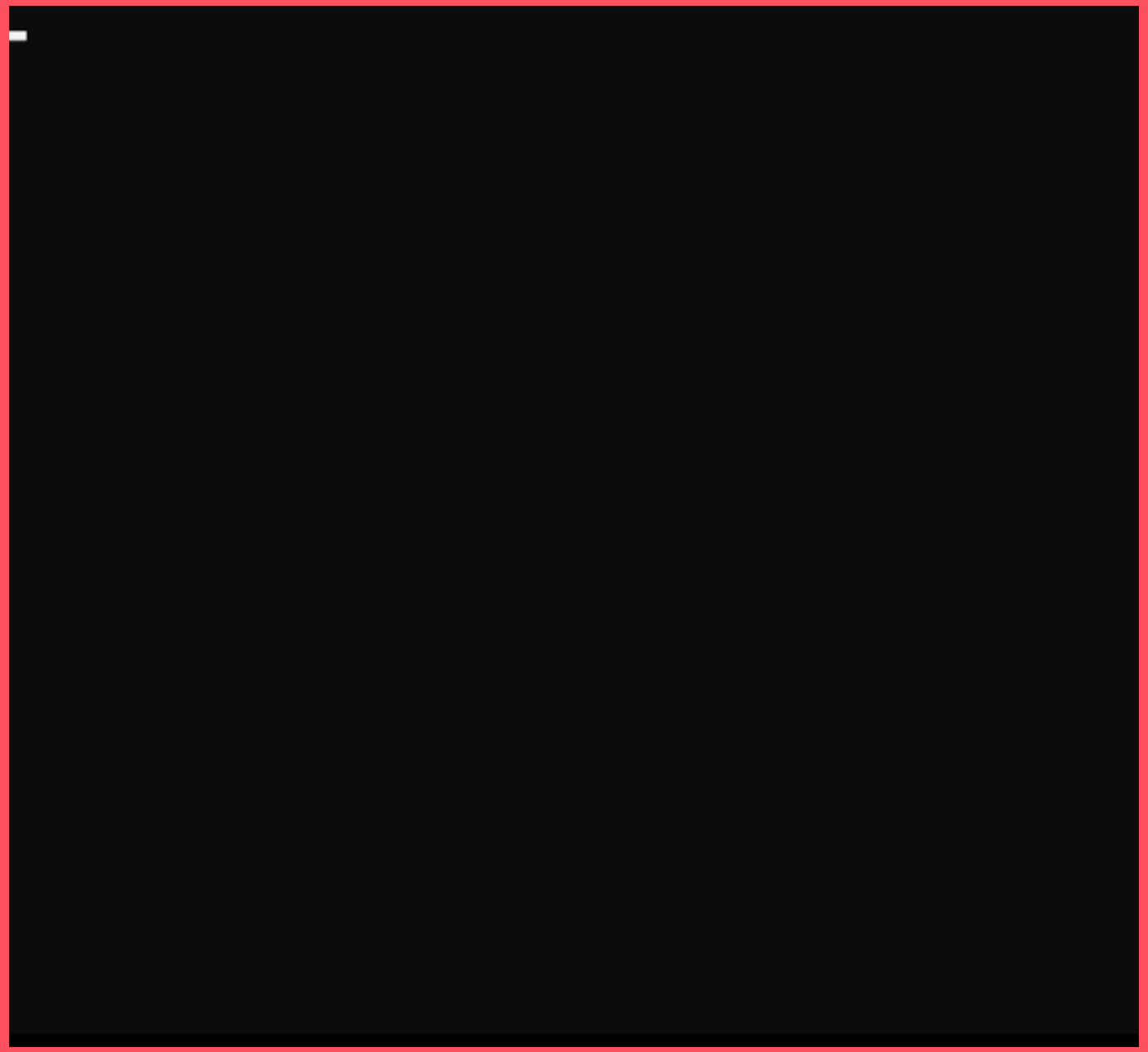
Other important functions

- Assign_stats
  - Input: vector of raw data, pos of the player in the vector
  - Output: struct stat
  - Iterates through the row of the player and assigns the stats as needed
  - Time complexity O(n)
- Assign_tournaments/Assign_stage/Assign_mtype
  - Input: vector of raw data, pos of the player in the vector
  - Output: struct tournament/struct stage/struct tournament
  - Assigns elements of the respective vector from the raw data
  - Time complexity O(n)
- Num_check
  - Input: Range of selection
  - Output: number selected by user
  - Asks the user for a number and uses recursion and type errors to check that it is valid
  - Time complexity O(1)

Rating Algorithm

$$\left(\frac{acs}{100} * \frac{adr}{100} - fdpr + fkpr\right) * \frac{kast}{100} * \left(\frac{kpr + apr}{2}\right)$$

- ACS: Average combat store, in game rating calculated by kills, deaths, ect
- ADR: Average damage per round
- Fdpr: first kills per round
- Fdpr: first deaths per round
- KAST: kill, assist, survival, trade percentage
- Kpr: Kills per round
- Apr: Assists per round



Challenges

- No available public API data
- Program is dependent on specific data formatting
- Program is horribly inefficient because of the large amount of data
- Data structure prioritizes player searching as the top layer

Next steps

- Clean up code and formatting
- Improve efficiency and optimize code further
- Add more searching functionality
  - Best players from each tournament, stage, match etc.
- Add support for different data sets
- Rework data structure and functions for further flexibility in data processing

Refrences

[1]: https://www.kaggle.com/datasets/ryanluong1/valorant-champion-tour-2021-2023-data?resource=download