# Project Summary

This project is motivated by my experience in a popular online game League of Legends. The game provides some descriptive stats at the end of each game but those stats didn't give me any directional guidance on how I can improve my chances of winning. Motivated by this, the project aims to provide both descriptive and predictive stats, organise them into a storyline, and interactively visualise them.

# Game Overview

League of Legends is a multi-players arena battle game. There are two teams in a game, each having five players. The purpose of the game is to destroy the enemy's Nexus. There are three lanes to get to the enemy. In each lane, there are several defence towers and minions.

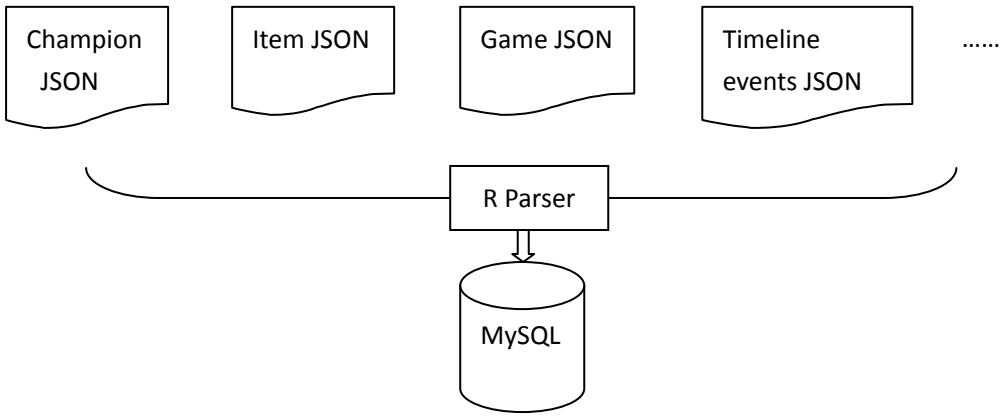
Each player controls a champion which has a set of different attributes, such as magic damage, physical damage, moving speed, armour, etc. So each champion has its own pros and cons, e.g., some champions can do a great deal of magic damage, minimum physical damage and are easy to get killed, while some others have high levels of armour, magic resistance and health regeneration but can barely do any damage. Therefore, a good team combination is important. These attributes can be enhanced from buying items using the gold earned in a game from killing enemies and minions, destroying enemy's defence towers, assisting teammates, etc.


Figure 1. Bird View of the Arena

# Data Parsing

All the data are originally in JSON format and are parsed into tables in MySQL using parsers written in R. The database contains tables such as

- Item table, including attributes of each item
- Champion table, including attributes of each champion
- Game table, including high-level measures of a game, e.g., game mode, game duration, gold earned at the end of a game, etc.
- Timeline events table, including events and their time stamps of each participant in a game. The types of events can be killing an enemy, purchasing/selling an item, assisting a teammate, killed by an enemy, etc.




**Corresponding Github scripts:**

- 1_DataParsing_01-jsonDdragon.R
- 1_DataParsing_04-GameTimeline.R

# Data Processing

The goal is to create a table that can be used to explore and analyse the relationships between variables and win rate. Which table to create depends on the level of the analysis required. I am conducting analyses of performance at the following three levels**:**

- **Team level** – viewing a team in a game as the unit of analysis
- **Champion level** – viewing a champion in a game as the unit of analysis
- **Player x champion level**. Because even with the same champion, people can play it differently or build different based on their skills and preference, e.g., one who plays a champion aggressively may build more offensive items while a passive player may be after a more balanced build. So this level of analysis considers the interaction of a type of player and a champion as the unit of analysis

Within each level, two types of questions I intend to answer:

- Compared with the average, is the performance of a team/champion/player x champion above or below average performance? What are the causes of losing? This is motivated by a problem in the community that players tend to overestimate their performance and blame others for losing a game so sometimes the gaming experience can be quite negative.
- Further break down the performance by time interval and compare the performance of a team/champion/player x champion to the average at each interval. What is my winning strategy at each point of time in a game?

Starting from the easiest, I created a table of the team level analysis after a series of joining and aggregating:

1. Create derived variables from timeline events table. E.g., rather than keeping information on who assisted in killing an enemy, the table will have information on how many teammates assisted a kill, an indicator of teamwork.

2. Join the item table to the timeline events table. So rather than recording the time stamp a participant bought item A, the table will have a time stamp that a participant increased these attributes through purchasing an item.

3. Aggregate timeline events table to one minute one participant per record. A participant can buy/sell multiple items, or kill multiple enemies in one minute.

4. Join timeline table with timeline events table. The timeline table contains information on the amount of gold earned, the number of minion killed, etc. at every minute.

5. Aggregate the table to one participant per record by adding additional timeline variables of every 3-min interval. E.g., the TotalGold from the timeline table will become TotalGold_4, TotalGold_7, TotalGold_10 till TotalGold_31. Each interval is the sum of the values of last 3 minutes. The decision of total 31 minutes is based on the average duration of a game, and the decision of number of intervals is due to the total size of a table allowed in MySQL.

6. Attach champion table and rune table.

7. Aggregate to one game per row

**Corresponding Github scripts:**

- 2_DataPrep-02-modellingTable.R: batching the 03-participant_rollup sql scripts by timeline date
- 2_DataPrep_03-2-participant_rollup.sql: create derived variables from timelineevents table
- 2_DataPrep_03-3-participant_rollup.sql: decompose string in column assistingParticipantIds
- 2_DataPrep_03-4-participant_rollup.sql: aggregated columns that don't have participantId by gameId and frames
- 2_DataPrep_03-5-participant_rollup.sql: grab the complete version for timelineevents table

- 2_DataPrep_03-7-participant_rollup.sql: join the item stats to timelineevents table
- 2_DataPrep_03-8-participant_rollup.sql: aggregate timelineevent to one frame/min per record
- 2_DataPrep_03-9-participant_rollup.sql: attach Num_AssistBld
- 2_DataPrep_03-10-participant_rollup.sql: attach Num_AssistChamp
- 2_DataPrep_03-11-participant_rollup.sql: attach Num_Deaths
- 2_DataPrep_03-12-participant_rollup.sql: attach Num_MonsterKilled, Num_Ward_Killed, Num_Ward_Place
- 2_DataPrep_03-13-participant_rollup.sql: join events to timeline
- 2_DataPrep_03-14-participant_rollup.sql: add rank column to the table
- 2_DataPrep_04-transParticipant.R: transpose aggregated attributes of each min to 3min interval
- 2_DataPrep_04-participant_champion.sql: read in by 05-attachChamp_Rune.R. Attach champions table to participant_rollup
- 2_DataPrep_05-participant_champion_rune.sql: read in by 05-attachChamp_Rune.R. Attach runes to participant_champion
- 2_DataPrep_05_attachChamp_rune.R: attach champion stats & rune stats. Accumulate runes stats from runeId1 to runeId10, on top of item stats

# Data Modelling

The analysis that has been performed so far is at the game level.

**Descriptive Analysis**

The goal here is to compare the performance of a team/champion/player x champion with the average performance, and drill down to find the under-performed areas. I use the win rate (i.e., proportion of wins) to measure the performance. For each measure, because an end-game measure is a function of the game duration, I first calculate the measure per minute, discretise it into 50 bins with equal volume and take the mean, calculate the proportion of wins in each bin, and chart the average as the x-axis and the win rate as the y-axis.

**Predictive Analysis**

The descriptive analysis can give a pretty good idea on how well a variable correlates with the win rate. To be able further quantify the predictivity of variables to the win rate, I built models using information at each time interval to predict the final win and lose. The reasons why I chose to build models based on variables at each interval rather than all variables I have are:

- The same variables at different intervals are highly correlated with each other. For example, if I earn more gold in a previous interval, I would be able to buy more items or more powerful items in the next interval, which would increase my chance to kill and earn more gold. So the final model will be dominated by variables from later intervals and fail to reveal the importance change of a variable during the whole course of a game. This can be crucial information to formulate in-game strategies.

- If I can identify at which interval I can obtain the best prediction, it can save me time by only using a subset of variables rather than all variables. Particularly, the game allows players to surrender after the 20th minute and sometimes players forfeit a game even if they can win in the end. It would be beneficial to provide some guidance to help players make an informed decision.

The modelling process is as below

- At each time interval,
  - ✓ Use variables from that interval and prepare data for modelling
  - ✓ Cross validate using xgboost models on the training set to find the best learning rate
  - ✓ Use the best learning rate to build an xgboost model and score on the testing set for performance evaluation
  - ✓ Remove unimportant variables based on the variable importance list. The remaining variables will be used for building glmnet models
  - ✓ Apply polynomial and 10-bin discretisation transforms to the variables
  - ✓ Cross validate using glmnet models on the training set to find the best alpha for regularisation
  - ✓ Use the best alpha to build an glmnet model and score on the testing set for performance evaluation
  - ✓ Store the model performance measures and variable importance & coefficients
- Repeat the process for the next time interval

**Corresponding Github scripts:**

- 3_Analysis_02-1_team_descriptiveChart.R
- 3_Analysis_02-3_teamSeriesChart.R
- 3_Analysis_02-6_team_predByMin.R

# Visualisation

The visualisation is rendered using D3 (a Javascript library) in html. The interactive charts will be organised to tell a story. For example,

1. The descriptive gold earned per min chart shows the overall performance of my team and the enemy team against the average. If my team earns less than the average or the enemy team, I can drill down to areas that contribute to the gold earning.
2. The areas that contribute to the gold earning can be kills per min, death per min, minion killed per min, etc. By viewing those descriptive charts, I can have an idea which category that my team under-performed.
3. I know what areas my team needs to improve and now I need to formulate a strategy. I can view the variable importance chart which shoes me at what time in a game my team should focus on what categories. For example, my team should avoid dying as much as possible during the first 15 minutes.

**Corresponding Github scripts:**

- 4_Visualisation_01-team_goldEarned.html: use a high-level library based on D3, an interactive chart of the result from 3_Analysis_02-1_team_descriptiveChart.R
- 4_Visualisation_01-team_varImp.html (4_Visualisation_01-team_lineTransition.js): an interactive chart built from scratch using D3, showing result from 3_Analysis_02-6_team_predByMin.R

You can login below to access the visualisation URL hosted by my server:

**Username**: KPMG

**Password**: allowed

http://felicity.v1.pe/generated/chartJSON/team_goldEarned.html

http://felicity.v1.pe/generated/chartJSON/html/varImp.html