Premier League Analysis

Contributors: Mina Sonbol, Faisal Shahid

1. Introduction

1.1 The challenges

The English Premier League is the oldest association football league in the world, established in the late 19th century, the competition has been taking place every year since 1888-1889. According to wikipedia, "the English Premier League is the top-tier division in the English football league system"[1]. Each year there are twenty teams contesting the league, where all teams face each other twice, and are ranked by the total points they accrue throughout the season[2].
Beside winning the competition, the top 6 teams qualify to continental competitions the following season (UEFA Champions League and UEFA Europa League), providing additional income sources and more exposure, not to mention prestige.
Finally, the bottom three teams are relegated to the second division, and are replaced by the top 3 teams from that division. The football business has seen immense growth in recent decades, driven by increased coverage and spending, making the mere participation in the league a very lucrative proposition, let alone winning it, something reflected in the yearly relegation battles

Accordingly, we can define these three challenges taking place as such:

1) Win the league → Rank = 1
2) Qualify for continental competitions → Rank > 7
3) Avoid relegation → Rank > 18

Defining the challenges will give a glimpse into what is going on in the mind of a team's leadership at the beginning of each new season, and how they develop plans for future seasons.

Initially, our goal was to analyse each team and provide suggestions on what it needs to do to achieve better results. However, after working on several ideas we found that it would take a lot of time to analyse each team individually. At that point, It became apparent to us that a better approach would be to develop a method that can help any team measure its performances and identify weak areas that it can address in the future.

1.2 The dataset

Now, a quick work on the data. We will be using data from the understat[3] website. The site provides data for the 6 European leagues (English, Spanish, Italian, German, French, & Russian), starting from the 2014/15 season. The site's data is accessible through the understatr[4] R package. It provides league, team, & player data. In our study we will only focus

on team data, accessible through the get_league_teams_stats() function. Appendix 1 provides a guide for the data collected using this function. We also decided to exclude the 2019/20 season.

2. The Path

2.1 The winning formula

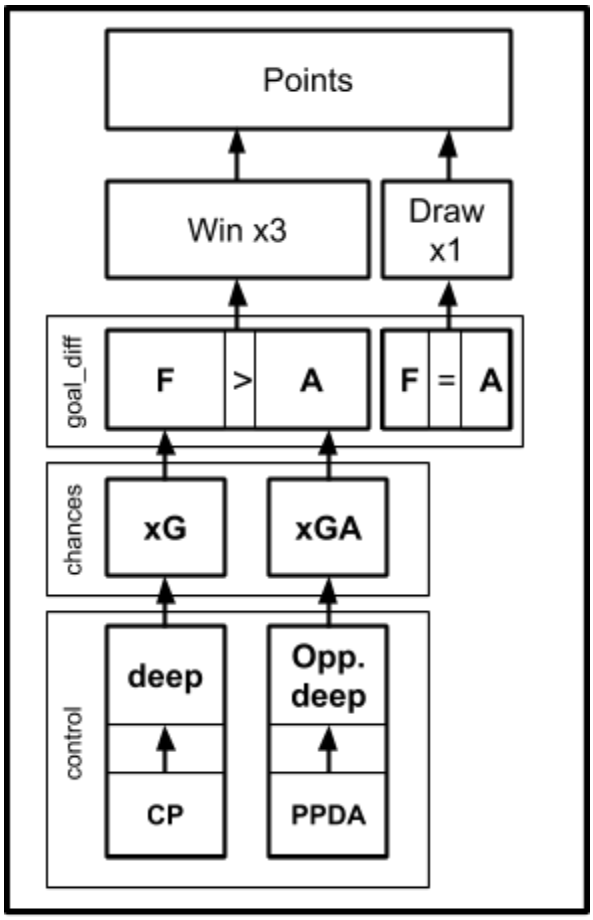


Figure 1: Points Flowchart

Having identified the challenges and 'what' is at stake, we move to the next part; that is: 'how' a team can accomplish its goals. As mentioned earlier, teams are ranked by points total, accordingly total points will be the deciding factor. Figure 1 describes the progression of factors that contribute to accumulating more points.
Teams are awarded 3 points for winning a match, 1 for a draw, and 0 for a loss. Therefore, a team wanting to challenge for the title needs to win most of its matches, to draw as little as possible, and to avoid losses. Winning a match is straightforward, team A needs to score more goals than team B, in other words, goals decide the winner. Not only that, they are the most exciting part of the game, associated with huge celebrations - sometimes amounting to mass hysteria - among players and fans. This begs a question, why is that? After all, you don't see a basketball game stopping after each scored bucket. To answer this question we need to lock beyond the technical or tactical aspects that are associated with a goal. Football is unique among sports in its low-scoring nature, something that gives goals an almost mystical nature. So how do we measure goals, or better yet, how do we predict them? Using the past to predict the future is usually a good place to start. We can predict how many many goals a team might score in the next match using the number of goals it has scored in the previous match. However, this method does not account for important factors, such as the situation from which a goal was scored (open-play or a set-piece), the distance between the point where the ball was struck from to the goal line, whether the ball was shot by the leg (which leg?) or headed. This is where the expected goals[5] statistic (xG) comes in. It describes the quality of a chance by accounting for the aforementioned factors.
This raises another question, how does a team increase its xG? The answer is simple, by keeping the ball, or more specifically, by keeping it close to your opponent's goal. This is exactly what the 'deep' and 'completed passes' (CP) statistics tell us; CP is the total number of passes

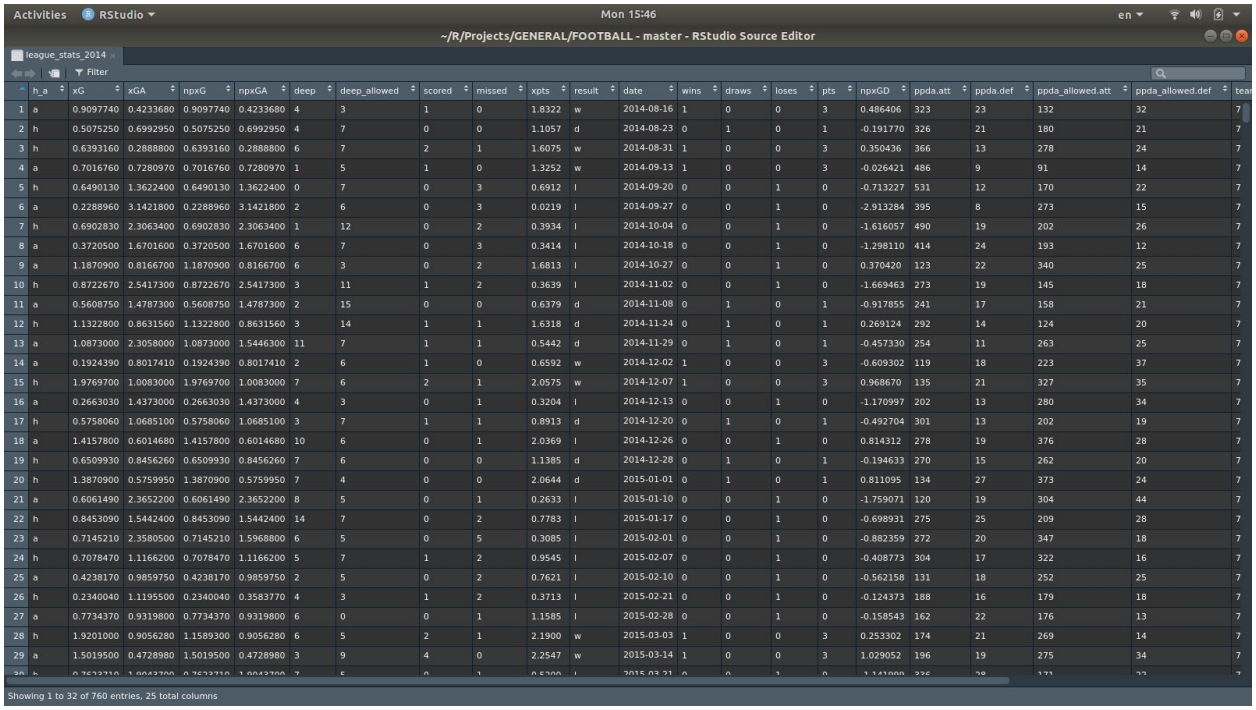
a team successfully completes during a match, while deep is the number of passes completed within 20 yards from the opponents' goal.
Now that we have defined what a team needs to do to score more goals, we look at the other side of the equation, avoiding conceding them. Much like with scoring goals, xGA describes the quality of the chances a team concedes. Teams need to keep this number low. Also, if your opponent has the ball near to your goal, they are more likely to score, so your team needs to keep the 'deep_allowed' statistic low. Finally, there is the 'passes per defensive action' statistic (PPDA)[6], which explains how many passes your opponent completes without contention; a high value for this statistic means that the opponent is having more success when in possession of the ball.
In conclusion, we can say that for a team to amass as many points as it can it needs to do the following: control the game by keeping the ball more than its opponent. It can do so by completing more accurate passing (CP), preferably nearer to its opponent goal (deep). This will open the door to creating more chances to score (xG), increasing the probability of scoring goals and winning games. Simultaneously, it needs to limit the opponents' possession by trying to win the ball back as fast as possible (PPDA), and as far from its goal as possible (deep_allowed). Failing to do so will result in the opponent creating better chances (xGA), which it could turn into goals against, thus limiting its chances of winning.

2.2 Points and other important factors

To develop this flowchart, we used our intuitive understanding of the game, assisted by the data collected from understat. To download the data you need to call the get_league_teams_stats() function, which takes both a league_name and year as parameters. A call might look something like this:

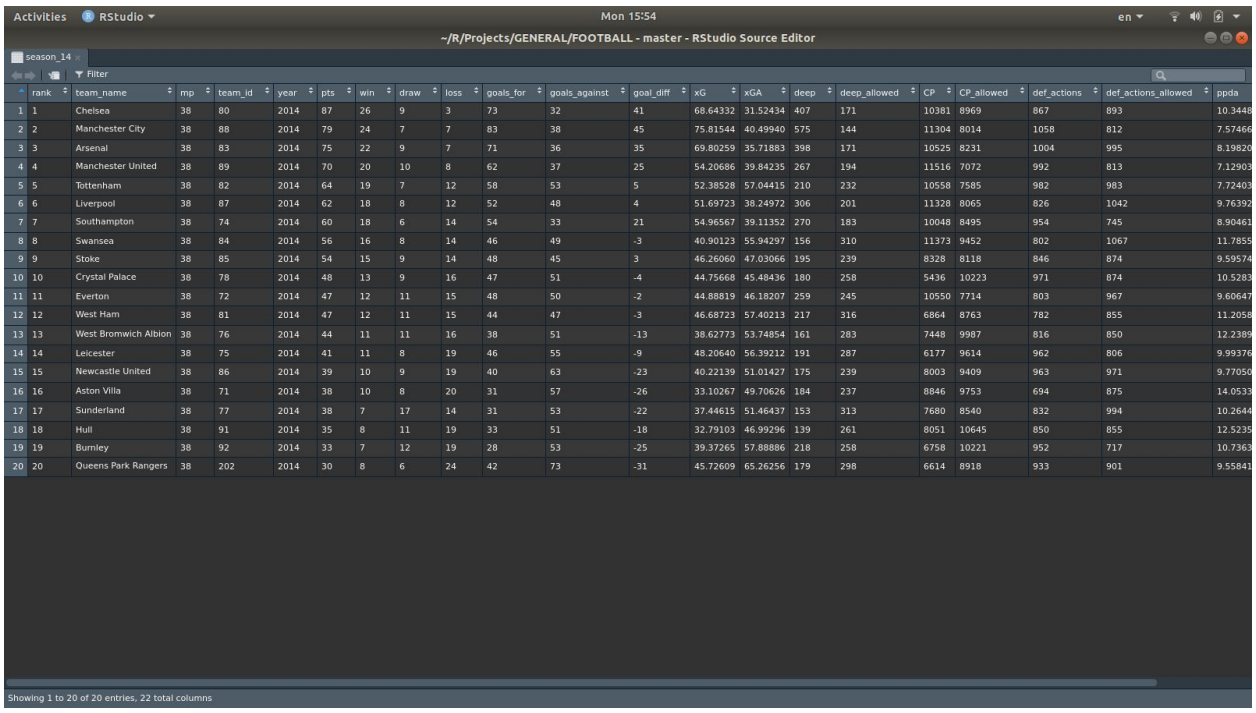
```
league_stats_2014 <- get_league_teams_stats(league_name = "EPL", year = 2014)
```

This call will extract statistics for all matches played during the 2014/15 English Premier League season. The output is a tible object, with 25 columns describing different features of the match, and 760 rows for all matches played throughout the season (20 teams * 38 matches per team). It looks like this:



	h_a	xG	xGA	npxG	npxGA	deep	deep_allowed	scored	missed	xpts	result	date	wins	draws	loses	pts	npxGD	ppda.att	ppda.def	ppda_allowed.att	ppda_allowed.def	tea
1	a	0.9097740	0.4233680	0.9097740	0.4233680	4	3	1	0	1.8322	w	2014-08-16	1	0	0	3	0.486406	323	23	132	32	7
2	h	0.5075250	0.6992950	0.5075250	0.6992950	4	7	0	0	1.1057	d	2014-08-23	0	1	0	1	-0.191770	326	21	180	21	7
3	h	0.6393160	0.2888800	0.6393160	0.2888800	6	7	2	1	1.6075	w	2014-08-31	1	0	0	3	0.350436	366	13	278	24	7
4	a	0.7016760	0.7280970	0.7016760	0.7280970	1	5	1	0	1.3252	w	2014-09-13	1	0	0	3	-0.026421	486	9	91	14	7
5	h	0.6490130	1.3622400	0.6490130	1.3622400	0	7	0	3	0.6912	l	2014-09-20	0	0	1	0	-0.713227	531	12	170	22	7
6	a	0.2288960	3.1421800	0.2288960	3.1421800	2	6	0	3	0.0219	l	2014-09-27	0	0	1	0	-2.913284	395	8	273	15	7
7	h	0.6902830	2.3063400	0.6902830	2.3063400	1	12	0	2	0.3934	l	2014-10-04	0	0	1	0	-1.616057	490	19	202	26	7
8	a	0.3720500	1.6701600	0.3720500	1.6701600	6	7	0	3	0.3414	l	2014-10-18	0	0	1	0	-1.298110	414	24	193	12	7
9	a	1.1870900	0.8166700	1.1870900	0.8166700	6	3	0	2	1.6813	l	2014-10-27	0	0	1	0	0.370420	123	22	340	25	7
10	h	0.8722670	2.5417300	0.8722670	2.5417300	3	11	1	2	0.3639	l	2014-11-02	0	0	1	0	-1.669463	273	19	145	18	7
11	a	0.5608750	1.4787300	0.5608750	1.4787300	2	15	0	0	0.6379	d	2014-11-08	0	1	0	1	-0.917855	241	17	158	21	7
12	h	1.1322800	0.8631560	1.1322800	0.8631560	3	14	1	1	1.6318	d	2014-11-24	0	1	0	1	0.269124	292	14	124	20	7
13	a	1.0873000	2.3058000	1.0873000	1.5446300	11	7	1	1	0.5442	d	2014-11-29	0	1	0	1	-0.457330	254	11	263	25	7
14	a	0.1924390	0.8017410	0.1924390	0.8017410	2	6	1	0	0.6592	w	2014-12-02	1	0	0	3	-0.609302	119	18	223	37	7
15	h	1.9769700	1.0083000	1.9769700	1.0083000	7	6	2	1	2.0575	w	2014-12-07	1	0	0	3	0.968670	135	21	327	35	7
16	a	0.2663030	1.4373000	0.2663030	1.4373000	4	3	0	1	0.3204	l	2014-12-13	0	0	1	0	-1.170997	202	13	280	34	7
17	h	0.5758060	1.0685100	0.5758060	1.0685100	3	7	1	1	0.8913	d	2014-12-20	0	1	0	1	-0.492704	301	13	202	19	7
18	a	1.4157800	0.6014680	1.4157800	0.6014680	10	6	0	1	2.0369	l	2014-12-26	0	0	1	0	0.814312	278	19	376	28	7
19	h	0.6509930	0.8456260	0.6509930	0.8456260	7	6	0	0	1.1385	d	2014-12-28	0	1	0	1	-0.194633	270	15	262	20	7
20	h	1.3870900	0.5759950	1.3870900	0.5759950	7	4	0	0	2.0644	d	2015-01-01	0	1	0	1	0.811095	134	27	373	24	7
21	a	0.6061490	2.3652200	0.6061490	2.3652200	8	5	0	1	0.2633	l	2015-01-10	0	0	1	0	-1.759071	120	19	304	44	7
22	h	0.8453090	1.5442400	0.8453090	1.5442400	14	7	0	2	0.7783	l	2015-01-17	0	0	1	0	-0.698931	275	25	209	28	7
23	a	0.7145210	2.3580500	0.7145210	1.5968800	6	5	0	5	0.3085	l	2015-02-01	0	0	1	0	-0.882359	272	20	347	18	7
24	h	0.7078470	1.1166200	0.7078470	1.1166200	5	7	1	2	0.9545	l	2015-02-07	0	0	1	0	-0.408773	304	17	322	16	7
25	a	0.4238170	0.9859750	0.4238170	0.9859750	2	5	0	2	0.7621	l	2015-02-10	0	0	1	0	-0.562158	131	18	252	25	7
26	h	0.2340040	1.1195500	0.2340040	0.3583770	4	3	1	2	0.3713	l	2015-02-21	0	0	1	0	-0.124373	188	16	179	18	7
27	a	0.7734370	0.9319800	0.7734370	0.9319800	6	0	0	1	1.1585	l	2015-02-28	0	0	1	0	-0.158543	162	22	176	13	7
28	h	1.9201000	0.9056280	1.1589300	0.9056280	6	5	2	1	2.1900	w	2015-03-03	1	0	0	3	0.253302	174	21	269	14	7
29	a	1.5019500	0.4728980	1.5019500	0.4728980	3	9	4	0	2.2547	w	2015-03-14	1	0	0	3	1.029052	196	19	275	34	7

Our analysis centers around total points, being the deciding criteria for the various challenges. Therefore, we had to transform this table into an overall ranking, and sum the total points accumulated by each team. We achieved this by creating the get_season() function, which uses groupby team to calculate and aggregate sum of all statistics:



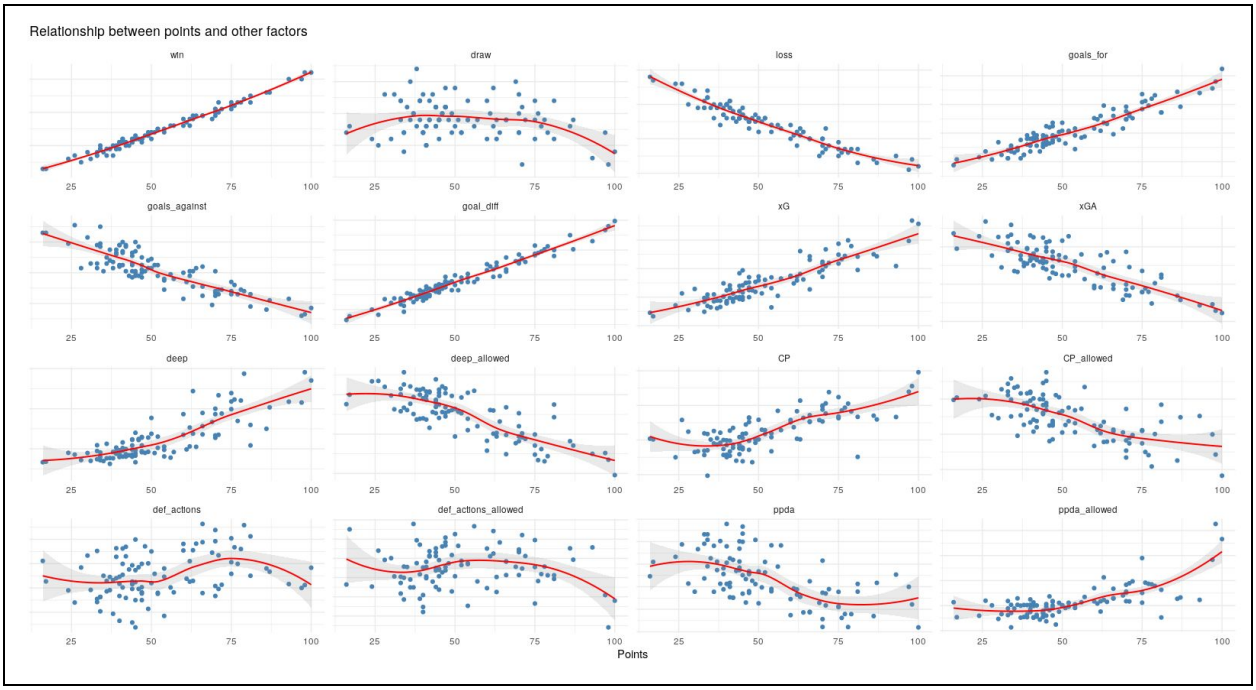
	rank	team_name	mp	team_id	year	pts	win	draw	loss	goals_for	goals_against	goal_diff	xG	xGA	deep	deep_allowed	CP	CP_allowed	def_actions	def_actions_allowed	ppda
1	1	Chelsea	38	80	2014	87	26	9	3	73	32	41	68.64332	31.52434	407	171	10381	8969	867	893	10.3448
2	2	Manchester City	38	88	2014	79	24	7	7	83	38	45	75.81544	40.49940	575	144	11304	8014	1058	812	7.57466
3	3	Arsenal	38	83	2014	75	22	9	7	71	36	35	69.80259	35.71883	398	171	10525	8231	1004	995	8.19820
4	4	Manchester United	38	89	2014	70	20	10	8	62	37	25	54.20686	39.84235	267	194	11516	7072	992	813	7.12903
5	5	Tottenham	38	82	2014	64	19	7	12	58	53	5	52.38528	57.04415	210	232	10558	7585	982	983	7.72403
6	6	Liverpool	38	87	2014	62	18	8	12	52	48	4	51.69723	38.24972	306	201	11328	8065	826	1042	9.76392
7	7	Southampton	38	74	2014	60	18	6	14	54	33	21	54.96567	39.11352	270	183	10048	8495	954	745	8.90461
8	8	Swansea	38	84	2014	56	16	8	14	46	49	-3	40.90123	55.94297	156	310	11373	9452	802	1067	11.7855
9	9	Stoke	38	85	2014	54	15	9	14	48	45	3	46.26060	47.03066	195	239	8328	8118	846	874	9.59574
10	10	Crystal Palace	38	78	2014	48	13	9	16	47	51	-4	44.75668	45.48436	180	258	5436	10223	971	874	10.5283
11	11	Everton	38	72	2014	47	12	11	15	48	50	-2	44.88819	46.18207	259	245	10550	7714	803	967	9.60647
12	12	West Ham	38	81	2014	47	12	11	15	44	47	-3	46.68723	57.40213	217	316	6864	8763	782	855	11.2058
13	13	West Bromwich Albion	38	76	2014	44	11	11	16	38	51	-13	38.62773	53.74854	161	283	7448	9987	816	850	12.2389
14	14	Leicester	38	75	2014	41	11	8	19	46	55	-9	48.20640	56.39212	191	287	6177	9614	962	806	9.99376
15	15	Newcastle United	38	86	2014	39	10	9	19	40	63	-23	40.22139	51.01427	175	239	8003	9409	963	971	9.77050
16	16	Aston Villa	38	71	2014	38	10	8	20	31	57	-26	33.10267	49.70626	184	237	8846	9753	694	875	14.0533
17	17	Sunderland	38	77	2014	38	7	17	14	31	53	-22	37.44615	51.46437	153	313	7680	8540	832	994	10.2644
18	18	Hull	38	91	2014	35	8	11	19	33	51	-18	32.79103	46.99296	139	261	8051	10645	850	855	12.5235
19	19	Burnley	38	92	2014	33	7	12	19	28	53	-25	39.37265	57.88886	218	258	6758	10221	952	717	10.7363
20	20	Queens Park Rangers	38	202	2014	30	8	6	24	42	73	-31	45.72609	65.26256	179	298	6614	8918	933	901	9.55841

A quick note here, we excluded a few features (such as h_a which tells whether a match was played at home or away). We also used the data itself to create a few important features like PPDA, which does not come as a standalone statistic, but can be calculated using its constituents ($PPDA = \frac{PASSES\ COMPLETED\ BY\ OPPONENT}{DEFENSIVE\ ACTIONS}$).

We constructed a season table for the years 2014 through 2019, and combined them all together in the season_all object.
We then started analysing the season_all object, by looking at relationships between different features. These relationships are explored below.

2.2.1 Scatter plots

The first visualization is a combined scatter plot showing the kind of relationship between points and the other predictors:

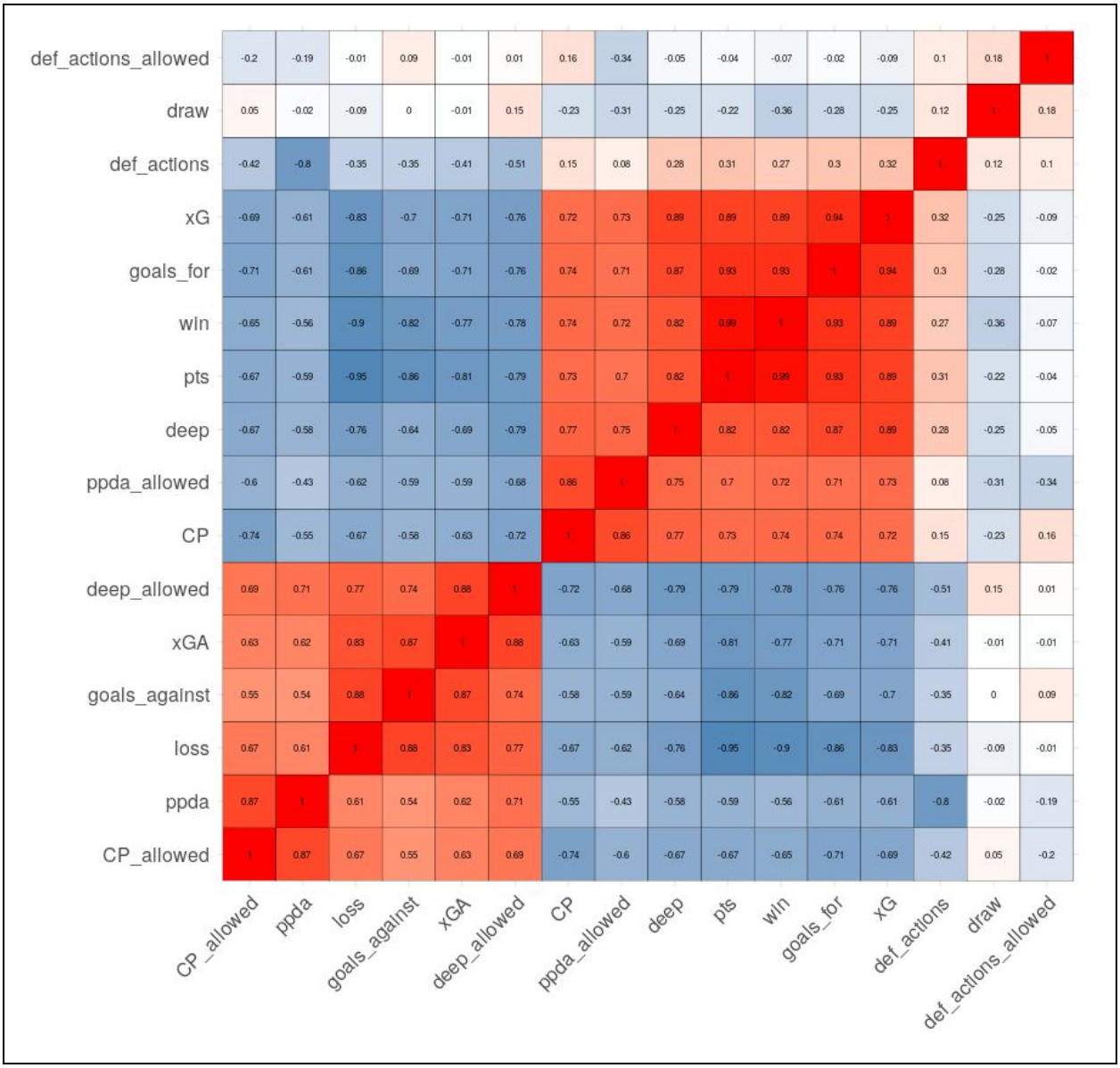


The scatter plots above tell us that most of these relationships are linear in nature, especially so with when it comes to wins and goals. Other relationships are more complex, that is to say, the road from wins/goals to points is straightforward, however, the road to wins/goals is more complex.

2.2.2 Correlation Matrix

Now lets see which features correlate together. By using the ggcorrplot() function, we were able to use hierarchical clustering[7] to order the matrix' columns. As you can see, features describing factors contributing to conceding goals [avgCP_allowed, ppda, loss, goals_against, xGA,

deep_allowed] are grouped together, while features describing factors contributing to scoring goals [avgCP, ppda_allowed, deep, pts, win, goals_for, xG] are grouped separately:



Below, we highlight some correlation coefficients, in a cascading manner, i.e. we only include features that were not already examined to avoid redundancy:

- ‘pts’ correlate the most with ‘win’, with a correlation coefficient of 0.99
- ‘win’ correlate the most with ‘goal_for’, with a correlation coefficient of 0.93
- ‘goal_for’ correlate the most with ‘xG’, with a correlation coefficient of 0.94
- ‘xG’ correlate the most with ‘deep’, with a correlation coefficient of 0.89
- ‘deep’ correlate the most with ‘avg_CP’, with a correlation coefficient of 0.77
- ‘goals_against’ correlate the most with ‘xGA’, with a correlation coefficient of 0.87
- ‘xGA’ correlate the most with ‘deep_allowed’, with a correlation coefficient of 0.88

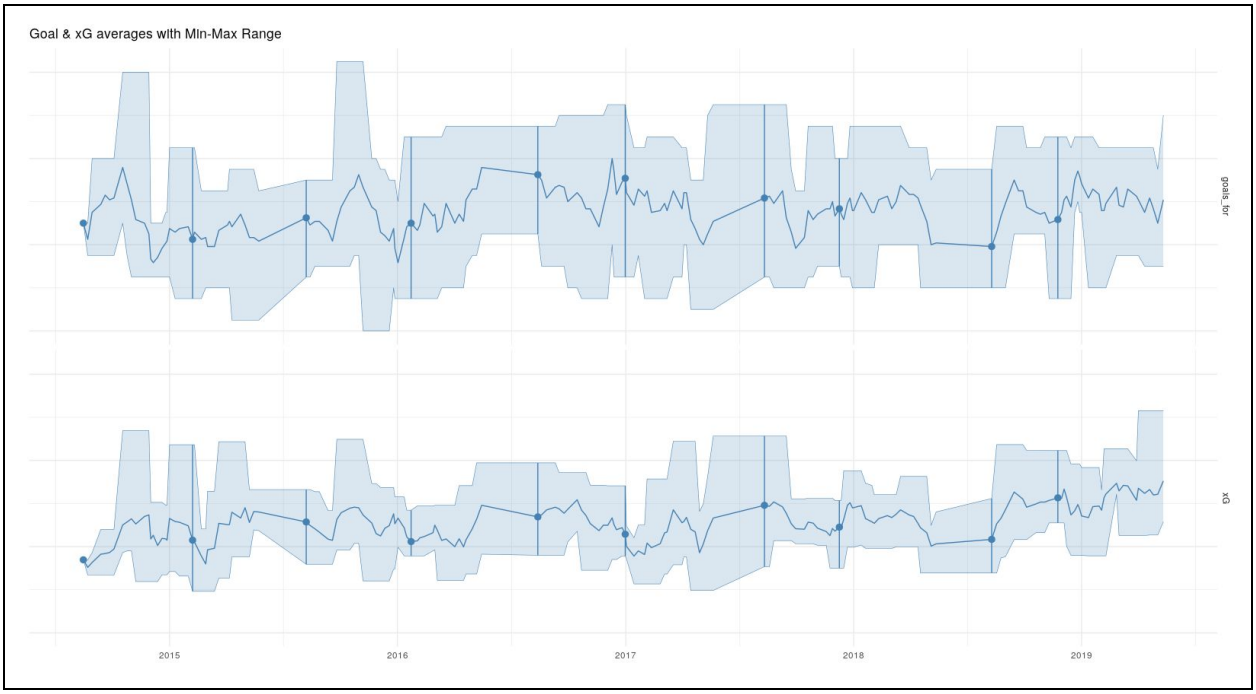
- ‘deep_allowed’ correlate the most with ‘ppda’, with a correlation coefficient of 0.71

The first 2 visuals oriented our focus, and provided the first glimpses into the flowchart. The next thing we look at is the relationship between xG and goals.

2.2.3 The case for xG

Goal Lines

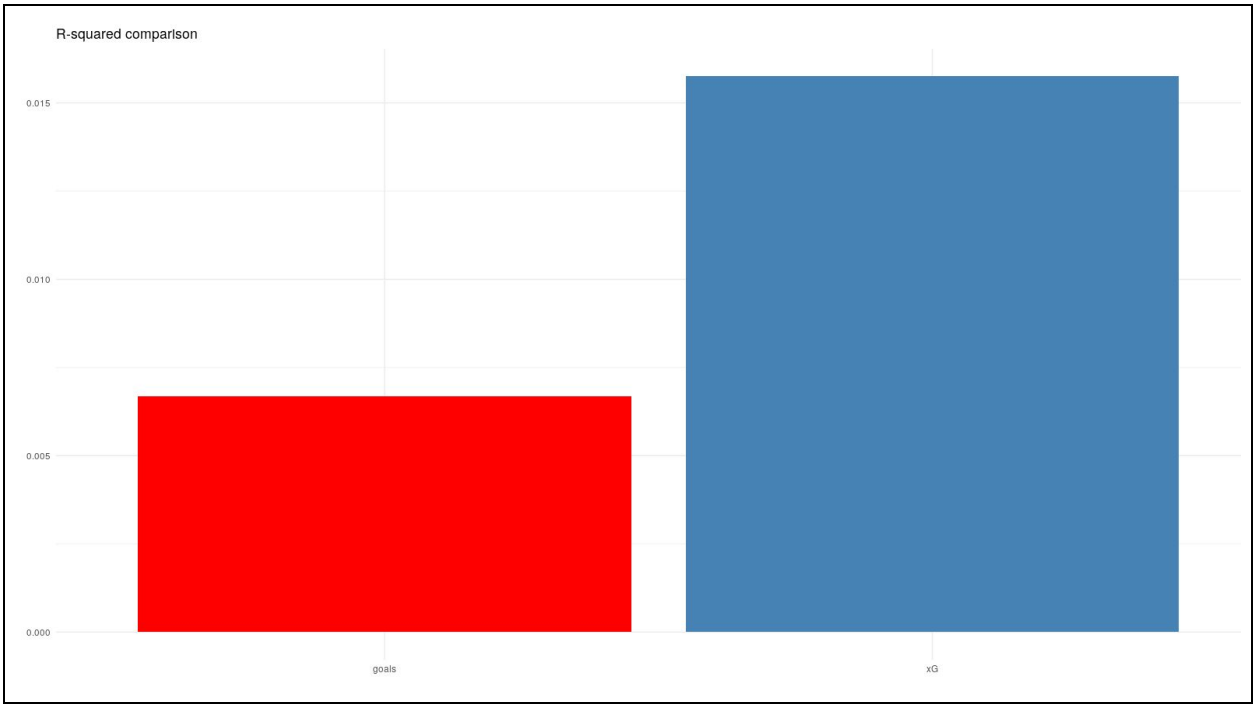
Here we highlight differences between xG and goals_scored. The first chart we look at is a line chart showing the rolling average for xG vs. the rolling average goal scored:



The average goal scored line has more variance than xG, making it noisier, which can ultimately affect its predictive power. The process we used to plot these lines is illustrated below:
1. This is a time series, so we will use the original data which has a date column, instead of the the aggregated season ending table we created for the scatterplot and correlation matrix
2. Next, we calculate 3 aggregated values, all over a 6 match period. The aggregated values are the average, min, max for goals and xG, using the roll_mean(), roll_min(), roll_max() functions
3. This point is associated with the point before. Since we are aggregating over a 6-match period, the first 5 matches will have null aggregate values. We fill those.
4. After that, we gather the resulting dataframe, and plot the lines above

Separate Models

Next, we build and compare two linear regression models, the first using the xG of the previous match to predict how many goals will be scored in the following match, while the second model uses the goals scored in the previous match. Below we show R-squared values of both values:



The process we used to create these models and make this plot is illustrated below:
1- Just like with the previous plot, we use the original data source
2- We collect the team name, match date and goals scored in each match
3- Then, for every observation we add 2 columns: the goals scored and the xG of the previous match. We make sure that these values are team relevant, we do this step on a per-team basis
4- Now that we have the data, we use the lm() method to build 2 linear regression models one using the xG as a predictor the other using goals

We can see that the r-squared value of the xG model is more than double that of the goals model. (Note: these are very simple models, with no predictive power, we just want to illustrate the difference between the predictive power of xG and goals).

Combined Model

We also built a linear regression model using both xG and goals_scored, and it turns out that only xG is statistically significant as indicated by the p-values shown in the below table:

	Coefficient	Std. Error	t-value	p-value

(Intercept)	1.109689	0.037486	29.603	< 2e-16 ***
xG of previous match	0.185639	0.031379	5.916	3.59e-09 ***
Goals scored in previous match	0.003125	0.020890	0.150	0.881

This could be a further indication on why xG might be a better predictor than goals_scored, but the model is too simple to actually be certain.

3. Benchmarks

A quick recap of what we have done so far: First, we identified the challenges within a given premier league season. Second, we identify the factors involved in achieving those challenges, and explore the relationships between them. The next step is building a benchmark that acts as a standard for each challenge.
In the following segment, we take a holistic view at the teams that have competed in the Premier League from 2014 to 2019, and create a 'platonic table' that shows the averages for all features over the 5 year period. This way, teams will have an idea of what they need to do to accomplish a certain challenge.

3.1 Platonic Table

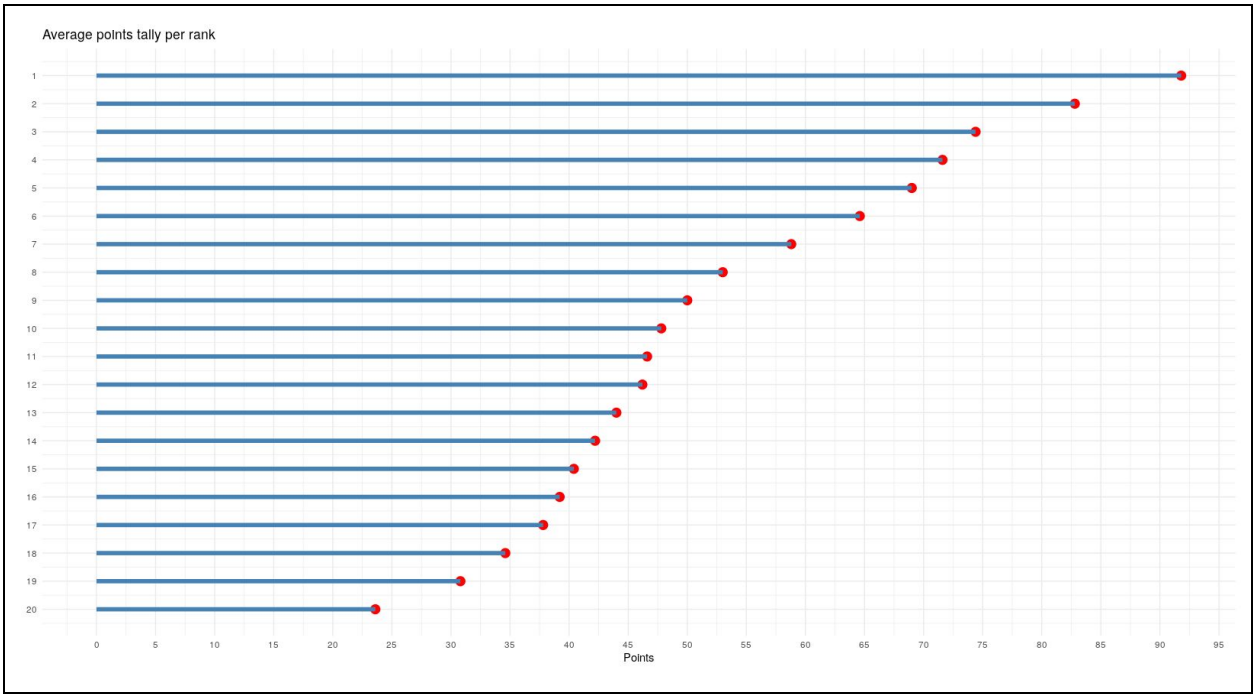
We create the platonic table described above, call it platonicbale. It looks something like this:

	rank	mp	pts	win	draw	loss	goal_diff
1	1	38	91.8	28.6	6.0	3.4	55.2
2	2	38	82.8	25.0	7.8	5.2	48.2
3	3	38	74.4	21.6	9.6	6.8	34.4
4	4	38	71.6	21.0	8.6	8.4	33.0
5	5	38	69.0	20.6	7.2	10.2	19.6
6	6	38	64.6	18.4	9.4	10.2	16.2
7	7	38	58.8	16.2	10.2	11.6	10.2

To better understand this table, consider the first row, describing what it looks like to be a champion. Over the 5 year period considered in our study, the average number of points needed to win the title is 91.8 points, accumulated from 28.6 wins, 6.0 draws, 3.4 loss, and a goal difference of 55.2. Now we visually explore the various features.
To create this table we take the season_all object (created in section 2.2), and aggregate by rank instead of team name.

Points[8]

The first visual is a lollipop bar chart showing the average points for each ranking:



By looking at this visual, we can identify several separations:

1- Between teams ranked 1 & 2, with a 9 point different between them

2- Between teams ranked 1-2, and the rest of the pack, with an 8 point difference between 2nd and 3rd

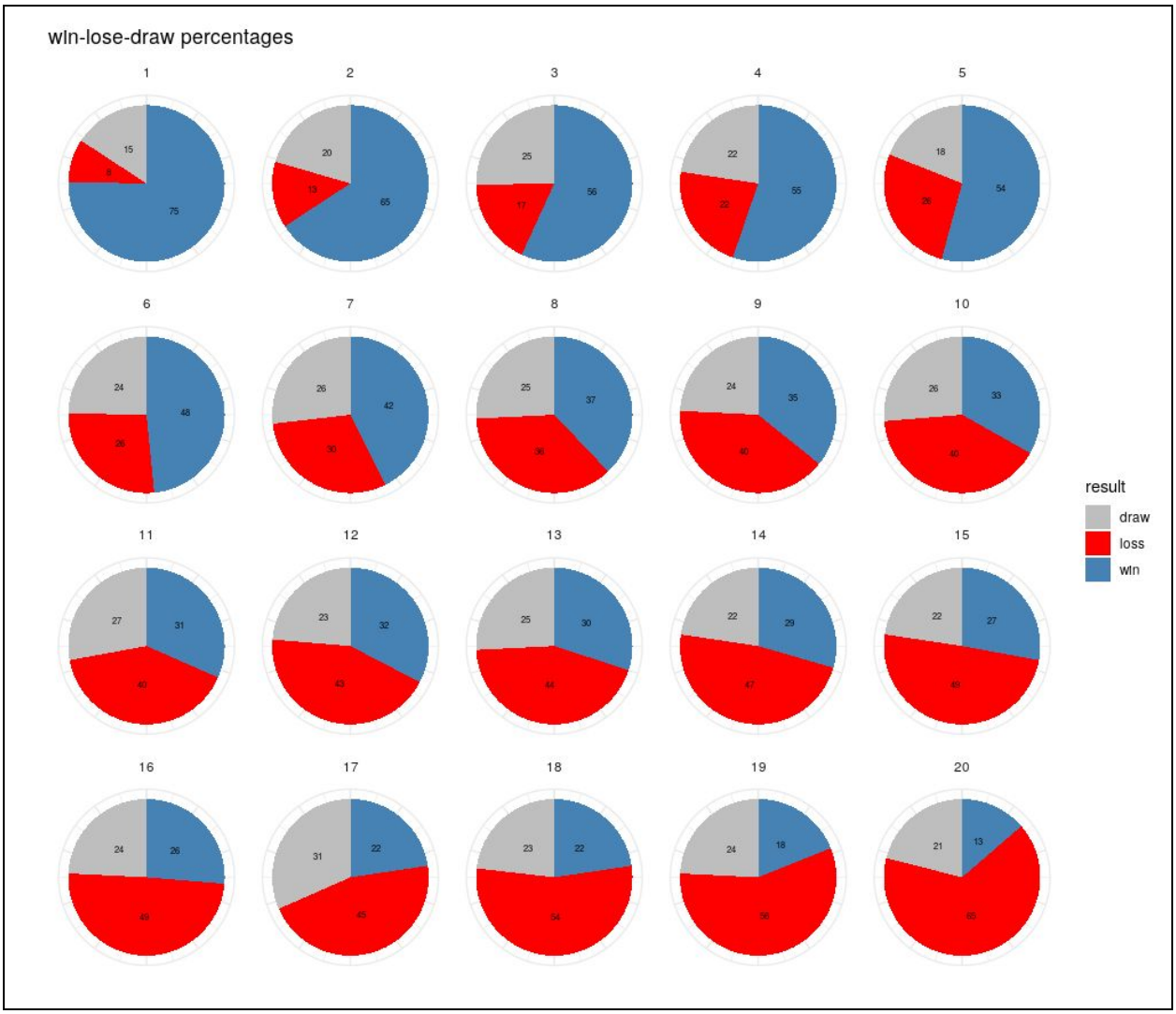
3- Teams ranked 3-6 are grouped together, with an average separation of 3.2 points

4- Teams ranked 8-16 are grouped together, with an average separation of 1.68 points

We used geom_point and geom_segment to construct this visual. The data was used as-is

Win/lose/draw percentage[9]

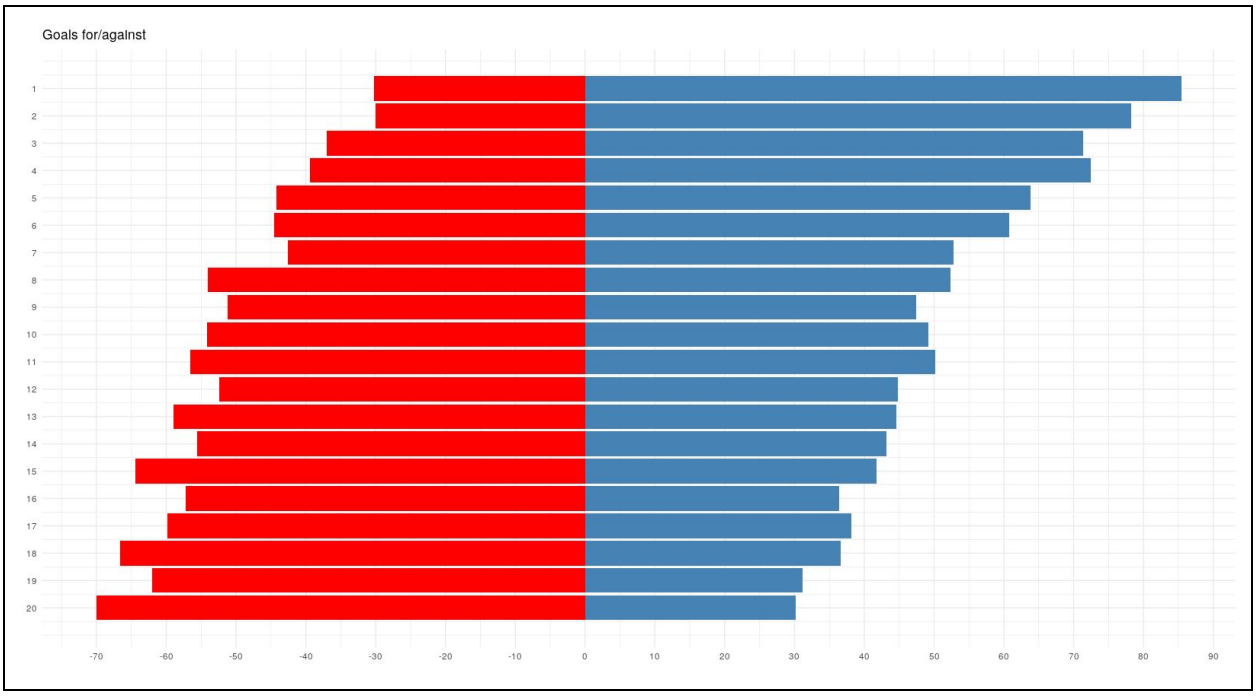
The second visual is a group pie chart showing the win-lose-draw percentages for each rank:



Just like the previous visual, we can identify several separations here. These are similar to the previous so there is no point of listing them again. This visual quickly identifies the win percentage a team needs to achieve a certain goal (ex. 75% win-percentage to win the league).

Goal difference

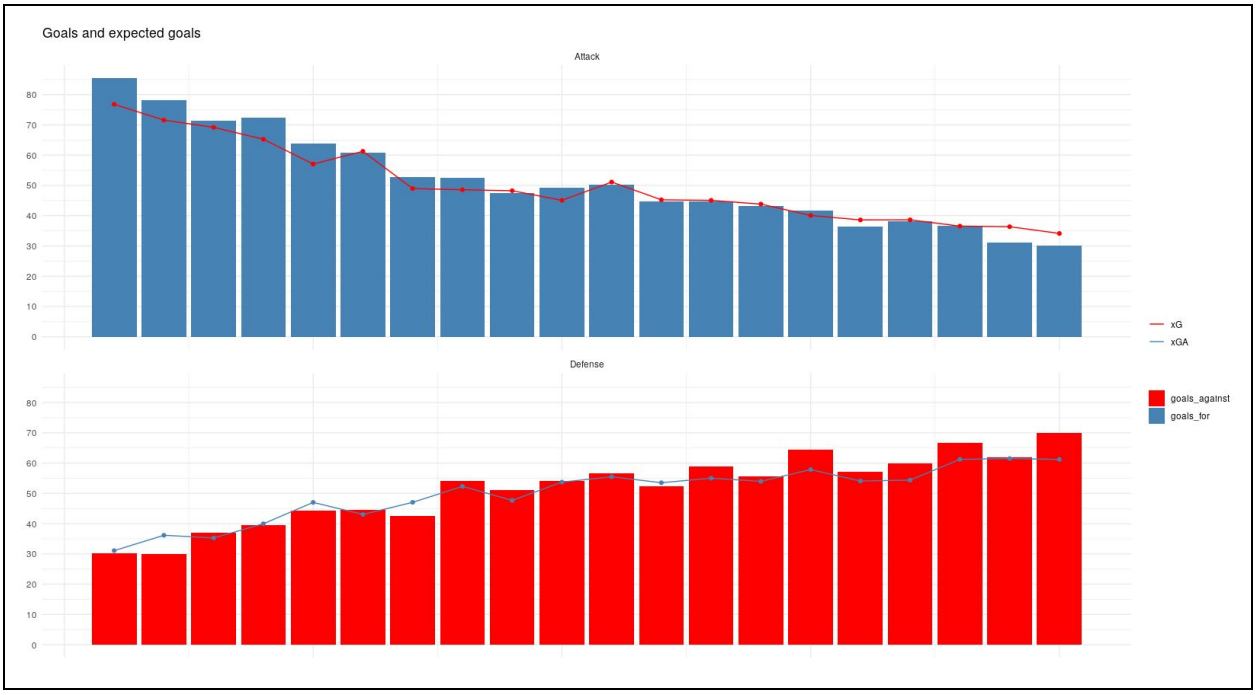
Visual number 3 is the a spread of the goals:



The main takeaway from this plot is the fact that offense is key. You can identify this fact by looking at the top 2 teams, while they concede the same amount of goals, the top ranked team scores 7 more goals on average.
We used geom_col() to produce this plot.

Goals and xG

In the 4th visual we return to goals and expected goals, analysing the relationship between them:



Both on the offensive and the defensive side, teams outscoring their expected goals values usually fare better. This could be an indication of the role of luck in sports.
To construct this visual we had to make several transformations on the platonicbale. First we gather on the four plotted stats (goals_for, goals_against, xG, xGA), then we filter on rows that have either goals_for & xG, or goals_against & xGA only. The bars are created using geom_bar, the lines using geom_line, and facet_col for the separation.

Attitude and work

The final visualization combines together several features. Specifically, deep passes, completed passes, xG and PPDA. The idea here was to capture the attitude and workrate of teams and combine those two factors with chance quality (xG). By attitude we mean how active a team is during possession, we explain this by Active Possession

$$Active\ Possession\ =\ \frac{Deep}{Completed\ Passes}$$

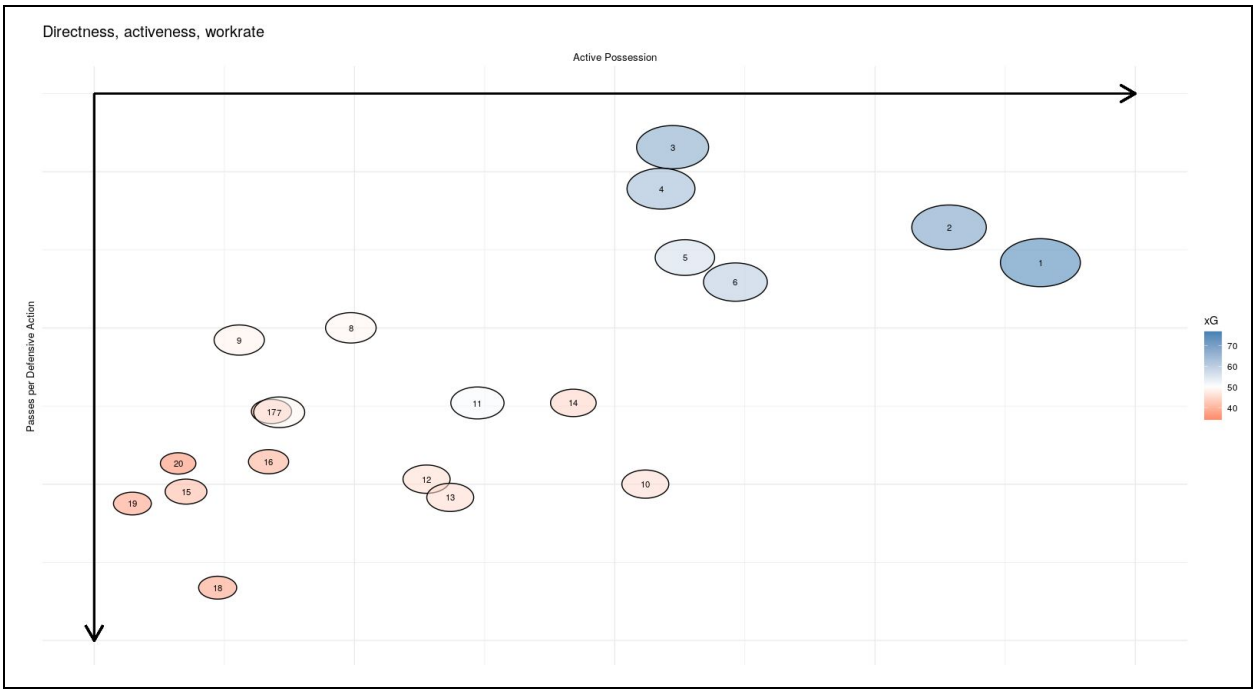
It is the ratio of deep passes to completed passes. It measures how active a team is when in possession of the ball. This value is plotted on the x-axis. Note: this value was calculated exclusively for this visualization.

Workrate is captured by PPDA. A team allowing less passes per defensive action is a team that tries to retrieve the ball as fast as possible. PPDA was constructed earlier, using the following equation:

$$Passes\ Per\ Defensive\ Action\ =\ \frac{Opponent\ Completed\ Passes}{Defensive\ Actions}$$

This value is plotted on the y-axis. We use an inverted axis here.

Finally, the circle area and color gradient represent the xG value. The bigger & bluer the higher the xG:

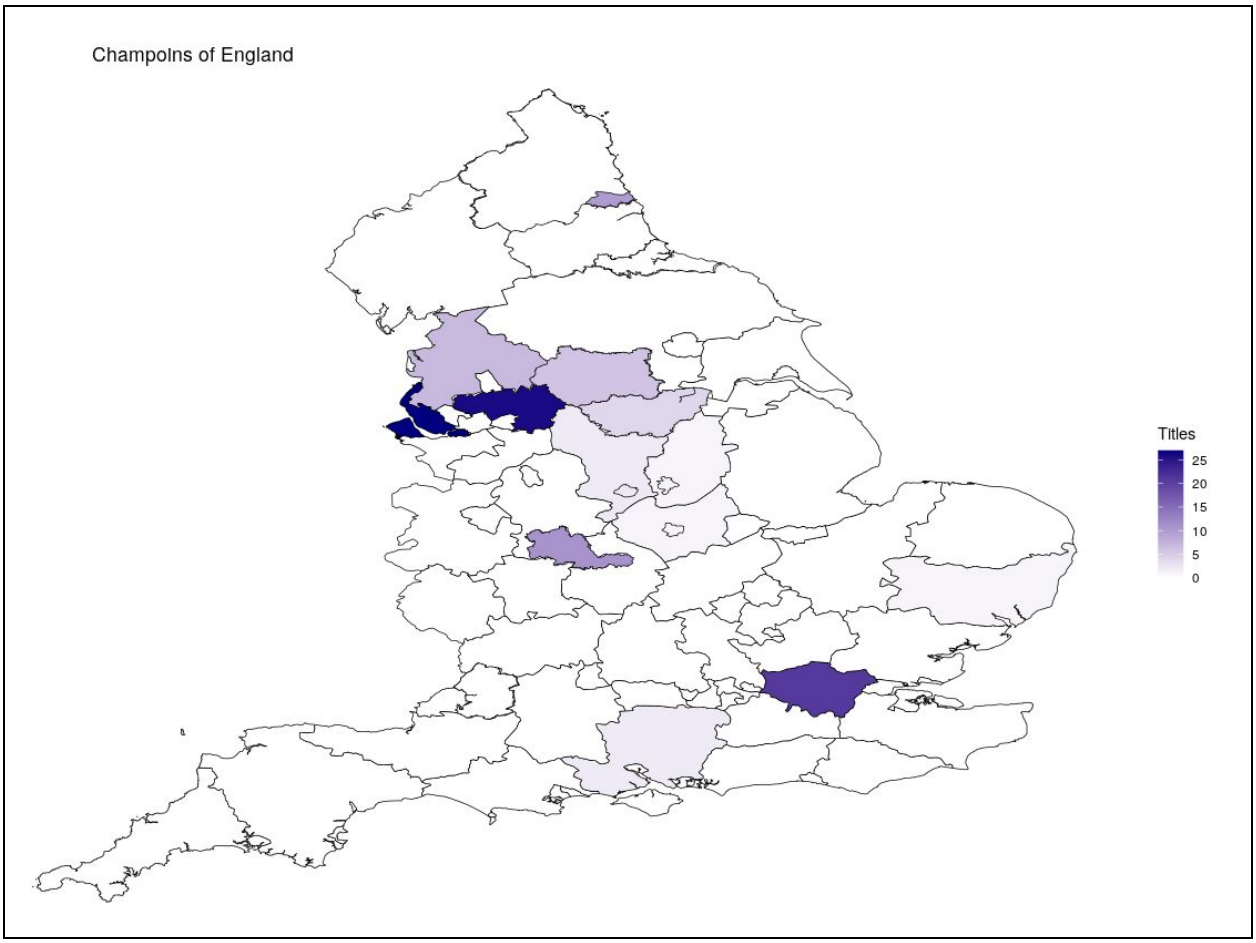


Teams with higher active possession (to the right), and less PPDA (upper), tend to have better xG values.

4. Champions, challengers, and also-rans

4.1 Pedigree[10]

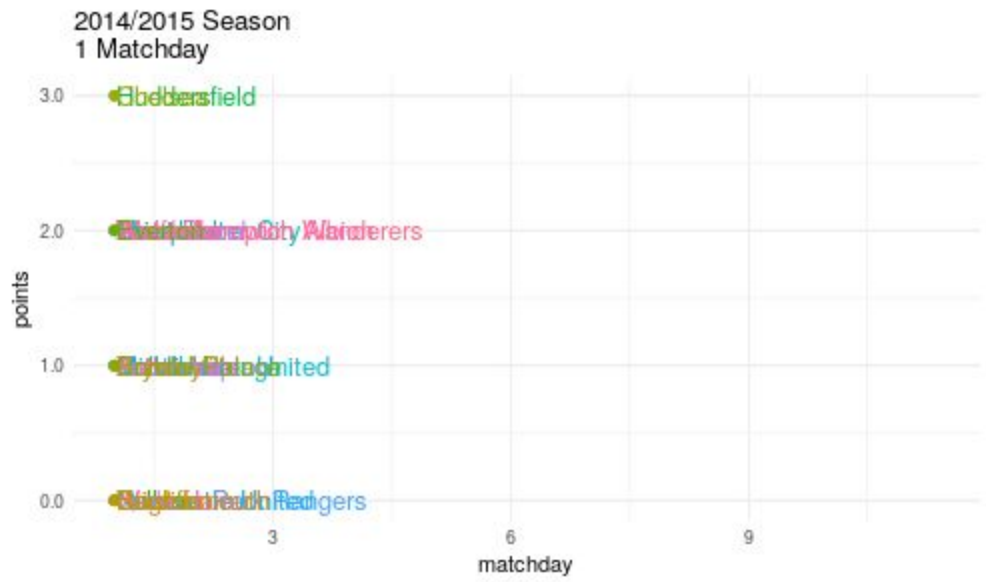
After describing the challenges, framework, and benchmarks, we now move on to looking at actual teams. The first thing any team should do is to set itself a challenge at the start of the season. Not all teams are meant to challenge for the title, in fact only 31.25% of the teams that ever played in the English top division have won a championship, however, out of the 120 league titles, 50% were won by 4 teams



This visual reiterates the fact that not all teams have the capabilities and the wherewithal to challenge for the titles.
The above is a county map of England. It was created using two separate maps imposed over each other.

4.2 Race to the top[11]

The following visual looks at the progress of all teams throughout the 5 year period of our study:

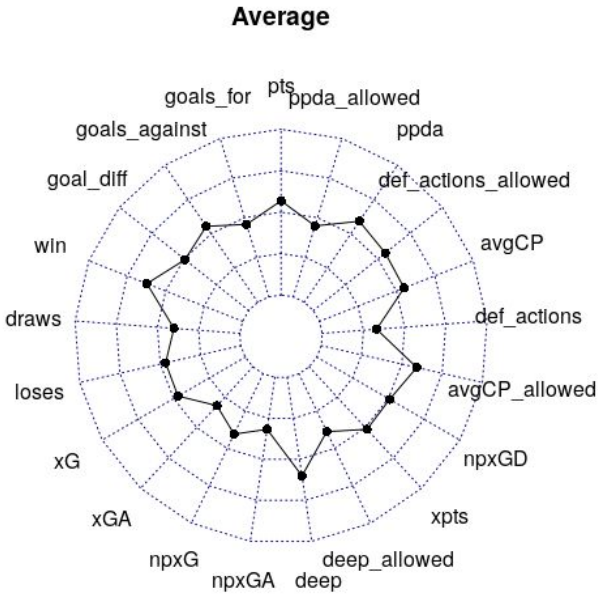


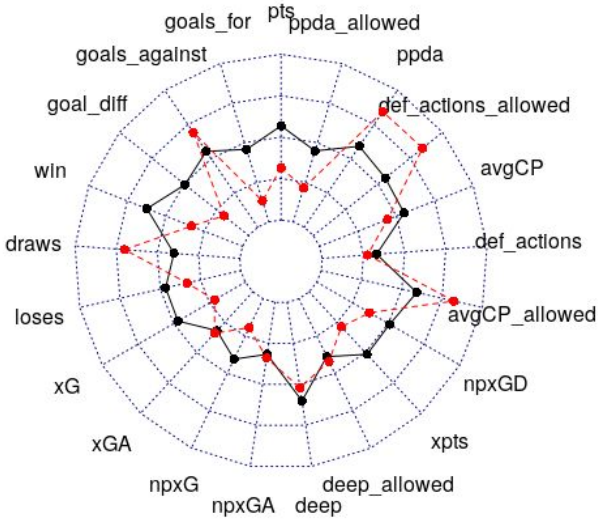
This is a good way to look at identify major events, like a takeover or a managerial change.

4.3 Challenge 1: Win the league[12]

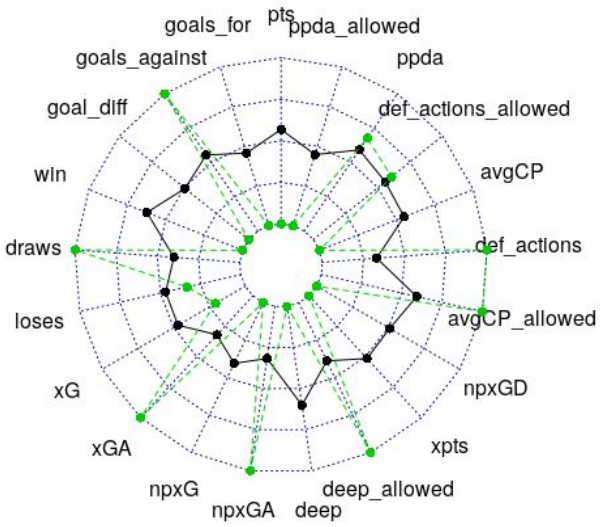
We look at the 5 winners (seasons 2015, 16, 17, 18 & 19). We average their numbers, and use that average as a benchmark for comparison. This will help us identify areas where each champion was better/worse than average Radar charts are a great way to quickly identify weak areas that need improvement:

Anatomy of a champoin

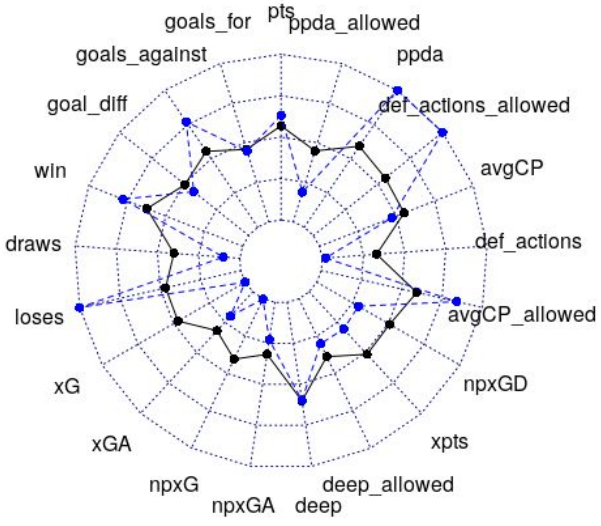




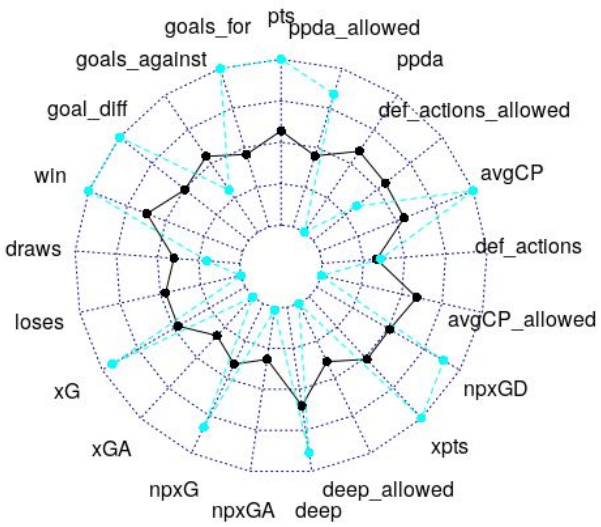
Leicester 16



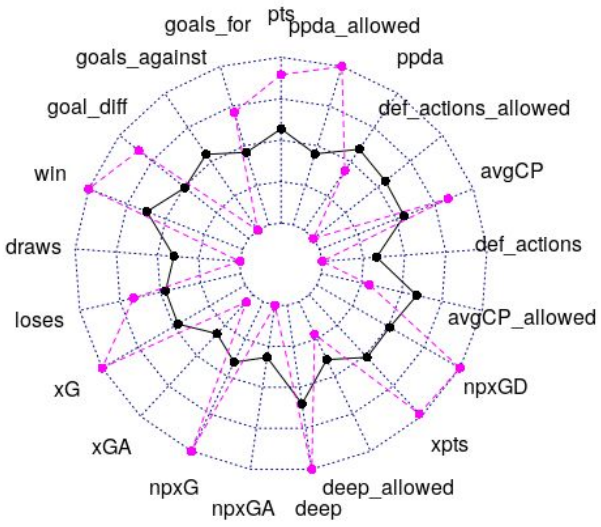
Chelsea 17



Manchester City 18



Manchester City 19



APPENDIX-1

This appendix provides a definition for all the features collected by the get_league_teams_stats function:

Column Name	Type	Description
h_a	categorical	home/away
xG	numerical	expected goals for
xGA	numerical	expected goals against
npxG	numerical	expected goals for without penalties and own goals
npxGA	numerical	expected goals against without penalties and own goals
deep	numerical	passes completed within an estimated 20 years of goal (crosses excluded)
deep_allowed	numerical	opponent passes completed within an estimated 20 years of goal (crosses excluded)
scored	numerical	goals scored
missed	numerical	goals allowed
xpts	numerical	expected points
result	categorical	win/lose/draw
date	date	date
wins	binary	win/no win
draws	binary	draw/no draw
loses	binary	loss/no loss
pts	categorical	3/1/0
npxGD	numerical	the difference between "for" and "against" expected goals without penalties and own goals

ppda.att	numerical	passes allowed in the opposition half
ppda.def	numerical	defensive actions in the opposition half
ppda_allowed.att	numerical	opponent passes allowed in the opposition half
ppda_allowed.def	numerical	opponent defensive actions in the opposition half
team_id	categorical	1:20
team_name	categorical	team name
league_name	char	league name
year	date	year

APPENDIX-2

This appendix provides a list of the excluded and constructed features in the get_season() function.

Excluded features

h_a: home/away → we're taking into account a whole picture. A separate analysis could be done for home & away performances

result → redundant, replaced by wins, losses and draws columns

date → we're only considering season ending rank, so we don't need it

team_id → we're using team_name as an identifier, so we don't need it

league_name → implicit knowledge

year → see date

season → see date

npxG: non-penalty expected goals → for simplification, and the assumption that xG will be a better representation for the overall team activity - attacking teams are expected to win penalties and take advantage of own-goals by virtue of occupying opponents defensive space, however could be included in player analysis to remove noise - for a clearer picture of what a player is capable of

npxGA: non-penalty expected goals against → see npxG

xpts: expected points → for simplification

npxGD: non-penalty expected goal difference → see np

Constructed features

goal_diff → total goals for - total goals against

ppda → ppda.att / ppda.def

ppda_allowed → ppda_allowed.att / ppda_allowed.def

APPENDIX-3

This appendix lists all the libraries that need to be installed for the code to run successfully:

```
# loading libraries

install.packages('remotes')
remotes::install_github('ewenme/understatr')
library(understatr) # > main
library(tidyverse) # > main2
library(fmsb) # > for radar chart
library(moments) # ???
library(reshape2) # > scatterplot
library(effectsize) # > for racing lines gif
library(gganimate) # > for racing lines gif
library(gifski) # > for racing lines gif
library(transformr) # > for racing lines gif
library(scales) # ???
library(ggrepel) # ???
library(ggcorrplot)  # > for correlation plot
library(RcppRoll) # > line plot
library(ggforce) # > goals xG
library(maps) # > geovisualization
library(mapdata) # > geovisualization
library(maptools) # > geovisualization
library(rgdal) # > geovisualization
library(ggmap) # > geovisualization
library(rgeos) # > geovisualization
library(broom) # > geovisualization
library(geosphere) # > geovisualization
library(sp) # > geovisualization
library(gpclib)
library(sf)
```

The code for installating the understatr package is provided here. The other libraries could be easily installed using the install.packages() command.

Sources:

1- https://en.wikipedia.org/wiki/Premier_League

2- https://en.wikipedia.org/wiki/Premier_League#Competition_format

3- https://understat.com/

4- https://github.com/ewenme/understatr

5- https://cartilagefreecaptain.sbnation.com/2015/4/10/8381071/football-statistics-expected-goals-michael-caley-deadspin

6- https://statsbomb.com/2014/07/defensive-metrics-measuring-the-intensity-of-a-high-press/

7- https://en.wikipedia.org/wiki/Hierarchical_clustering

8- a- http://www.sthda.com/english/wiki/ggplot2-rotate-a-graph-reverse-and-flip-the-plot

b- http://www.sthda.com/english/wiki/ggplot2-axis-ticks-a-guide-to-customize-tick-marks-and-labels

9- a- https://stackoverflow.com/questions/43721603/r-changing-color-of-stacked-barplot

b- https://stackoverflow.com/questions/35090883/remove-all-of-x-axis-labels-in-ggplot

c- https://www.datanovia.com/en/blog/how-to-create-a-pie-chart-in-r-using-ggplot2/

10- a- https://www.theguardian.com/football/datablog/2013/apr/17/football-league-125-years

b- https://en.wikipedia.org/wiki/List_of_English_football_champions#Total_titles_won

c- https://en.wikipedia.org/wiki/Metropolitan_and_non-metropolitan_counties_of_England

d- https://geoportal.statistics.gov.uk/

e- https://gadm.org/download_country_v3.html

f- https://stackoverflow.com/questions/47078123/filter-shapefile-polygons-by-area

g- https://gis.stackexchange.com/questions/264250/merge-two-shapefiles-in-r#264325

h- https://rpubs.com/mbacou/ukPubs

i- http://bl.ocks.org/prabhasp/raw/5030005/

j- https://datatricks.co.uk/creating-maps-in-r

k- https://klein.uk/teaching/viz/datavis-maps/

11- https://github.com/lukebornn/outbreak_animation

12- https://www.r-graph-gallery.com/143-spider-chart-with-saveral-individuals.html

Other sources:

https://www.pinnacle.com/en/betting-articles/Soccer/expected-goals-explained/B8Q2HGJ7XMJRZ58C

https://rdrr.io/cran/geomnet/man/geom_circle.html