1. Introduction

1.1 Background

In recent years, the Chinese society has been in a stage of rapid development. People’s interest in online gaming has gradually intensified with the progress and development of the society. In this context, the new industry of eSports has emerged. E-sports was listed as a formal sports competition by the State Sports General Administration in 2003 since when its development began to accelerate. Although it experienced the low tide brought by the global financial crisis, it quickly regained its upward momentum thanks to the support of a large game user base.

Ranking is a widely-studied problem in eSport. Most of the competitions involve numerous teams and many factors, making the gaming data more complicated. Therefore, a fair and transparent ranking rule is particularly important. We need to build a model that can overcome many uncertain factors, so that the ranking results can accurately reflect the true strength of the team.

A good ranking algorithm should meet the following basic requirements.

- Order preservation. We believe that the true strength level of each team is reflected in the ranking, so according to the purpose of the ranking, the ranking order is consistent with the true level of each team reflected in the transcript

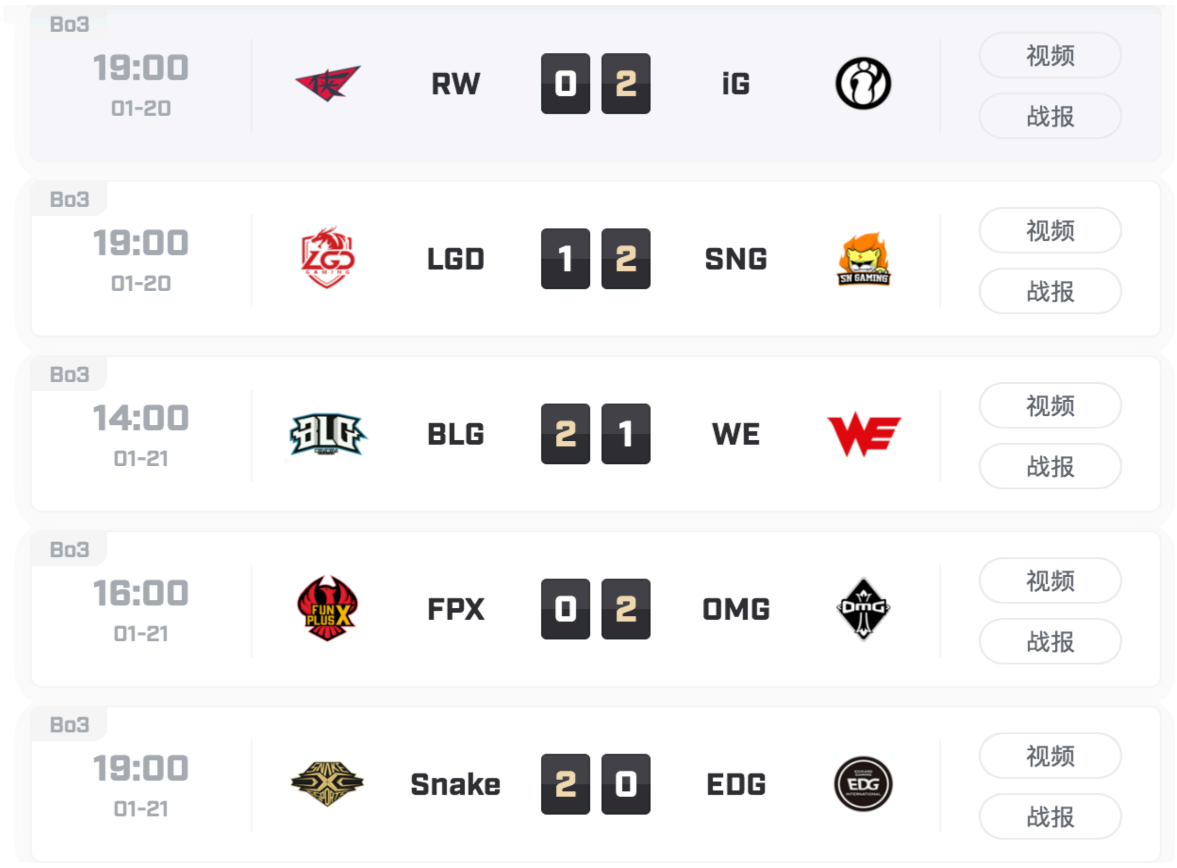
- Stability. Minor changes will not result in a huge impact on the rankings result.

1.2 Data Description

In predicting ranking in Online eSports, it is critical to select a suitable eSports dataset and apply different ranking algorithms and mathematical methods so as to make relatively accurate predictions on the rankings of teams in the future.

Here, we chose the gaming data of LPL(League of Legends Pro League) 2019 Spring as our raw data. LPL is the highest level professional competition of the game League of Legends in China, whose data is easily to access and comparatively complete. The LPL 2019 Spring session, as a tournament, involves 16 teams, which are：EDG、RNG、IG、WE、OMG、FPX、SNG、SS、LGD、JDG、BLG、TOP、VG、RW, SDG and V5. All of the teams has participated in the game and each has competed against one another.

The scoring data was obtained through the website below. There are 10 weeks in the LPL 2019 Spring session, and 120 games altogether.



Bo3	19:00	01-20	RW	0	2	iG	视频	战报
Bo3	19:00	01-20	LGD	1	2	SNG	视频	战报
Bo3	14:00	01-21	BLG	2	1	WE	视频	战报
Bo3	16:00	01-21	FPX	0	2	OMG	视频	战报
Bo3	19:00	01-21	Snake	2	0	EDG	视频	战报

Figure：Scoring data recorded on the website

2. Algorithm design and implementation

—Using three main methods

2.1 Ranking using Massey's method

2.1.1 Algorithm Description

For any given game between Team i and Team j, the point differential for team i is the score of team i minus the score for team j. At any given point in a tournament, the point differential for team i will be the sum of their point differentials for the games they have played.

Massey's ratings are based on the simple principle that for each game the difference in the ratings should be equal to the point differential,

$\boldsymbol{r_i}\ \boldsymbol{r_j} = \boldsymbol{y_k}$ where $\boldsymbol{r_i}$ represents rating for team i, $\boldsymbol{r_j}$ represents rating for team j and $\boldsymbol{y_k}$ represents score for team i-score for team j for game k. This gives us a linear equation in the unknowns, $\boldsymbol{r_i}$ and $\boldsymbol{r_j}$ for every game played and thus gives us a system of linear equations.

Our goal here is to minimize such a matrix, i.e. $\mathbf{min}\ ||\boldsymbol{Xr} - \boldsymbol{v}||$, where X is a 120×16 matrix, representing the total games that 16 teams played. $\boldsymbol{r}$ =($\boldsymbol{r_1}$,$\boldsymbol{r_2}$,...,$\boldsymbol{r_{14}}$)represents the rating of the 16 teams, while $\boldsymbol{v}$=($\boldsymbol{v_1}$,$\boldsymbol{v_2}$,...,$\boldsymbol{v_{91}}$) represents the scores of the 120 games. Then we denote $\boldsymbol{M} = \boldsymbol{X^tX}$ and $\boldsymbol{P} = \boldsymbol{X^tv}$.

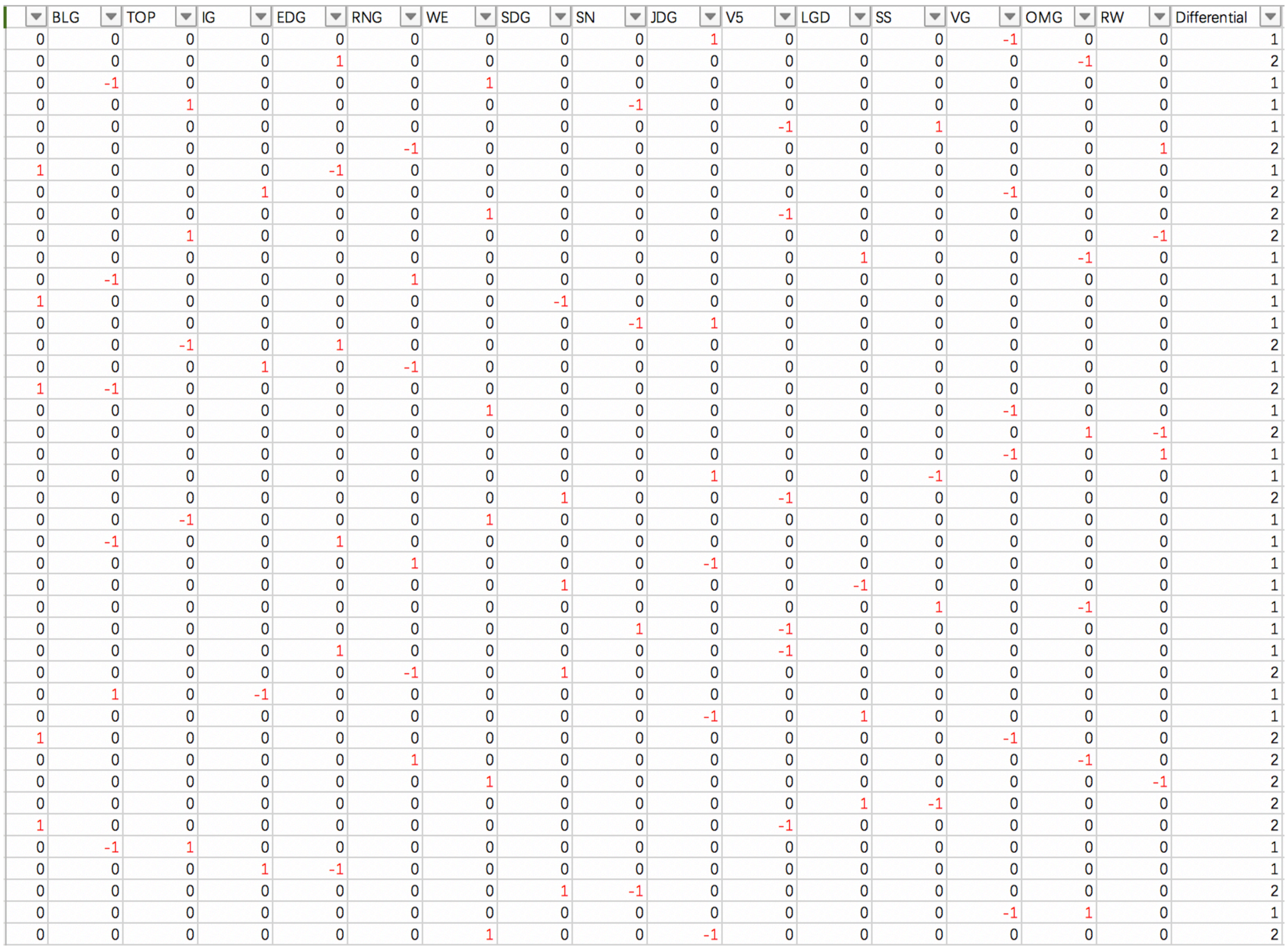
However the matrix M above is not invertible. To solve this problem, the n-th row of the matrix M is replaced by a row of *1*s and replacing the final row of P by a ***0***. The new matrix is invertible and and the new system is solvable and Massey's ratings are the solutions for $\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_n}$ for this system. We demote the adjusted Massey matrix by $\underline{\boldsymbol{M}}$ and the adjusted point differential column by $\underline{\boldsymbol{P}}$. The new equation is $\underline{\boldsymbol{M}}\boldsymbol{r} = \underline{\boldsymbol{P}}$. Then the final solution is $\boldsymbol{r} = (\underline{\boldsymbol{M}})^{-1}\underline{\boldsymbol{P}}$.

2.1.2 Data processing

In order to apply the Massey's method, it is necessary to first interpret our data into a matrix form. We create a form in excel to record the gaming situations of all the 120 games.

For each game, we use the number 1 to denote the status of the winner, and -1 to denote the status of the loser, while 0 means that the certain team did not compete in the game. The last column indicates the point differential of the competing teams.

The first 16 columns combined together to form a matrix ***X***, while the last column is understood as vector ***r***. The sample is as follows (Since the matrix is too large, only partial of the matrix is shown here)



	BLG	TOP	IG	EDG	RNG	WE	SDG	SN	JDG	V5	LGD	SS	VG	OMG	RW	Differential
0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	2
0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	-1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	1
0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	2
1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	0	2
0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	2
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	2
0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	1
0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	1
0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	1
1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	1	0	0	0	0	0	0	-1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	2
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	1
0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	2
0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	1
0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	1
0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	2
0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	2
0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	0	2
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	2
0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	2
1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	2
0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	1
0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	2

Figure: Processed data for the Massey's method

2.1.3 Implementation

```r
erg3020 <- read.csv("erg3020_data.csv")
names(erg3020)
Diff_KDA <- erg3020$WKDA - erg3020$LKDA
erg3020_data <- cbind(erg3020,Diff_KDA)
B <- as.matrix(erg3020[,1:16])
Bt <- t(B)
BtB <- Bt %*% B
new_BtB <- rbind(BtB[-16,],rep(1,16))
v <- as.matrix(erg3020[,17])
Btv <- Bt %*% v
new_Btv <- rbind(as.matrix(Btv[-16,1]),c(0))
#r * BtB = Btv
r <- solve(new_BtB) %*% new_Btv
rank(-r[,1])
```

The result is as follows:

```
# FPX BLG TOP IG EDG RNG WE SDG SN JDG V5 LGD SS VG OMG RW
# 1 8 3 2 6 4 7 5 10 9 13 11 12 16 14 15
```

2.2 Ranking using Bradley-Terry Model

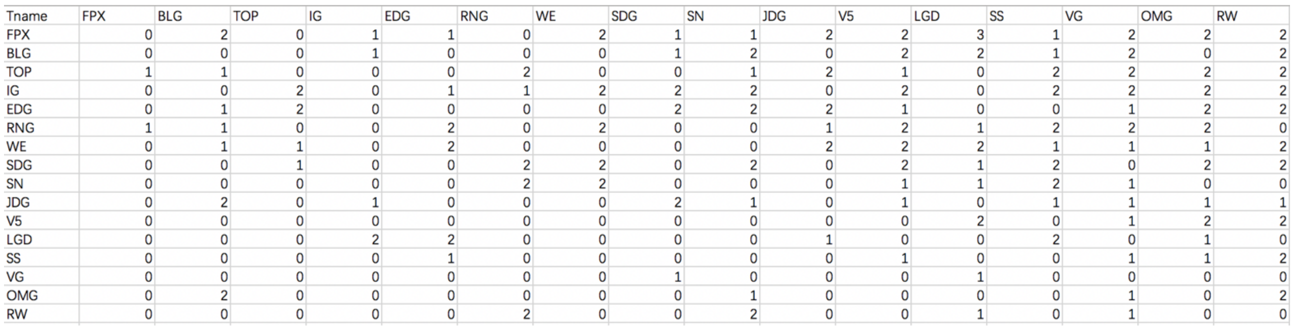
2.2.1 Algorithm Description

The Bradley-Terry model assumes that in a competition between any two players i and j, the odds that i beats j is **αi** / **αj** , where **αi** and **αj** are positive-valued parameters which might be thought of as representing 'ability'. The model can alternatively be expressed in the logit-linear form $\mathbf{P}(\boldsymbol{r_{ij}} = \boldsymbol{a}) = (\frac{\boldsymbol{\theta_i}}{\boldsymbol{\theta_i}+\boldsymbol{\theta_j}})^{\boldsymbol{a}}$ where $\boldsymbol{r_{ij}}$ = $\boldsymbol{a}$ means that team i beats team j $\boldsymbol{a}$ times. $\boldsymbol{\theta}$ = ($\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, … ,$\boldsymbol{\theta_{14}}$), where $\boldsymbol{\theta_m}$ is the rank of the team m. Our goal is to find $\boldsymbol{\theta_m}$ that maximizes above probability, subject to $\boldsymbol{\theta} > \mathbf{0}$, $\sum \boldsymbol{\theta_m} = \mathbf{1}$. Assuming independence of all contests, the parameters $\boldsymbol{\theta}_{\mathrm{i}}$, $\boldsymbol{\theta}_{\mathrm{j}}$, etc., can be estimated by maximum likelihood using standard software for generalized linear models. When calculating the maximum likelihood, we first obtain the log-likelihood of the parameter $\mathbf{p} = (\boldsymbol{p_1}, \boldsymbol{p_2} \dots \boldsymbol{p_n})$ as:

$$\mathbf{L}(\mathbf{p}) = \sum_{i}^{n}\sum_{j}^{n} \boldsymbol{w_{ij}} \, \mathbf{ln}(\boldsymbol{p_i} + \boldsymbol{p_j})$$

2.2.2 Data processing

Another dataset is built for *Bradley-Terry Model* and *Elo Algorithm*. This is a 16 x 16 matrix, where the team *i* in the row represents the winner and the team *j* in column is loser. And the matrix stores the number of victories for each team. For example, the number in row 1 and column 2 represent, team FPX beats team BLG for two times.



Tname	FPX	BLG	TOP	IG	EDG	RNG	WE	SDG	SN	JDG	V5	LGD	SS	VG	OMG	RW
FPX	0	2	0	1	1	0	2	1	1	2	2	3	1	2	2	2
BLG	0	0	0	1	0	0	0	1	2	0	2	2	1	2	0	2
TOP	1	1	0	0	0	2	0	0	1	2	1	0	2	2	2	2
IG	0	0	2	0	1	1	2	2	2	0	2	0	2	2	2	2
EDG	0	1	2	0	0	0	0	2	2	2	1	0	0	1	2	2
RNG	1	1	0	0	2	0	2	0	0	1	2	1	2	2	2	0
WE	0	1	1	0	2	0	0	0	0	2	2	2	1	1	1	2
SDG	0	0	1	0	0	2	2	0	2	0	2	1	2	0	2	2
SN	0	0	0	0	0	2	2	0	0	0	1	1	2	1	0	0
JDG	0	2	0	1	0	0	0	2	1	0	1	0	1	1	1	1
V5	0	0	0	0	0	0	0	0	0	0	0	2	0	1	2	2
LGD	0	0	0	2	2	0	0	0	0	1	0	0	2	0	1	0
SS	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	2
VG	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
OMG	0	2	0	0	0	0	0	0	1	0	0	0	0	1	0	2
RW	0	0	0	0	0	2	0	0	2	0	0	1	0	1	0	0

Figure: Processed data for Bradley-Terry Model

2.1.3 Implementation

```r
# Transform the dataform to a 16*16 matrix
data1<-
read.csv("/Users/raoshun/Desktop/ERG3020 Web Analytics and Intelligence/Assignment/report/
BT_data1.csv", header=T)
row.names(data1) = data1[,1]
data1 = data1[,-1]

# Transform the original optimization function to a convex function
library(CVXR)
sita = Variable(16)
function1 = 0
for (i in 1:16){
  for (j in 1:16){
    part1 =  data1[i,j]*sita[i]
    sel = matrix(rep(0,16))
    sel[i] = 1
    sel[j] = 1
    part2 = log_sum_exp(sel*sita)
    part3 = data1[i,j]*part2
    part= part3-part1
    function1 = function1+part
    }
  }

# Get the rank
constraint1 = list(sum(exp(sita))<=1) # subject to the constraint
problem1 <- Problem(Minimize(function1),constraints = constraint1)
result1 <- solve(problem1)
rank1 = result1$getValue(sita)
row.names(rank1) = row.names(data1)
rank(-(rank1[,1]))
```

The result is as follows:

```
# FPX BLG TOP  IG EDG RNG  WE SDG  SN JDG  V5 LGD  SS  VG OMG  RW
#   1   8   3   2   6   5   7   4  10   9  12  11  13  16  14  15
```

2.3 Ranking using Elo algorithm
2.3.1 Algorithm Description

In Elo Algorithm, Players with higher ELO rating have a higher probability of winning a game than a player with lower ELO rating. After each game, ELO rating of players is updated. If a player with higher ELO rating wins, only a few points are transferred from the lower rated player. However, if lower rated player wins, then transferred points from a higher rated player are far greater. We denote:

r1: rating of player 1   r2: rating of player 2

P1: Probability of winning of player with rating2
P2: Probability of winning of player with rating1.

$$P1 = \frac{1}{1 + 10^{\frac{r1-r2}{400}}}$$

$$P2 = \frac{1}{1 + 10^{\frac{r2-r1}{400}}}$$

And P1 + P2 = 1.

The rating of player is updated using the formula given below :-

$$\boldsymbol{r'_1 = r_1 + k \times (s_i - P_{ij})}$$

$\boldsymbol{s_i}$ is 1 if team i wins and is 0 if team i loses. $\mathbf{P}_{\boldsymbol{ij}}$ is the probability that team i beats team j, which depends on the existed competitions and is better to be under logistics statistics. According to international practice, we set the initial raking of all the teams 1500, and $\boldsymbol{k}$ = 32. We further use this function to update the rank for each team.

2.3.2 Data processing

Here, we use the same dataset as the one used in the Massey' method. Refer to 2.1.2 for this part.

2.3.3 Implementation

```
erg3020 <- read.csv("erg3020_data.csv")
names(erg3020)
elo_data <- erg3020[,1:16]
initial_matrix <- matrix(rep(1500,16),nrow=1)
colnames(initial_matrix) <- names(erg3020)[1:16]
for (i in 1:120){
  for (j in 1:16){
    if (elo_data[i,j]==1){
      win_location <- j
    }
    if (elo_data[i,j]==-1){
      lose_location <- j
    }
  }
  E_win <- 1/(1+10^((initial_matrix[lose_location]-initial_matrix[win_location])/400))
  E_lose <- 1/(1+10^((initial_matrix[win_location]-initial_matrix[lose_location])/400))
  point_win <- initial_matrix[win_location] + 32*(1-E_win)
  point_lose <- initial_matrix[lose_location] + 32*(0-E_lose)
  initial_matrix[win_location] <- point_win
  initial_matrix[lose_location] <- point_lose
}
rank(-initial_matrix)
```

The result is as follows:

```
# FPX BLG TOP  IG EDG RNG  WE SDG  SN JDG  V5 LGD  SS  VG OMG  RW
#   1   9   3   2   8   4   5   7  10  16  15  11  12  16  14  13
```

3. Evaluation: Simulation Using the Three Models

3.1 Description

By far we have presented three different method, namely, Massey's method, Bradley-Terry Model and Elo algorithm. In order to see which of the methods gives the best result, we use random simulation to evaluate the accuracy of the model.

The methodology is as follows: We use the notation T1,T2,T3...T16 to represent for the 16 teams, whose real rankings are arranged in decending order. We first simulate 100 seasons of games which are similar to the figures that are processed data for the Massey's method as well as Bradley-Terry Model. Then we apply the three methods above to calculate the sum of the absolute average value of differences between the real rankings and the simulated result of each team. In this way can we find out the model whose accuracy stands out.

$$Error = \sum_{i=1}^{16} |\bar{\hat{R}}_i - R_i|$$

3.2 Implementation of the simulation

3.2.1 Simulating using Massey's method

After we collected the data, we find that 16 teams have 120 games just right. According to C_{16}^2, we find that if two different teams fight only once, the number of game is 120.Therefore, we assume that only one competition takes place between any two teams, and the real strength of Ti is i. Therefore, for any two teams, denote Ti and Tj, the probability for the former to win is $\frac{i}{i+j}$, and the probability for the latter to win is $\frac{j}{i+j}$. The point differential of the two teams is a random number chosen between 1 and 2.

```r
game_initial <- matrix(rep(0,1920),nrow = 120,ncol = 16)
colnames(game_initial) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T1
2","T13","T14","T15","T16")
n <- 1
for (i in 1:16){
  for (j in 1:16){
      if (i < j){
        win_prob <- j/(i+j)
        lose_prob <- i/(i+j)
        result <- sample(c(1,-1),1,prob = c(win_prob,lose_prob))
        game_initial[n,j] <- result
        game_initial[n,i] <- 0-result
        n <- n+1
      }
    }
  }
  v_random <- matrix(sample(c(1,2),120,replace = T),ncol = 1)
  B <- as.matrix(game_initial[,1:16])
  Bt <- t(B)
  BtB <- Bt %*% B
  new_BtB <- rbind(BtB[-16,],rep(1,16))
  v <- v_random
  Btv <- Bt %*% v
  new_Btv <- rbind(as.matrix(Btv[-16,1]),c(0))
  #r * BtB = Btv
  r <- solve(new_BtB) %*% new_Btv
  all_matrix <- rbind(all_matrix,rank(-r[,1]))
  k <- k+1
}
#all_matrix <- all_matrix[-1,]
average_matrix <- matrix(c(16:1),nrow=1)
colnames(average_matrix) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
for (i in 1:16){
  average_matrix[1,i] <- (sum(all_matrix[-1,i])/100)-all_matrix[1,i]
}
sum(abs(average_matrix))
```

3.2.2 Simulation using Bradley-Terry Model

```r
all_matrix <- matrix(c(16:1),nrow=1)
colnames(all_matrix) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
k <- 1
while (k <= 100){
  game_initial <- matrix(rep(0,256),nrow=16)
  colnames(game_initial) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
  rownames(game_initial) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
  for (i in 1:16){
    for (j in 1:16){
      if (i<j){
        win_prob <- j/(i+j)
        lose_prob <- i/(i+j)
        result <- sample(c(1,-1),1,prob = c(win_prob,lose_prob))
        if (result == 1){
          game_initial[i,j] <- sample(c(1,2),1)
        }
        else{
          game_initial[j,i] <- sample(c(1,2),1)
        }
      }
    }
  }
  data1 <- game_initial
  library(CVXR)
  sita = Variable(16)
  function1 = 0

  for (i in 1:16){
    for (j in 1:16){
      part1 =  data1[i,j]*sita[i]
      sel = matrix(rep(0,16))
      sel[i] = 1
      sel[j] = 1
      part2 = log_sum_exp(sel*sita)
      part3 = data1[i,j]*part2
      part= part3-part1
      function1 = function1+part
    }
  }
  constraint1 = list(sum(exp(sita))<=1)
  problem1 <- Problem(Minimize(function1),constraints = constraint1)
  result1 <- solve(problem1)
  rank1 = result1$getValue(sita)
  row.names(rank1) = row.names(data1)  #??
  #rank1[order(rank1[,1],decreasing = T),]
  #rank(-rank1[,1])
  all_matrix <- rbind(all_matrix,rank(-rank1[,1]))
  k <- k+1
}
#all_matrix <- all_matrix[-1,]
average_matrix <- matrix(c(16:1),nrow=1)
colnames(average_matrix) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
for (i in 1:16){
  average_matrix[1,i] <- (sum(all_matrix[-1,i])/100)-all_matrix[1,i]
}
sum(abs(average_matrix))
```

3.2.3 Simulation using Elo Algorithm

```r
all_matrix <- matrix(c(16:1),nrow=1)
colnames(all_matrix) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
k <- 1
while (k <= 100){
  game_initial <- matrix(rep(0,1920),nrow = 120,ncol = 16)
  colnames(game_initial) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T12","T13","T14","T15","T16")
  n <- 1
  for (i in 1:16){
    for (j in 1:16){
      if (i < j){
        win_prob <- j/(i+j)
        lose_prob <- i/(i+j)
        result <- sample(c(1,-1),1,prob = c(win_prob,lose_prob))
        game_initial[n,j] <- result
        game_initial[n,i] <- 0-result
        n <- n+1
      }
    }
  }
  initial_matrix <- matrix(rep(1500,16),nrow=1)
  colnames(initial_matrix) <- names(game_initial)[1:16]
  for (i in 1:120){
    for (j in 1:16){
      if (game_initial[i,j]==1){
        win_location <- j
      }
      if (game_initial[i,j]==-1){
        lose_location <- j
      }
    }
    E_win <- 1/(1+10^((initial_matrix[lose_location]-initial_matrix[win_location])/400))
    E_lose <- 1/(1+10^((initial_matrix[win_location]-
initial_matrix[lose_location])/400))
    point_win <- initial_matrix[win_location] + 32*(1-E_win)
    point_lose <- initial_matrix[lose_location] + 32*(0-E_lose)
    initial_matrix[win_location] <- point_win
    initial_matrix[lose_location] <- point_lose
  }
  all_matrix <- rbind(all_matrix,rank(-initial_matrix))
  k <- k+1
}
#all_matrix <- all_matrix[-1,]
average_matrix <- matrix(c(16:1),nrow=1)
colnames(average_matrix) <- c("T1","T2","T3","T4","T5","T6","T7","T8","T9","T10","T11","T1
2","T13","T14","T15","T16")
for (i in 1:16){
  average_matrix[1,i] <- (sum(all_matrix[-1,i])/100)-all_matrix[1,i]
}
sum(abs(average_matrix))
```

3.3 Summary

After the simulation of 100 random seasons, we can get three results to judge the accuracy of the three models.

Ranking Methods	Error
Massey's method	18.36
Bradley-Terry Model	114.8
Elo algorithm	17.6

We can conclude from the table that both Massey's method and Elo algorithm are more suitable to use to give the eSports ranking since their error is much much smaller than Bradley-Terry Model's. However, this simulation is based on ranking by the score difference of each game. Our score difference can only be 1 or 2 since the competition system of LPL does not provide a larger score differences (like 20 or 30 score difference in NBA). The Bradley-Terry Model may be more efficient if the score difference is larger.

4. Improving the model: Using entropy weight

4.1.1ntroduction of entropy weight

When making a decision involving numerous attributes and criterions, it is important to evaluate and select over the available alternatives and assign weight to each criterion. However, since each criterion has a different meaning, they cannot be simply assumed that they all have equal weights, and as a result, finding the appropriate weight for each criterion is one the main points.

In information theory, entropy is a measure of uncertainty. The larger the amount of information, the smaller the uncertainty and the smaller the entropy; the smaller the amount of information, the greater the uncertainty and the greater the entropy. According to the characteristics of entropy, the greater the degree of dispersion of the index, the greater the impact, or weight of this factor on the comprehensive evaluation.

In our problem, the structure of the alternative performance matrix is shown as below. Massey's method, Bradley-Terry Model and Elo algorithm are herein after referring to as three different criterions. Here, x_{ij} is the rating of alternative i with respect to criterion j and w_j is the weight of criterion j (in this paper, we consider the case that the rating of alternative i with respect to criterion j

	Massey's Method	Bradley-Terry Model	Elo's Algorithm
Alternative 1	X_{11}	X_{12}	X_{13}
Alternative 2	X_{21}	X_{22}	X_{23}
…	…	…	…
Alternative n	X_{n1}	X_{n2}	X_{n3}
	W_1	W_2	W_3

4.2 Algorithm description

According to Li, X., Wang, K., Liu, L., Xin, J., Yang, H., & Gao, C. (2011), the entropy can be calculated as follow:

$$E_j = -\ln(n)^{-1} \sum_{i=1}^{n} \ln p_{ij}$$

Where p_{ij} is noted as:

$$p_{ij} = \frac{Y_{ij}}{\sum_{i=1}^{n} Y_{ij}}$$

Therefore, the entropy weight is known as:

$$W_i = \frac{1 - E_i}{3 - \sum E_i} \ (i = 1,2,3)$$

4.3 Implementation

Since the original scores of each method on which the rankings depend are of different scales. Therefore, the first step is to normalize all the data obtained from the former steps using the scaling standard：

$$Y_{ij} = \frac{X_{ij} - \min(X_i)}{\max(X_i) - \min(Xi)}$$

And apply the data to this formula.

```
col1<-
c(1.1561265, 0.1875000, 0.6897233, 0.9375000, 0.3750000, 0.4375000, 0.3125000, 0.4375000,-
0.3125000, 0.0625000, -0.6250000, -0.3458498, -0.6250000, -1.0000000, -0.7500000, -
0.9375000)

col2<-c(-2.145455, -2.671200, -2.463173, -2.241147, -2.528466, -2.464303, -2.528840, -
2.464301,-3.038931, -2.837499, -3.290629, -3.156339, -3.444029, -4.541894, -3.444781, -
3.445914)

col3<-
c(1645.787, 1498.134, 1565.325, 1571.855, 1520.8, 1558.871, 1557.633, 1538.172, 1487.46, 1
540.562, 1409.334, 1452.295, 1440.481, 1374.738, 1416.65, 1421.957)

col1=(col1-min(col1))/(max(col1)-min(col1))
col2=(col2-min(col2))/(max(col2)-min(col2))
col3=(col3-min(col3))/(max(col3)-min(col3))
```

The result obtained from the code above is the scaled input, which are:

```
[1.00000000, 0.55075618, 0.78368468, 0.89860219, 0.63771768, 0.66670485, 0.60873052, 0.666
70485, 0.31885884, 0.49278185, 0.17392300, 0.30339138, 0.17392300, 0.00000000, 0.11594867,
 0.02898717]';
```

```
[1.0000000, 0.7806141, 0.8674208, 0.9600691, 0.8401749, 0.8669493, 0.8400189, 0.8669501, 0
.6271651, 0.7112199, 0.5221351, 0.5781724, 0.4581235, 0.0000000, 0.4578097, 0.4573369]';


[1.0000000, 0.4552535, 0.7031459, 0.7272375, 0.5388767, 0.6793347, 0.6747673, 0.6029685, 0
.4158731, 0.6117861, 0.1276374, 0.2861365, 0.2425502, 0.0000000, 0.1546289, 0.1742084]';
```

Use the data to implement the entropy weight algorithm

```
function weights = EntropyWeight(R)

[rows,cols]=size(R);
k=1/log(rows);

f=zeros(rows,cols);
sumBycols=sum(R,1);

for i=1:rows
  for j=1:cols
      f(i,j)=R(i,j)./sumBycols(1,j);
  end
end

lnfij=zeros(rows,cols);

for i=1:rows
  for j=1:cols
      if f(i,j)==0
          lnfij(i,j)=0;
      else
          lnfij(i,j)=log(f(i,j));
      end
  end
end

Hj=-k*(sum(f.*lnfij,1));
weights=(1-Hj)/(cols-sum(Hj));
end
col1=[1.00000000, 0.55075618, 0.78368468, 0.89860219, 0.63771768, 0.66670485, 0.60873052, 0.66670485, 0.31885884, 0.49278185, 0.17392300, 0.30339138, 0.17392300, 0.00000000, 0.11594867, 0.02898717]';

col2=[1.0000000, 0.7806141, 0.8674208, 0.9600691, 0.8401749, 0.8669493, 0.8400189, 0.8669501, 0.6271651, 0.7112199, 0.5221351, 0.5781724, 0.4581235, 0.0000000, 0.4578097, 0.4573369]';

col3=[1.0000000, 0.4552535, 0.7031459, 0.7272375, 0.5388767, 0.6793347, 0.6747673, 0.6029685, 0.4158731, 0.6117861, 0.1276374, 0.2861365, 0.2425502, 0.0000000, 0.1546289, 0.1742084]';

testData=[col1,col2,col3];
EntropyWeight(testData)
```

The answer shows that:

```
ans =
0.4644 0.1762 0.3594
```

Which means that the weight of the three methods are 0.4644 0.1762 0.3594, respectively. That is to say, our modified model using entropy weight is:

$$\textbf{Ensemble Model} = \mathbf{0.0.4644M} + \mathbf{0.1763B} + \mathbf{0.3594E},$$

Where M = Normalized scores generated using Massey's Method;
B = Normalized scores generated using Bradley-Terry Model;
E = Normalized scores generated using Elo Algorithm.

4.4 Fitting in the new model

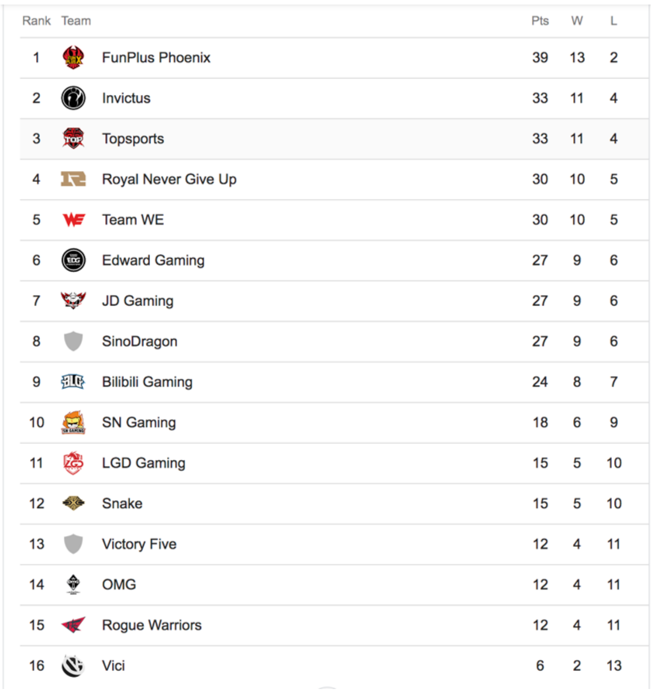
Now that we have already obtained a new model. We have interest in using it to rank all the teams.

```
ensemble_model=col1*0.4644+col2*0.1762+col3*0.3594
ensemble_model=matrix(ensemble_model)
row.names(ensemble_model) = row.names(data1)
rank(-(ensemble_model[,1]))
```

And finally we have the newest rank:

```
FPX BLG TOP  IG EDG RNG  WE SDG  SN JDG  V5 LGD  SS  VG OMG  RW
  1   9   3   2   7   4   6   5  10   8  13  11  12  16  14  15
```

It is worth mentioning that the teams in LPL 2019 Spring tournament are ranked based on the margin of victory. And our newest rank using entropy weight nearly perfectly match the rankings that LPL officially provides.



Rank	Team	Pts	W	L
1	FunPlus Phoenix	39	13	2
2	Invictus	33	11	4
3	Topsports	33	11	4
4	Royal Never Give Up	30	10	5
5	Team WE	30	10	5
6	Edward Gaming	27	9	6
7	JD Gaming	27	9	6
8	SinoDragon	27	9	6
9	Bilibili Gaming	24	8	7
10	SN Gaming	18	6	9
11	LGD Gaming	15	5	10
12	Snake	15	5	10
13	Victory Five	12	4	11
14	OMG	12	4	11
15	Rogue Warriors	12	4	11
16	Vici	6	2	13

Figure: Official rankings are based on the margin of victory

5. Reflection：Modify the input using KDA index

5.1 Introduction of KDA index

In this project, we will first use three major methods in ranking the teams, which are Massey method, Bradley-Terry method, and Elo algorithm.

However, the matches in the LPL 2019 Spring tournament use best-of-three criteria to determine which team to win, the score differences could not exceed 2. Since the differences are relatively too small to give a precise ranking result, we introduced a new scoring index, which is the KDA ratio.

The KDA ratio is the combined total of a team’s kills and assists, divided by the number of deaths in a game.

$$\text{KDA} = \frac{K + A}{D}$$

For example, if a team’s K/D/A is 18/4/11 in a game, its KDA ratio is : (18 + 11 = 29) / 4 = 7.25.

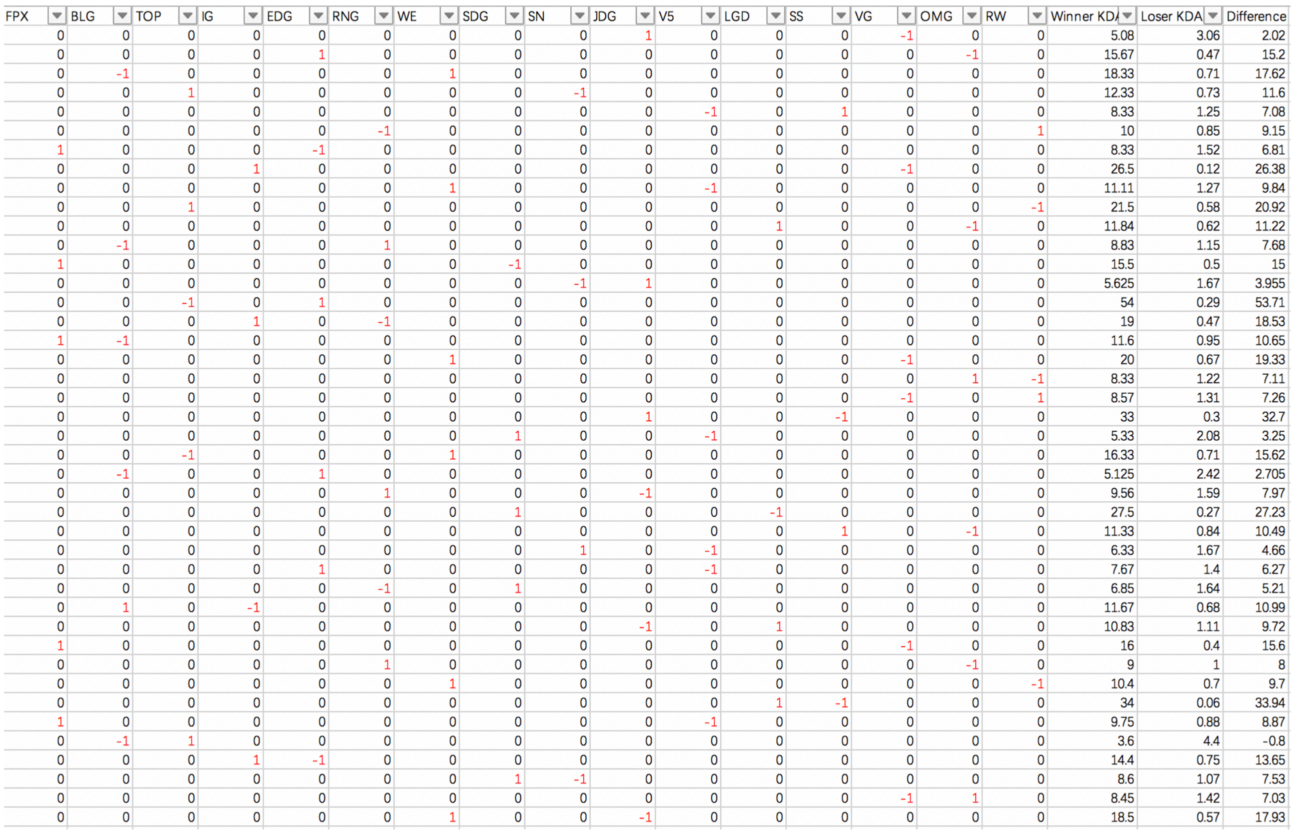
This to a large extent extends the score difference between a winner and a loser, which is expected to produce better ranking result.

It is worth noticing that, since Elo algorithm only concerns about winning or losing situations, using the KDA indices is not suitable here.

5.2 Implementation using KDA index

5.2.1 Massey’s Method

As mentioned, we hope to use KDA index to replace the point differentials to extinguish the ranking result. Here, we obtained the average KDA of each team in a game and calculate their differences. We then use the KDA differences to replace the point differentials and update our matrix used to apply the Massey’s method.



FPX	BLG	TOP	IG	EDG	RNG	WE	SDG	SN	JDG	V5	LGD	SS	VG	OMG	RW	Winner KDA	Loser KDA	Difference
0	0	0	0	0	0	0	0	0	1	0	0	0	-1	0	0	5.08	3.06	2.02
0	0	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	15.67	0.47	15.2
0	-1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	18.33	0.71	17.62
0	0	1	0	0	0	0	0	-1	0	0	0	0	0	0	0	12.33	0.73	11.6
0	0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	8.33	1.25	7.08
0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	1	10	0.85	9.15
1	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	8.33	1.52	6.81
0	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	0	26.5	0.12	26.38
0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	11.11	1.27	9.84
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	-1	21.5	0.58	20.92
0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	11.84	0.62	11.22
0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8.83	1.15	7.68
1	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	15.5	0.5	15
0	0	0	0	0	0	0	0	-1	1	0	0	0	0	0	0	5.625	1.67	3.955
0	0	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	54	0.29	53.71
0	0	0	1	0	-1	0	0	0	0	0	0	0	0	0	0	19	0.47	18.53
1	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.6	0.95	10.65
0	0	0	0	0	0	1	0	0	0	0	0	0	-1	0	0	20	0.67	19.33
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-1	8.33	1.22	7.11
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	1	8.57	1.31	7.26
0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	33	0.3	32.7
0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	5.33	2.08	3.25
0	0	-1	0	0	0	1	0	0	0	0	0	0	0	0	0	16.33	0.71	15.62
0	-1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	5.125	2.42	2.705
0	0	0	0	0	1	0	0	0	-1	0	0	0	0	0	0	9.56	1.59	7.97
0	0	0	0	0	0	0	1	0	0	0	-1	0	0	0	0	27.5	0.27	27.23
0	0	0	0	0	0	0	0	0	0	0	0	1	0	-1	0	11.33	0.84	10.49
0	0	0	0	0	0	0	0	1	0	-1	0	0	0	0	0	6.33	1.67	4.66
0	0	0	0	1	0	0	0	0	0	-1	0	0	0	0	0	7.67	1.4	6.27
0	0	0	0	0	-1	0	1	0	0	0	0	0	0	0	0	6.85	1.64	5.21
0	1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	11.67	0.68	10.99
0	0	0	0	0	0	0	0	0	-1	0	1	0	0	0	0	10.83	1.11	9.72
1	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	16	0.4	15.6
0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	0	9	1	8
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	-1	10.4	0.7	9.7
0	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	34	0.06	33.94
1	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	9.75	0.88	8.87
0	-1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3.6	4.4	-0.8
0	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	14.4	0.75	13.65
0	0	0	0	0	0	0	1	-1	0	0	0	0	0	0	0	8.6	1.07	7.53
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	0	8.45	1.42	7.03
0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	18.5	0.57	17.93

Figure: Processed data using KDA

Where the code using the new dataset is shown as below:

```r
B <- as.matrix(erg3020[,1:16])
Bt <- t(B)
BtB <- Bt %*% B
new_BtB <- rbind(BtB[-16,],rep(1,16))
v <- as.matrix(erg3020_data[,20])
Btv <- Bt %*% v
new_Btv <- rbind(as.matrix(Btv[-16,1]),c(0))
```

The result is as follows:

```
# FPX BLG TOP IG EDG RNG WE SDG SN JDG V5 LGD SS VG OMG RW
# 1 5 7 3 2 6 4 8 13 9 12 11 10 16 15 14
```

5.2.2 Bradley-Terry Model

```r
data2<-
read.csv("/Users/raoshun/Desktop/ERG3020 Web Analytics and Intelligence/Assignment/report/
BT_data2.csv", header=T)
row.names(data2) = data2[,1]
data2 = data2[,-1]

# Transform the original optimization function to a convex function
library(CVXR)
sita = Variable(16)
function2 = 0
for (i in 1:16){
  for (j in 1:16){
    part1 =  data2[i,j]*sita[i]
    sel = matrix(rep(0,16))
    sel[i] = 1
    sel[j] = 1
    part2 = log_sum_exp(sel*sita)
    part3 = data2[i,j]*part2
    part= part3-part1
    function2 = function2+part
  }
}

# Get the rank
constraint2 = list(sum(exp(sita))<=1) # subject to the constraint
problem2 <- Problem(Minimize(function2),constraints = constraint2)
result2 <- solve(problem2, solver="SCS")
rank2 = result2$getValue(sita)
row.names(rank2) = row.names(data2)
rank(-(rank2[,1]))
```

The result is as follows:

```
# FPX BLG TOP  IG EDG RNG  WE SDG  SN JDG  V5 LGD  SS  VG OMG  RW
#   1   7   5   6   2   8   3  10  13  11  12   9   4  16  15  14
```

5.3 Summary

We found that the ranking generated by KDA result in remarkable differences compare to the ranking generated using point differentials. Comparing to the real rankings and our result derived from entropy weights, the original method seems to be more accurate.

That is because that in LOL, winning does not rely on how many kills or assists that a team achieves. The goal of the game is to destroy the enemy nexus. Once a team has destroyed the enemy nexus, the game is over and that team is declared victorious.

6. Summary

In this project, we chose the gaming data of LPL(League of Legends Pro League) 2019 Spring as our raw data. In Part 2, we rank the teams using three different methods: Massey's Method, Bradley-Terry Model and Elo Algorithm. To evaluate their accuracy, we then use computer to produce the simulated rankings using the three methods. In Part 4, we improve our model using entropy weight method, and largely raise the accuracy. Finally, to seek for further advancements, we try to use KDA instead of points differentials to fit in the model.

We finally come to an conclusion that applying entropy weight method generates a fairly satisfying result, the model is:

$$\mathbf{Ensemble\ Model} = \mathbf{0.0.4644M} + \mathbf{0.1763B} + \mathbf{0.3594E,}$$

Where M = Normalized scores generated using Massey's Method;
B = Normalized scores generated using Bradley-Terry Model;
E = Normalized scores generated using Elo Algorithm.

Citation

Gao, Y. (2009). Rank of football teams. Retrieved from https://wenku.baidu.com/view/1e6e5eaf01f69e314232944d.html

Macabasco, A. (2019). How to Use the League of Legends Stats Tab to Improve - Mobalytics. Retrieved from https://mobalytics.gg/blog/how-to-use-the-league-of-legends-stats-tab-to-improve/

Li, X., Wang, K., Liu, L., Xin, J., Yang, H., & Gao, C. (2011). Application of the Entropy Weight and TOPSIS Method in Safety Evaluation of Coal Mines. Procedia Engineering, 26, 2085-2091. doi: 10.1016/j.proeng.2011.11.2410