Final Project Check-In

April 26, 2021

1 Final Project Check-In

League of Legends is a strategic 5v5 video game in which players control characters, called "champions," with the primary objective of destroying the opposing team's base. A "meta" is a term used to describe a collection of strategies, many of which revolve around choice of champions, items, or playstyle, which are widely utilized by the general community or competitive scene at a given time. Metas may vary for multiple reasons, such as developers updating the game over time and different regions developing distinct philosophies on how to play the game. Our primary objective with this data is to characterize the various champion metas that have emerged across different regions of competitive League of Legends, exploring how metas vary across regions and over time.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

[2]:
```
data = pd.read_csv('.\data\LeagueofLegends.csv')
```

[3]:
```
positions = ['Top', 'Jungle', 'Middle', 'ADC', 'Support']

regions = data['League'].unique()
major_regions = ['EULCS', 'LCK', 'NALCS', 'WC'] # EULCS - Europe, LCK - Korea,␣
↪NALCS - US/Canada, WC - World Championship; unfortunately, China not␣
↪included in the dataset
years = data['Year'].unique()
seasons = data['Season'].unique()
teams = list(set(data['blueTeamTag'].unique()).union(set(data['redTeamTag'].
↪unique())))
```

Below we condense the data to make it easier to work with. This includes removing unnecessary columns and restricting the data to only that of the major regions.

[4]:
```
cols = [team + pos + 'Champ' for team in ['blue', 'red'] for pos in positions]
data_condensed = data[['League', 'Year', 'Season', 'blueTeamTag', 'bResult',␣
↪'rResult', 'redTeamTag'] + cols]
data_condensed = data_condensed[data_condensed['League'].isin(major_regions)]
data_condensed.head()
```

[4]:

	League	Year	Season	blueTeamTag	bResult	rResult	redTeamTag	blueTopChamp
0	NALCS	2015	Spring	TSM	1	0	C9	Irelia
1	NALCS	2015	Spring	CST	0	1	DIG	Gnar
2	NALCS	2015	Spring	WFX	1	0	GV	Renekton
3	NALCS	2015	Spring	TIP	0	1	TL	Irelia
4	NALCS	2015	Spring	CLG	1	0	T8	Gnar

	blueJungleChamp	blueMiddleChamp	blueADCChamp	blueSupportChamp	redTopChamp
0	RekSai	Ahri	Jinx	Janna	Gnar
1	Rengar	Ahri	Caitlyn	Leona	Irelia
2	Rengar	Fizz	Sivir	Annie	Sion
3	JarvanIV	Leblanc	Sivir	Thresh	Gnar
4	JarvanIV	Lissandra	Tristana	Janna	Sion

	redJungleChamp	redMiddleChamp	redADCChamp	redSupportChamp
0	Elise	Fizz	Sivir	Thresh
1	JarvanIV	Azir	Corki	Annie
2	LeeSin	Azir	Corki	Janna
3	Nunu	Lulu	KogMaw	Janna
4	RekSai	Lulu	Corki	Annie

As a preliminary form of analysis, we want to take the raw counts of the number of times each champion was picked in a given year by a particular region. As an example, we do this for the year 2016.

[5]:
```
data_2016 = data_condensed[data_condensed['Year'] == 2016]
data_2016.head()
```

[5]:

	League	Year	Season	blueTeamTag	bResult	rResult	redTeamTag
245	NALCS	2016	Spring	TSM	0	1	CLG
246	NALCS	2016	Spring	C9	0	1	IMT
247	NALCS	2016	Spring	DIG	0	1	NRG
248	NALCS	2016	Spring	REN	1	0	TL
249	NALCS	2016	Spring	FOX	1	0	TIP

	blueTopChamp	blueJungleChamp	blueMiddleChamp	blueADCChamp
245	DrMundo	Elise	TwistedFate	Tristana
246	Ryze	LeeSin	Ekko	MissFortune
247	Malphite	Kindred	Anivia	Ezreal
248	DrMundo	RekSai	Orianna	Kalista
249	Lulu	Elise	Lissandra	Lucian

	blueSupportChamp	redTopChamp	redJungleChamp	redMiddleChamp	redADCChamp
245	Alistar	Jax	RekSai	Lissandra	Kalista
246	Braum	Chogath	RekSai	TwistedFate	Lucian
247	Alistar	Fiora	LeeSin	Viktor	Lucian
248	Alistar	Gnar	Elise	Viktor	Lucian
249	Thresh	Trundle	Poppy	Viktor	Kindred

```
    redSupportChamp
245            Bard
246           Janna
247           Janna
248           Janna
249         Alistar
```

[6]:
```
# initialize a dictionary for each region, each of which will contain the␣
↪counts for each champion
regional_champ_counts = {}
for region in major_regions:
    regional_champ_counts[region] = {}

# loop through rows, incrementing the respective counters for all champions in␣
↪that game
for idx, row in data_2016.iterrows():
    region = row[0]
    for champ in row[7:]: # for every champion present in that game
        if champ in regional_champ_counts[region].keys():
            regional_champ_counts[region][champ] += 1
        else: # intialize new entry in dictionary, if necessary
            regional_champ_counts[region][champ] = 1
```

[7]:
```
# for each region, create lists of champions and their counts, sorted by count
for region, champ_counts in regional_champ_counts.items():
    champ_list = np.array(list(champ_counts.keys()))
    counts = np.array(list(champ_counts.values()))
    idx = np.flip(np.argsort(counts))
    champ_list = champ_list[idx]
    counts = counts[idx]

    if len(champ_list) < 1:
        continue

    print(region)
    for i in range(min(len(champ_list), 5)):
        print(champ_list[i], counts[i])
    print()
```

```
EULCS
Lucian 162
Braum 158
Gragas 147
Elise 142
Sivir 138
```

```
LCK
Alistar 244
Elise 237
Braum 200
Lucian 197
Trundle 196

NALCS
RekSai 212
Braum 186
Karma 166
Lucian 164
Gragas 157

WC
Karma 42
Jhin 39
LeeSin 37
Olaf 36
Caitlyn 34
```

The above solution is somewhat crude and does not particularly take advantage of Panda's built-in data manipulation tools. We include it, however, as it will likely form the general template for our A-priori (the dictionary of counts can easily be modified to take pairs of champions as keys and store additional information such as number of wins). However, below is a slightly cleaner solution for the purposes of finding counts of individual champions under more complex and explorational filtering criteria. While this method is nice for specifically that purpose, this will not easily generalize to A-priori, nor to counting banned champions or common matchups.

[8]:
```
dfs = []
for pos in positions:
    # generate counts for each champion when on blue team
    blue = data_condensed.groupby(['League', 'Year', 'Season', 'blue'+ pos␣
↪+'Champ']).size().reset_index(name='blue_counts').
↪sort_values(['Year','League','blue_counts'], ascending = [True, True, False])
    blue = blue.rename(columns = {'blue'+ pos +'Champ': 'Champion'})

    # generate counts for each champion when on red team
    red = data_condensed.groupby(['League', 'Year', 'Season', 'red'+ pos␣
↪+'Champ']).size().reset_index(name='red_counts').
↪sort_values(['Year','League','red_counts'], ascending = [True, True, False])
    red = red.rename(columns = {'red'+ pos +'Champ': 'Champion'})

    # combine these data frames and calculate an aggregate count of the number␣
↪of times
    # a champion is present in (either side of) a game
```

```
    pos_df = blue.merge(red, on=['League', 'Year', 'Season', 'Champion'],␣
↪how='outer')
    pos_df['blue_counts'] = pos_df['blue_counts'].fillna(0)
    pos_df['red_counts'] = pos_df['red_counts'].fillna(0)
    pos_df['counts'] = pos_df['blue_counts'] + pos_df['red_counts']
    pos_df['Position'] = pos
    dfs.append(pos_df)

counts_df = pd.concat(dfs)
counts_df.head()
```

[8]:

	League	Year	Season	Champion	blue_counts	red_counts	counts	Position
0	WC	2014	Summer	Maokai	21.0	6.0	27.0	Top
1	WC	2014	Summer	Ryze	18.0	16.0	34.0	Top
2	WC	2014	Summer	Rumble	13.0	12.0	25.0	Top
3	WC	2014	Summer	Irelia	9.0	6.0	15.0	Top
4	WC	2014	Summer	Alistar	4.0	0.0	4.0	Top

We can easily verify that this produces the same results as the previous method by finding that the most picked champions (in any role) during the year of 2016 are the same as above:

[9]:
```
# restrict to 2016 and major regions
(counts_df[(counts_df['Year'].eq(2016)) & (counts_df['League'].
↪isin(major_regions))]
# total the counts by champion within each league
    .groupby(['League', 'Champion'])['counts'].sum()
# convert back into a dataframe rather than a series
    .reset_index(name='counts')
# sort results, most importantly such that counts are in descending order
    .sort_values(['League', 'counts'], ascending = [True, False])
# get first five in each region
    .groupby(['League']).head(5))
```

[9]:

	League	Champion	counts
42	EULCS	Lucian	162.0
7	EULCS	Braum	158.0
20	EULCS	Gragas	147.0
14	EULCS	Elise	142.0
68	EULCS	Sivir	138.0
96	LCK	Alistar	244.0
112	LCK	Elise	237.0
104	LCK	Braum	200.0
139	LCK	Lucian	197.0
175	LCK	Trundle	196.0
256	NALCS	RekSai	212.0
197	NALCS	Braum	186.0
225	NALCS	Karma	166.0

```
236  NALCS   Lucian   164.0
213  NALCS   Gragas   157.0
312     WC    Karma    42.0
310     WC     Jhin    39.0
317     WC   LeeSin    37.0
327     WC     Olaf    36.0
299     WC  Caitlyn    34.0
```

From this we get a very high-level understanding of the various 2016 metas in each region and how they differ. We can see that Braum was the only champion that was top three across more than one region in terms of times played. Meanwhile, Karma, which was a high-presence champion only in North America, did end up also having a high play rate at the World Championship (denoted "WC"), which could suggest that North American teams somewhat helped "shape" the meta in international competition during 2016.

We can further separate the data by year half (Spring or Summer) as well as role (Top, Jungle, etc.):

[10]:
```
counts_df[(counts_df['Year'].eq(2016)) & (counts_df['League'].
↪isin(major_regions))].groupby(['Season', 'League', 'Position',␣
↪'Champion'])['counts'].sum().reset_index(name='counts').
↪sort_values(['Season', 'Position', 'League', 'counts'], ascending = [True,␣
↪True, True, False]).groupby(['Season', 'League', 'Position']).head(1)
```

[10]:

	Season	League	Position	Champion	counts
8	Spring	EULCS	ADC	Lucian	66.0
103	Spring	LCK	ADC	Lucian	133.0
208	Spring	NALCS	ADC	Lucian	63.0
13	Spring	EULCS	Jungle	Elise	49.0
112	Spring	LCK	Jungle	Elise	124.0
213	Spring	NALCS	Jungle	Elise	59.0
39	Spring	EULCS	Middle	Lissandra	31.0
143	Spring	LCK	Middle	Lulu	62.0
233	Spring	NALCS	Middle	Corki	36.0
55	Spring	EULCS	Support	Braum	57.0
160	Spring	LCK	Support	Alistar	168.0
253	Spring	NALCS	Support	Alistar	58.0
85	Spring	EULCS	Top	Poppy	36.0
190	Spring	LCK	Top	Poppy	105.0
286	Spring	NALCS	Top	Poppy	37.0
306	Summer	EULCS	ADC	Sivir	108.0
415	Summer	LCK	ADC	Sivir	134.0
505	Summer	NALCS	ADC	Sivir	116.0
611	Summer	WC	ADC	Jhin	39.0
321	Summer	EULCS	Jungle	RekSai	107.0
429	Summer	LCK	Jungle	RekSai	123.0
521	Summer	NALCS	Jungle	RekSai	175.0
621	Summer	WC	Jungle	LeeSin	36.0

351	Summer	EULCS	Middle	Viktor	73.0
456	Summer	LCK	Middle	Viktor	74.0
557	Summer	NALCS	Middle	Viktor	90.0
640	Summer	WC	Middle	Viktor	23.0
359	Summer	EULCS	Support	Braum	101.0
462	Summer	LCK	Support	Braum	105.0
564	Summer	NALCS	Support	Braum	138.0
647	Summer	WC	Support	Karma	37.0
379	Summer	EULCS	Top	Gnar	81.0
473	Summer	LCK	Top	Ekko	91.0
586	Summer	NALCS	Top	Irelia	77.0
667	Summer	WC	Top	Rumble	32.0

This indicates that within a given half of the 2016 season, the most picked champions within each role were fairly consistent between regions (not including the World Championship), meaning the difference in metas may not be as pronounced as the full-year, all-position data may have suggested. Notable exceptions are mid lane during the spring and top lane during the fall, both of which saw all three regions having distinct most-picked champions.

More detailed analysis like this may prove to be more accurate, but may also be too overwhelming to compare between each year. When moving into A-priori and other further analysis, we will likely just use the all-position data for each half-year

Moving forward, we will at least: 1) Apply A-priori to determine not only what champions are common, but also what champion combinations or opposing matchups are common, 2) Include winrate data for champions, champion pairs, and champion matchups to evaluate the performance of "meta picks," 3) Extend our analysis to the remaining years in the dataset, and 4) Cluster seasons/regions based on common champion picks.