**Global Power Rankings write-up**

# ELO system

# when player 1 moves to team A from B, team A ELO becomes 1/5 \* team\_B + 4/5 \* team\_A

# could also add difference in team B ELO since player 1 joined the team (\* some constant)

# (this method will inflate ELO though if all players move to same, but new, team)

# also could use stats to decide how important player 1 was to team B, however it can be hard to compare players in different positions

# then set every team within a region (who didn't attend the tournament) to a specific value

# e.g., LPL = 1000

# this value can either be arbitrarily chosen depending on how strong I believe the region was at that time

# or could be, e.g., 50 ELO lower than the lowest ranked team from that region that attended the tournament

# could also rank the regional teams by the playoffs before tournament, but might be more effort or little change

# store every teams ELO throughout time (set to NaN if the team does not exist) for ELO difference to be calculated

# may need to instead store ELO at certain times (e.g., end of each month) if this is too much data

# can we run through all data in S3 bucket on AWS (i.e., without downloading it)?

# (this is only if we want the specific game data, not just knowing who wins/loses)

# may need to deal with teams changing name/getting bought out

# could also create an object for each team instead of a pandas series

# should only take the 5 highest elo\_contr from roster

# or maybe 5 most recent including the new player?

# this is only called when a new player joins the roster

# and plays a game so which method is more accurate?

# update elo on roster change

# KEEP FIRST 5 PLAYERS AS ACTIVE PLAYERS AND ONLY ALLOW

# THEM TO USE ELO FROM PREVIOUS TEAM

# assume only 5 players per team

# should also make old team inactive if

# not also the new team (as old team will

# now have < 5 players)

# need a way to track if a player is active or sub

# maybe the first 5 are always the active players

# update last\_game with date of game

# calculate elo change of a match

# will only be zero-sum if there are no additional conditions

# result(win) = 1, result(loss) = 0

# initial elo of new player

# OPTIONS:

# 1) find most recent international event

# find elo of lowest finisher from region

# set elo to const \* team\_elo / 5

# where const < 1

# with open("data/esports-data/leagues.json", "r", encoding='utf-8') as json\_leagues:

# leagues\_data = json.load(json\_leagues)

# # find region's main league

# priority = 999999

# for league in leagues\_data:

# if league['region'].lower() == region:

# # assume main league is that with the lowest priority

# if league['priority'] < priority:

# league\_id, priority = league['id'], league['priority']

# 2) average elo of all teams in ranking from the league

# need to be careful as if multiple teams are added at the same time

# the ones added last will have a lower elo

# 3) average elo of all teams at start/end of split/season and use

# throughout that split/season

# assume new player is worse than average

# def update\_region\_elo():

# # can either use most recent international tournament

# # or average region ELO \* const

# # calculating the final tournament standings using tournaments.json

# # is more work, so use average for now

# # world\_id = '98767975604431411'

# # msi\_id = '98767991325878492'

# # with open("data/esports-data/tournaments.json", "r", encoding='utf-8') as json\_file:

# # tournaments\_data = json.load(json\_file)

# # for tournament in tournaments\_data:

# # if (tournament['leagueId'] == world\_id) or (tournament['leagueId'] == msi\_id):

# # # loop through stages building up group tables and playoff bracket

# # break

# get chronological list of region major tournaments (i.e., TCL, PCS, LCO, LEC, LPL)

# check if tournament is a region major tournament

# # ignore international tournaments

# leagues\_dict = leagues\_dict.loc[leagues\_dict['region'] != 'international']

# can't use leagueId to differentiate playoffs from regular season

# maybe use the fact that in tournaments\_data[0]['stages] each section

# (regular season, playoffs) is named differently.

# Don't know if this is consistent though

# may need to make this high at international events to allow for

# rebalancing between regions (which only play themselves during season)

# should probably include a check here in case one of the teams is new (not in rankings)

# but also doesn't have a full roster in mapping\_data, i.e., the update\_team call

# above hasn't happened. this team will not have an elo in rankings (or even an entry)

# check order in tournaments.json matches mapping\_data.json

# check that every team in LEC plays the expected number of games during the regular season

# there are many 'unstarted' games at the end of the regular season

# how to determine which player is on which team?

# mapping\_data.json only shows players that were in the game

# then cross-reference this against all of the players on the team in tournaments.json

# can then see which players are on which team

# can either assume all player played all matches (may give strange results when there are subs used)

# or do each game individually

# maybe we assume team does not change from previous roster

# if game is not included in mapping\_data if < 10 players

# Nukeduck found in mapping\_data for Astralis, but not in

# tournament data. Which one is true?

# Astralis was actually Origen at the time.

# Nukeduck did play, meaning mapping\_data is correct.

# tournaments.json only shows current players (or at least

# is not guaranteed correct at the time of the match)

# therefore seems to be no way to know what players were on

# the team at the time of the match (without using lol.fandom API)

# maybe the correct players are contained within the game data?

# might be awkward to extract it, especially with how much data there is

# is there an easy way of quickly extracting just what we need?

# MAYBE A FIX

# the numbering of participants in mapping\_data seems to be

# 1-5: teamMapping=100, 6-10: teamMapping=200

# also seems to be in order top-jng-mid-bot-sup

# from tests we will assume this is true

# Games seem out of order in tournaments.json

# First game in LEC Spring 2020 in tournaments.json

# is Astralis vs SK which didn't happen until Week 5

# might have to treat each stage in one go

# this might cause odd behaviour with players that move teams

# mid split, but I don't think this is possible any more due to

# roster locks

# Doc says that mapping data for LPL is only for Summer 2023

# so this method of checking players won't work for many tournaments

# found = 0

# for team\_player in teams[0]['players']:

# if team\_player['id'] == player:

# roster1.append(player)

# found = 1

# break

# if not found:

# for team\_player in teams[1]['players']:

# if team\_player['id'] == player:

# roster2.append(player)

# found = 1

# break

# if not found:

# raise ValueError('Player in mapping\_data not found in match[\'teams\']', player)

# might be a problem with leaving elo

# for an unactive team as-is

# since when a new player joins it may think a sub who was 6th in roster is now in first 5 players in

# roster making them active. need to think of a good way to keep track of this. maybe split 'roster'

# into 'active\_roster' and 'inactive\_roster' depending on if they played the previous game?

# when a player joins a new roster and is removed from old roster, recalculate the old roster's elo

# (sans the player). if the old roster now has 4 players, should set the elo to 4/5 the previous elo?

# should we label teams by league (which may change name) or region? be consistent

# potentially add some modifier or winstreaks (within the tournament?) hard to do when games aren’t always in order

# rookies get ELO according to ELO of region's lowest Worlds finish

# (or most recent international tournament)

# (or average region ELO \* const)

# team ELO calculated as average of its roster

# calculate new ELOs as players move teams

# treat subs who never play a game as new players (i.e., give them base region ELO)

# teams such as T1 have times when players are frequently being subbed out

# therefore need to keep track of every players last known ELO

# will need to check each game that the same players are still in the team, therefore

# must keep track of roster and whether a team is active

# e.g., Lowkey Esports in VCS 2020 Spring becomes Team Secret after Week 2

# will need to find what team each player of the (new) team Team Secret were on

# and calculate the ELO of the new team appropriately

# since all players were on Lowkey Esports this should just equal the new ELO

# will then need to set Lowkey Esports to inactive and Team Secret to active

# can move teams into a pandas dataframe

# each entry is a team with its name, current players, last played game, ELO, etc.

# whether a team is active or not can be calculated when needed by checking if

# a team has at least 5 players or if its most recent game was within, e.g., 6 months

# how to deal with retired players?