BasicPremiereLeagueTablePrediction

2022-10-13

The first step in this Premier League table prediction is to read in historical data over the last 5 seasons.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)

pl22 = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PL21-22.csv')
pl21 = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PL20-21.csv')
pl20 = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PL19-20.csv')
pl19 = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PL18-19.csv')
pl18 = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PL17-18.csv')

premLge = merge(merge(merge(merge(
  pl22,
  pl21, all = T),
  pl20, all = T),
  pl19, all = T),
  pl18, all = T)
premLge$Date = as.POSIXct(premLge$Date, format = "%m/%d/%y")

premLge = premLge %>%
  select(Date, HomeTeam, AwayTeam, FTHG, FTAG, FTR)
```

Next, I found the overall league scoring average over the last 5 seasons, as well as each teams individual home and away scoring averages.

```
lgAvg = data.frame(avgHG = mean(premLge$FTHG), avgAG = mean(premLge$FTAG))

tmAvgH = premLge %>%
  group_by(HomeTeam) %>%
  summarize(GoalsFor = mean(FTHG),
            GoalsAgainst = mean(FTAG))
tmAvgA = premLge %>%
  group_by(AwayTeam) %>%
  summarize(GoalsFor = mean(FTAG),
            GoalsAgainst = mean(FTHG))
tmAvgH
```

```
## # A tibble: 28 x 3
##    HomeTeam       GoalsFor GoalsAgainst
##    <chr>             <dbl>        <dbl>
##  1 Arsenal            2.01        1.03
##  2 Aston Villa        1.40        1.51
##  3 Bournemouth        1.37        1.49
##  4 Brentford          1.16        1.11
##  5 Brighton           1.09        1.32
##  6 Burnley            1.01        1.31
##  7 Cardiff            1.11        2
##  8 Chelsea            1.76        0.884
##  9 Crystal Palace     1.16        1.25
## 10 Everton            1.4         1.23
## # ... with 18 more rows
```

From these averages, I create a power index to rank a teams goals for and goals against rating against the league average. For offense, anything above 1 is better than the league average (i.e. scores more), and for defense anything less than 1 is better (i.e. holds teams to less goals scored).

```
tmHPwr = tmAvgH %>%
  mutate(#Team = HomeTeam,
         OPwrH = GoalsFor/lgAvg$avgHG,
         DPwrH = GoalsAgainst/lgAvg$avgAG)

tmAPwr = tmAvgA %>%
  mutate(#Team = AwayTeam,
         OPwrA = GoalsFor/lgAvg$avgAG,
         DPwrA = GoalsAgainst/lgAvg$avgHG)
tmHPwr
```

```
## # A tibble: 28 x 5
##    HomeTeam       GoalsFor GoalsAgainst OPwrH DPwrH
##    <chr>             <dbl>        <dbl> <dbl> <dbl>
##  1 Arsenal            2.01        1.03  1.34  0.825
##  2 Aston Villa        1.40        1.51  0.938 1.21
##  3 Bournemouth        1.37        1.49  0.915 1.19
##  4 Brentford          1.16        1.11  0.774 0.884
##  5 Brighton           1.09        1.32  0.732 1.05
##  6 Burnley            1.01        1.31  0.675 1.04
##  7 Cardiff            1.11        2     0.739 1.60
##  8 Chelsea            1.76        0.884 1.17  0.707
##  9 Crystal Palace     1.16        1.25  0.774 1.00
## 10 Everton            1.4         1.23  0.936 0.985
## # ... with 18 more rows
```

Next, I created a list of this years current PL teams, substituting Norwhich for Nottingham since Nottingham hasn't been to the Premier League in over 20 years. Norwhich is a team relegated from last season, whose skill level is comparable to Norwhich.

```r
teams = c(unique(tmHPwr$HomeTeam))
plTeams = c('Arsenal', 'Aston Villa', 'Bournemouth', 'Brentford',
            'Brighton', 'Chelsea', 'Crystal Palace', 'Everton',
            'Fulham', 'Leeds', 'Leicester', 'Liverpool',
            'Man City', 'Man United', 'Newcastle', 'Southampton',
            'Tottenham', 'West Ham', 'Wolves', 'Norwich')
# Substituting Norwich for Nottingham

tmHPwrPL = tmHPwr %>%
  filter(HomeTeam %in% plTeams) %>%
  select(HomeTeam, OPwrH, DPwrH)

tmAPwrPL = tmAPwr %>%
  filter(AwayTeam %in% plTeams) %>%
  select(AwayTeam, OPwrA, DPwrA)
```

Now I will load in the matches for this upcoming season, and bind each matchup to the respective teams home and away scoring power rankings.

```r
matches = read.csv('/Users/Shane/OneDrive/Desktop/Project-Work-main/Git/PremiereLeague/PLMatches.csv')

dfT = data.frame(matrix(ncol = 6, nrow = 0))
colnames(dfT) = c('HomeTeam', 'AwayTeam', 'OPwrA', 'DPwrA', 'OPwrH', 'DPwrH')

for (i in plTeams) {
  m = matches %>%
    filter(HomeTeam == i)
  df1 = merge(m, tmHPwrPL, by = 'HomeTeam')
  dfT1 = merge(df1, tmAPwrPL, by = 'AwayTeam')

  m2 = matches %>%
    filter(AwayTeam == i)
  df2 = merge(m2, tmAPwrPL, by = 'AwayTeam')
  dfT2 = merge(df2, tmHPwrPL, by = 'HomeTeam')

  dfT = rbind(dfT, dfT2, dfT1)
}

dfT = dfT %>%
  distinct()
```

Now I will estimate the scores of each mathcup based off of each teams power rankings.

- Estimated Home Goals = Home O Power Ranking * Away D Power Ranking * average goals scored in league
- Estimated Away Goals = Away O Power Ranking * Home D Power Ranking * average goals scored in league

```r
scores = dfT %>%
  mutate(EstHGoals = OPwrH * DPwrA * lgAvg$avgHG,
         EstAGoals = DPwrH * OPwrA * lgAvg$avgAG)
scoreEst = scores %>%
  select(HomeTeam, AwayTeam, EstHGoals, EstAGoals)

fixtures = scoreEst %>%
  select(home = HomeTeam, away = AwayTeam)

head(fixtures)
```

```
##             home    away
## 1    Aston Villa Arsenal
## 2    Bournemouth Arsenal
## 3      Brentford Arsenal
## 4       Brighton Arsenal
## 5        Chelsea Arsenal
## 6 Crystal Palace Arsenal
```

In the world of soccer, it is commonly accepted that the amount of goals scored comes from a poisson distribution. From this, I take the expected goals for each match from above, and use that value as the average in a poisson distribution. This allows me to find the probabilities of each team scoring anywhere from 0-7 goals against each other.

```
maxGoals = 7

allPredictions = map2_df(
  scoreEst$EstHGoals, scoreEst$EstAGoals,
  function(lambdaH, lambdaA, maxGoals) {
    hgoalProb = dpois(0:maxGoals, lambdaH) %>%  `names<-`(0:maxGoals)
    agoalProb = dpois(0:maxGoals, lambdaA) %>%  `names<-`(0:maxGoals)
    outer(hgoalProb, agoalProb) %>%
      as.data.frame() %>%
      gather() %>%
      rownames_to_column('row') %>%
      mutate(hgoal = as.numeric(row) %% (maxGoals + 1) - 1) %>%
      mutate(hgoal = case_when(hgoal < 0 ~ maxGoals, TRUE ~ hgoal),
             agoal = as.numeric(key)) %>%
      select(sample_hgoal = hgoal, sample_agoal = agoal, prob = value)
  },
  maxGoals) %>%
  cbind(fixtures[rep(seq_len(nrow(fixtures)), each = (maxGoals+1)^2), ], .) %>%
  group_by(home, away) %>%
  mutate( prob = prob/sum(prob) ) %>%
  ungroup()

head(allPredictions, 8)
```

```
## # A tibble: 8 x 5
##   home        away    sample_hgoal sample_agoal      prob
##   <chr>       <chr>          <dbl>        <dbl>     <dbl>
## 1 Aston Villa Arsenal            0            0 0.0499
## 2 Aston Villa Arsenal            1            0 0.0685
## 3 Aston Villa Arsenal            2            0 0.0470
## 4 Aston Villa Arsenal            3            0 0.0215
## 5 Aston Villa Arsenal            4            0 0.00738
## 6 Aston Villa Arsenal            5            0 0.00202
## 7 Aston Villa Arsenal            6            0 0.000463
## 8 Aston Villa Arsenal            7            0 0.0000908
```

Since this data frame is very large, I created a nested list of each individual matchup and the probabilities of potential scores

```
nestedProb = allPredictions %>%
  nest(probabilities = c(sample_hgoal, sample_agoal, prob))

nestedProb2 = nestedProb %>%
  mutate(sampled_result = map(probabilities, sample_n, 1, weight = prob)) %>%
  select(-probabilities) %>%
  unnest(cols = c(sampled_result))

head(nestedProb2)
```

```
## # A tibble: 6 x 5
##   home           away    sample_hgoal sample_agoal   prob
##   <chr>          <chr>          <dbl>        <dbl>  <dbl>
## 1 Aston Villa    Arsenal            2            1 0.0764
## 2 Bournemouth    Arsenal            0            1 0.0846
## 3 Brentford      Arsenal            0            2 0.0695
## 4 Brighton       Arsenal            0            2 0.0835
## 5 Chelsea        Arsenal            4            0 0.0252
## 6 Crystal Palace Arsenal            2            1 0.0723
```

nestedProb2 is to display the score that is most likely to happen between two teams.

Now, I use a monte carlo method to simulate the entirety of the Premier League season n times. This allows the model to simulate matches over and over to try to predict the overall most likely match results over an entire season. While doing this, I assigned the match results to the Premier League Format of 3 points for a win, 1 for a tie, and 0 for a loss.

```
n = 1000
matchSims = rerun(n, nestedProb %>%
                    mutate(sampled_result = map(probabilities, sample_n, 1, weight = prob)) %>%
                    select(-probabilities) %>%
                    unnest(cols = c(sampled_result)) %>%
                    select(-prob) %>%
                    pivot_longer(c(home, away), names_to = 'location', values_to = 'team') %>%
                    mutate(points = case_when(location == 'home' & sample_hgoal > sample_agoal ~ 3,
                                              location == 'away' & sample_agoal > sample_hgoal ~ 3,
                                              sample_hgoal == sample_agoal ~ 1,
                                              TRUE ~ 0)) %>%
                    mutate(gd = case_when(location == 'home' ~ sample_hgoal - sample_agoal,
                                          location == 'away' ~ sample_agoal - sample_hgoal)))
#n = 10
#matchSims = rerun(n, nestedProb %>%
#                    mutate(sampled_result = map(probabilities, #sample_n, 1, weight = prob)) %>%
#                    select(-probabilities) %>%
#                    unnest(cols = c(sampled_result)) %>%
#                    select(-prob) %>%
#                    pivot_longer(c(home, away), names_to = #'location', values_to = 'team') %>%
#                     mutate(points = case_when(
#                      location == 'home' & sample_hgoal > #sample_agoal ~ 3,
#                       location == 'away' & sample_agoal > #sample_hgoal ~ 3,
#                       sample_hgoal == sample_agoal ~ 1,
#                       TRUE ~ 0
#                     )) %>%
#                     mutate(gd = case_when(
#                       location == 'home' ~ sample_hgoal - #sample_agoal,
#                       location == 'away' ~ sample_agoal - #sample_hgoal)))
```

From these simulations, I then take the average points earned for each team across a season, as well as their average goal differential to create a simulated Premier League table.

```
tableSim = matchSims %>%
  bind_rows() %>%
  group_by(team) %>%
  summarize(points = sum(points)/n,
            goalDiff = sum(gd)/n)

tableSim['team'][tableSim['team'] == 'Norwich'] = 'Nottingham'

finalTable = tableSim[order(-tableSim$points), ]
finalTable
```

```
## # A tibble: 20 x 3
##    team           points goalDiff
##    <chr>           <dbl>    <dbl>
##  1 Man City         91.6    70.4
##  2 Liverpool        84.4    53.7
##  3 Tottenham        69.8    27.7
##  4 Chelsea          69.4    25.9
##  5 Man United       67.0    22.8
##  6 Arsenal          63.0    16.6
##  7 Leicester        58.3     8.17
##  8 West Ham         51.5    -3.34
##  9 Wolves           50.3    -3.08
## 10 Aston Villa      48.8    -6.86
## 11 Brentford        48.1    -8.47
## 12 Everton          47.2    -9.35
## 13 Crystal Palace   45.7   -11.3
## 14 Leeds            43.8   -16.3
## 15 Newcastle        43.7   -14.4
## 16 Brighton         42.6   -14.8
## 17 Bournemouth      41.3   -20.1
## 18 Southampton      40.9   -20.5
## 19 Fulham           29.1   -38.1
## 20 Nottingham       19.0   -58.7
```

From this method, the top 4 teams are expected to be:

- Man City
- Liverpool

- Tottenham
- Chelsea