Creating a statistical model to predict the EURO 2020

Constantin Kumaus

Abstract: Predicting football results accurately is receiving more and more interest. In this paper the main aspects of the Dixon and Coles (1997) model for predicting football scores are outlined. Especially the estimation of attacking and defending parameters is being focused on. Additionally, differences between modelling league football scores and international games are summarized. Furthermore, the model is applied to a training dataset in order to rate the current abilities for the teams participating in the EURO 2020 and predicting its most likely winner. Finally, the model is validated on a previous tournament to gain information about its prediction accuracy.

1. Introduction

Statistics and data analysis in sports is a growing field. Some sports like basketball, baseball or American football have a long history of using data to rate players and teams and to predict scorelines. Even a Hollywood movie has been made to acknowledge the success of data in baseball ('Moneyball'). In football, data analysis exceeding descriptive statistics has only started to receive noticeable interest in the last few years. This can be explained by the differing natures of the sports. Baseball for example is more static and round based as well as high scoring than football. On average only 2.4 goals are scored in a UEFA EURO football match (Worldfootball.net, 2021). This is a high contrast to basketball or baseball, where results like 100-80 and 9-6 respectively are not a rarity. Even if you adjust for the fact that scoring events are not all worth the same (a basket in basketball can be awarded up to 3 points whereas a goal in football always counts as 1), it remains that a scoring event takes place significantly more often. In a nutshell the main objective of football, which is scoring a goal, can be observed fewer times with fewer important factors driving the success of an attacking move. Shots and shots on goal as well as ball possession have been supplied by media for a longer time, mainly to demonstrate the dominance of a team. Only recently advanced metrics such as expected goals, expected assists (which take into account the quality of the chance, e.g. by measuring the angle and distance to goal) (Rathke, 2017) or defensive coverage have been collected and calculated by sports data companies such as Opta or StatsPerform to improve prediction of games and player ratings. These are not widely available since private companies collect or calculate this data and are not keen on providing it to the open public. Therefore this paper relies on some older, more traditional approaches to modelling football results which can do without sophisticated datasets.

One of the most popular models for predicting football results in scientific literature has been described by Dixon and Coles (1997), who in turn base their findings on a model set up by Maher (1982).

2. Dixon-Coles model

2.1. Idea

The Dixon-Coles model can be subordinated to a collection of goal based models. Prediction and analysis in this category revolves around the number of goals a team scores in a match. As aforementioned, scoring in football is a rare event in time. Therefore goals being poisson distributed seems to be a reasonable assumption. Indeed, football competitions around the world show poisson patterns if you look at the empiric distributions of goals. The basic idea of these models is to estimate an attacking and defending strength for each team using historic results. When two teams face each other these figures can be transformed into probabilities for scoring a goal and therefore every possible result has a corresponding chance of happening. (Robberechts and Davis, 2018)

Advantages of this approach are the little requirements to the data, as the model is mainly trained on the end result which is widely available for basically every competition. As teasered in the introduction more advanced models use metrics gathered by private enterprises unavailable to the public.

Unless stated otherwise, this chapter relies on Dixon and Coles' (1997) publication and every method and specification is taken from it. In their study they describe 5 important attributes a statistical model for football matches should feature which are briefly listed as follows:

1. Both teams have different abilities.
2. Home advantage is a factor in football.
3. By the nature of football, a team's total ability can best be outlined by its attacking and defending ability.
4. A team's ability should summarize and reflect its recent performance.
5. The opponent's ability should be taken into account when rating historic performances of a team.

For a match between a home team indexed $_i$ and an away team indexed $_j$, Dixon and Coles (1997) consider a setting with independently distributed variables for home goals $X_{i,j}$ and away goals $Y_{i,j}$:

$$X_{i,j} \sim Poisson(\alpha_i \beta_j \gamma),$$

$$Y_{i,j} \sim Poisson(\alpha_j \beta_i).$$

α_i and β_i stand for the home team's attacking and defending strength respectively while the variables with index $_j$ indicate the strengths for the away team.

Additionally a parameter $\gamma > 0$ is included to adjust for the home advantage, which has been subject of many papers, all of which conclude its existence (Pollard, 2008). In an international tournament such as the FIFA World Cup or the EURO the home advantage can usually be neglected since it takes place in just one or two countries. The EURO 2020, as the tournament is officially named despite being postponed to 2021 due to the COVID-19 pandemic, is different and takes place in 12 countries all over Europe, 9 of which participate in the competition. With more than a third of the teams experiencing home advantage, it would make sense to include it in the model. However, matches will be played in front of 25-50% of a stadiums potential capacity due to COVID-19 restrictions (UEFA, 2021). McCarrick et al. (2020) point out a significant decrease in the home advantage of leagues across Europe due to fewer spectators. With this in mind the home effect should be modelled in a reduced way compared to regular fixtures.

2.2. Model Specification

The aim of the inference is to find valid estimators for every α_i and β_i as well as the home advantage parameter γ. For simplicity Dixon and Coles (1997) assume γ to be equal for every team. One could extend this model with γ_i to adjust for possible differences in the magnitude of the home advantage.

Using the assumption of independence, the probabilities of specific results can be formulated as bivariately poisson distributed in the following way:

$$Pr(X_{i,j} = x, Y_{i,j} = y) = \frac{\lambda^x exp(-\lambda)}{x!} \frac{\mu^y exp(-\mu)}{y!},$$

with

$$\lambda = \alpha_i \beta_j \gamma$$

and

$$\mu = \alpha_j \beta_i.$$

When checking for independence, Dixon and Coles (1997) notice that there seems to be dependence among low scoring games 0-0, 0-1, 1-0, 1-1. Therefore they additionally include a parameter τ which adjusts for the departure of independence. The model described by Maher (1982) is simpler and renounces the

τ assuming independence for every possible result. For simplicity this paper is neglecting τ.

By now, only one of the desired features described in section 2.1 remains to be included in the model: The time dependency. Dixon and Coles (1997) solve this by adding a weighting function ϕ. A simple possibility would be

$$\phi = \begin{cases} 1 & t \leq t_0 \\ 0 & t > t_0 \end{cases}$$

which weighs all matches within the last t_0 time units the same.

A more advanced model with parameter $\xi > 0$ downweighs the previous results exponentially over time:

$$\phi = exp(-\xi t).$$

The bigger ξ is chosen the more weight is given to recent results. The value of ξ can either be optimized or chosen arbitrarily. Dixon and Coles (1997) point out the complexity of this optimization problem which will not be further examined in this paper.

2.3. Model Inference

In a setting with n teams, attacking parameters $\{\alpha_1, ..., \alpha_n\}$, defending parameters $\{\beta_1, ..., \beta_n\}$ and the home advantage parameter γ need to be estimated. Additionally, Dixon and Coles (1997) add the constraint

$$\frac{1}{n}\sum_{i=1}^{n}\alpha_i = 1$$

to prevent overfitting.

Combining all of the specifications above the likelihood function

$$L_t(\alpha_i, \beta_i, \gamma; i = 1, ..., n) = \prod_k \{exp(-\lambda_k)\lambda_k^{x_k} exp(-\mu_k)\mu_k^{y_k}\}^{\phi(t-t_k)}$$

where

$$\lambda_k = \alpha_{i(k)}\beta_{j(k)}\gamma$$

and

$$\mu_k = \alpha_{j(k)}\beta_{i(k)}$$

can be obtained with $_{j(k)}$ and $_{i(k)}$ determining the indices of the home and away teams playing in match k.

Using maximum likelihood, the set of parameter values which makes the observed results the most probable is estimated.

2.4. Limitations

An obvious limitation of the model is that it simply uses past results for predicting the future without taking into account current factors such as player ratings, player's market values or game metrics apart from the result.

For hierarchical league systems Dixon and Coles (1997) detect a potential issue. Their examined competition of interest stretches over four divisions. In this situation most of the games are only between teams from the same division, resulting in a scenario where the best team of the last division is rated similar to the best team of the first division, which can skew the results drastically. If enough seasons are included this effect is marginalised because good rated teams of lower leagues get promoted and play tougher opponents which automatically adjusts their rating. Cup games also help to address this issue.

With the EURO 2020 being the competition of interest, this will most likely not be a problem, since it is restricted to teams which are member of the UEFA and are therefore organised under the same umbrella organisation. Thus every team more or less has the same likelihood of facing any other opponent in the included matches of the training data and there is no "division-bias" as you might call it. Only the UEFA Nations League, which was founded in 2018, puts teams in leagues based on their strengths with the possibility of getting relegated or promoted. However, due to the recent start of the format, Nations League matches account only for a tiny fraction of the training data.

Furthermore, the described adaptations of Dixon and Coles' (1997) model - not including parameter τ for dependent results and downsizing the home advantage effect - remain to be evaluated. For international games they seem to make sense. A trade-off between simplicity of the model and accuracy of the prediction will be expected.

3. Application to EURO 2020

The aim in this section is to apply the Dixon-Coles model described in section 2 to real data in order to find the most likely winner of the EURO 2020. Opisthokonta (2014) provide a way of implementing the Dixon-Coles model for league systems into R. With few adaptations this can be applied to international matches as well. Siddhant (2016) implements a similar model for predicting the EURO 2016.

3.1. Data

As described above, the requirements for the Dixon-Coles model are very few, which makes it easy to find suitable datasets. Kaggle user Jürisoo (2021) hosts a dataset including every international football match result between November 30th, 1872 and March 31st, 2021.

TABLE 1
Extract of Dataset

	date	home_team	away_team	home_score	away_score
42079	2021-03-31	England	Poland	2	1
42080	2021-03-31	Andorra	Hungary	1	4
42081	2021-03-31	San Marino	Albania	0	2
42082	2021-03-31	Armenia	Romania	3	2
42083	2021-03-31	Germany	North Macedonia	1	2
42084	2021-03-31	Liechtenstein	Iceland	1	4

	tournament	city	country	neutral
42079	FIFA World Cup qualification	London	England	FALSE
42080	FIFA World Cup qualification	Andorra la Vella	Andorra	FALSE
42081	FIFA World Cup qualification	Serravalle	San Marino	FALSE
42082	FIFA World Cup qualification	Yerevan	Armenia	FALSE
42083	FIFA World Cup qualification	Duisburg	Germany	FALSE
42084	FIFA World Cup qualification	Vaduz	Liechtenstein	FALSE

Having information on both teams, the corresponding scores for each team and a binary variable stating whether home advantage is applicable, everything needed for the implementation of the Dixon-Coles Model is given.

Before the model can be applied, a training dataset has to be defined. Siddhant (2016) for example only includes games where both teams participate in the competition, limiting the set to a total number of 24 teams. This has two big advantages. Firstly, the algorithm used to compute the problem described by

Opisthokonta (2014) was originally introduced for league systems and requires the same number of home teams as away teams in the training data. Secondly, computation is fast and only relevant attacking and defending scores which are later needed for prediction are calculated. Brazil's values for example will not be further needed for predicting scores in a EURO tournament. For the model with 24 teams it takes few seconds to complete the estimation whereas a model trained on every game with at least one participant of the EURO 2020 takes more than 15 minutes.

However, this procedure comes at a cost as it misses out on a whole lot of relevant games, most prominently World Cup fixtures where hardly any games are between UEFA members. A balance of both approaches can be achieved by filtering the dataset for matches with teams that have played at least one home *and* away game against EURO 2020 participants in the specified period. Taking January 1^{st} 2012 as a threshold date for inclusion in the training set gives a total of 84 teams for which parameters are calculated. The estimation takes about 8 minutes. The parameters for teams not participating are biased, since they have significantly fewer games in the dataset as only the ones against EURO 2020 participants are taken into consideration. This can be neglected since no interest is attached to predicting scores for these teams.

For comparison in the next chapter both Siddhant's (2016) strategy with matches exclusively between the 24 participants in the training data and the choice of matches between 84 teams are considered.

3.2. Parameter Estimation

Optimization of the likelihood function in 2.3 is implemented using the *auglag* function within the R-Package *alabama*.

For all estimations the first time function ϕ introduced in section 2.2 is chosen, weighing every result the same.

When training on data from 2012 onwards, Siddhant's (2016) approach uses 556 games and gives the estimations shown in Table 2 (ordered by total strength). The home advantage parameter γ is estimated to be 0.261.

For the same period, the estimation results of the approach of including 84 teams (1799 games) can be seen in Table 3. Here γ is diminished and estimated to be 0.198.

TABLE 2
Training data 2012-2021 exclusively between EURO 2020 participants

	Attack	Defense
Spain	1.41	-1.39
France	1.37	-1.29
Belgium	1.35	-1.25
Netherlands	1.42	-1.01
Portugal	0.98	-1.41
England	1.14	-1.24
Germany	1.36	-0.96
Italy	0.91	-1.26
Switzerland	1.39	-0.74
Croatia	1.18	-0.79
Russia	1.09	-0.85
Poland	0.98	-0.92
Denmark	0.98	-0.90
Sweden	0.99	-0.87
Turkey	0.97	-0.87
Slovakia	0.84	-0.98
Austria	0.90	-0.85
Wales	0.82	-0.88
Ukraine	0.69	-0.98
Czech Republic	0.75	-0.80
Scotland	0.75	-0.65
Hungary	0.82	-0.50
Finland	0.49	-0.68
North Macedonia	0.42	-0.66
Avg. participants	1.00	-0.95

TABLE 3
Training data 2012-2021 between all teams matching filter criteria

	Attack	Defense
Spain	2.00	-1.93
Belgium	2.01	-1.69
France	1.87	-1.81
England	1.80	-1.85
Germany	2.00	-1.54
Portugal	1.75	-1.77
Netherlands	1.92	-1.60
Italy	1.64	-1.74
Switzerland	1.66	-1.52
Croatia	1.65	-1.48
Denmark	1.60	-1.52
Poland	1.63	-1.49
Sweden	1.55	-1.49
Ukraine	1.46	-1.58
Russia	1.61	-1.41
Austria	1.52	-1.38
Turkey	1.53	-1.37
Slovakia	1.38	-1.42
Czech Republic	1.46	-1.30
Wales	1.26	-1.49
Scotland	1.38	-1.24
Finland	1.20	-1.25
Hungary	1.33	-1.10
North Macedonia	1.15	-1.11
Avg. participants	1.60	-1.50

Note that both strategies give similar results in terms of the ranking of teams. The exact values however vary widely. This is due to the fact that out of the calculated values for 84 teams, only the estimates for the participating teams are displayed in Table 3 which all appear to be above average in attacking terms. According to the constraint set in section 2.2 the mean attacking value for all teams equals 1. The proportionally higher attacking values for participating teams do not lead to biased match simulations as the defending values are also proportionally stronger.

Seeing teams like North Macedonia and Finland at the bottom as well as Spain, France and Belgium on top for both training scenarios makes sense. The cases of Wales and the Netherlands help to see that some adjustments need to be made. In Model 1 the Netherlands are rated astonishingly high, despite not having done so well recently whilst achieving outstanding results in 2012 and 2014. They benefit from the exclusion of most World Cup matches as they did not participate in the World Cup 2018. Wales on the other hand is undervalued in the bigger model. The team had a stellar rise beginning in 2015 and peaking in the EURO 2016, in which it performed outstandingly good. The combination of both aspects indicates that the period of consideration is too long.

Training the model on data from 2015 onwards with 72 teams and 1152 games remaining in the dataset leads to plausible estimations displayed in Table 4, with $\gamma = 0.262$.

Table 4
Training data 2015-2021

	Attack	Defense
Spain	2.08	-2.04
England	1.86	-2.05
Belgium	2.18	-1.69
France	1.97	-1.84
Portugal	1.88	-1.93
Italy	1.74	-1.98
Germany	2.03	-1.64
Netherlands	1.97	-1.70
Denmark	1.73	-1.88
Poland	1.81	-1.59
Switzerland	1.68	-1.68
Croatia	1.75	-1.54
Sweden	1.58	-1.68
Turkey	1.55	-1.58
Wales	1.39	-1.71
Ukraine	1.49	-1.56
Russia	1.68	-1.33
Austria	1.57	-1.42
Slovakia	1.53	-1.45
Czech Republic	1.51	-1.36
Scotland	1.47	-1.25
Hungary	1.37	-1.24
Finland	1.16	-1.41
North Macedonia	1.24	-1.21
Avg. participants	1.68	-1.62

3.2.1. Parameter interpretation

Recall the parameter definitions introduced in section 2.2:

$$\lambda = \alpha_i \beta_j \gamma,$$

$$\mu = \alpha_j \beta_i.$$

The *auglag* function performs a logarithmic transformation to exploit computational advantages which actually makes it estimate the log-values of every α_i, β_i and γ.

It follows that for a specific match, the ratings can be transformed to parameters for the poisson distribution λ and μ in the following way:

$$\lambda = exp(\hat{\alpha_i} + \hat{\beta_j} + \hat{\gamma})$$

and

$$\mu = exp(\hat{\alpha_j} + \hat{\beta_i})$$

where $\hat{\alpha}$, $\hat{\beta}$ and $\hat{\gamma}$ denote the estimators for α, β and γ respectively.

These values correspond to the goals each team is expected to score in a certain match against a certain opponent.

For example for the opening match of the EURO 2020 Italy host Turkey in Rome. For this game the expected goals for Italy are 1.526 if the full home advantage is included and 1.174 without home advantage. For Turkey the value is 0.649.

Taking the exponential of the estimated home advantage value, it can be interpreted as a factor. A γ of 0.262 corresponds to 1.3 times the chance of scoring a goal. In other words, the expected goals are increased by about 30%.

3.3. Prediction

The estimated parameters can be used to simulate any match between the included teams which in this application are the participants of the EURO 2020. Using R, an algorithm has been implemented to simulate a large number of tournament runs using the estimated parameters from section 3.2, Table 4. Home advantage has not been included due to the reasons described in 2.1. Simulating the tournament 100000 times resulted in the following outcome statistics:

According to the implemented model, Spain has the highest chance of winning the tournament (20.652%) ahead of Belgium and England. Interestingly, Belgium is more likely to come out as a winner than England, although England

TABLE 5
Probability of a tournament win

Spain	0.20652
Belgium	0.12804
England	0.10875
France	0.09296
Portugal	0.08780
Netherlands	0.07351
Italy	0.07297
Germany	0.06039
Denmark	0.05605
Poland	0.02326
Switzerland	0.02113
Croatia	0.01582
Sweden	0.01294
Turkey	0.00824
Wales	0.00772
Ukraine	0.00704
Austria	0.00543
Russia	0.00412
Slovakia	0.00282
Czech Republic	0.00239
Scotland	0.00094
Finland	0.00063
North Macedonia	0.00029
Hungary	0.00024

TABLE 6
Probability of a group stage exit

Hungary	0.82185
Finland	0.69921
North Macedonia	0.66658
Slovakia	0.58506
Scotland	0.57969
Czech Republic	0.47514
Russia	0.45787
Wales	0.43212
Turkey	0.42102
Sweden	0.39469
Austria	0.32052
Poland	0.31638
Switzerland	0.30100
Ukraine	0.29220
Croatia	0.23071
Germany	0.17397
Italy	0.15607
Denmark	0.15101
Portugal	0.14377
France	0.13350
Netherlands	0.06448
Spain	0.06243
England	0.06099
Belgium	0.05974

has a higher rating in terms of attacking and defending values based on Table 4. This suggests that England face a tougher tournament path with a higher likelihood of facing strong opponents early. Similarly, despite not being the lowest rated team in Table 4, Hungary has the lowest chance of winning because they have been drawn into a group with 3 title contenders France, Portugal and Germany and will most likely not go past the group stage as Table 6 shows.

3.4. Shiny App

Additionally, a Shiny App has been deployed which lets the user run his own simulations, reproduce the results described above and understand the dynamics of the model. It can be accessed at https://kumaus.shinyapps.io/euro2020_simulation/. A brief instruction on how the app is to be used follows:

- The tab 'Single detailed Simulation' lets the user simulate one run and look at every group and the K.O.-stage in detail.
- At 'Multiple Simulations with outcome statistics' one can simulate any number of simulation runs and look at descriptive statistics for every team and stage. Note that 1000 simulations take about 40 seconds to compute and 10000 simulations about 7 minutes.

- The last tab lets you simulate any number of games between any two participants and is designated to give insights on the simulation behaviour.

3.5. Validation

In order to deduct prediction accuracy it is helpful to validate the model on a previous tournament. This can be done by comparing the predicted outcome to the actual result for every match. The EURO 2016 was played in the same format and is therefore suitable for validation. Using the same inclusion criteria for the training data as in the EURO 2020 model (teams having played at least one home and away match against a participant of the EURO; games within the last 6 years before the tournament year i. e. from 2010 onwards) results in the strength parameter estimations shown in Table 7:

TABLE 7
Estimated parameters for EURO 2016

	Attack	Defense
Spain	1.95	-2.00
Germany	2.16	-1.61
France	1.78	-1.93
England	1.81	-1.81
Belgium	1.83	-1.65
Portugal	1.76	-1.71
Russia	1.58	-1.84
Croatia	1.68	-1.67
Ukraine	1.63	-1.61
Italy	1.59	-1.61
Sweden	1.71	-1.43
Republic of Ireland	1.55	-1.57
Poland	1.63	-1.47
Switzerland	1.59	-1.46
Austria	1.66	-1.35
Turkey	1.52	-1.49
Czech Republic	1.53	-1.40
Romania	1.40	-1.46
Slovakia	1.33	-1.43
Hungary	1.53	-1.14
Albania	1.07	-1.46
Iceland	1.42	-1.10
Wales	1.20	-1.26
Northern Ireland	0.95	-1.04
Average	1.58	-1.52

For the sake of completeness γ is estimated with 0.172 although it is not further implemented in prediction.

Based on these parameters the theoretical probability for any possible result can be calculated. For example Table 8 displays the goal probability matrix for the opening match of the EURO 2016 between France and Romania, resulting in the outcome probabilities shown in Table 9.

Table 8
France vs. Romania: goal probabilities

	Romania: 0	1	2	3	4	>4
France: 0	0.14143	0.08327	0.02451	0.00481	0.00071	0.00009
1	0.19336	0.11385	0.03352	0.00658	0.00097	0.00013
2	0.13218	0.07782	0.02291	0.00450	0.00066	0.00009
3	0.06024	0.03547	0.01044	0.00205	0.00030	0.00004
4	0.02059	0.01212	0.00357	0.00070	0.00010	0.00001
>4	0.00721	0.00425	0.00125	0.00025	0.00004	0.00000

Table 9
France vs Romania: outcome probability

	France	D	Romania
1	0.55947	0.28034	0.16018

In this case the prediction was correct. A win for France, which was the actual result of the match in the real tournament, also had the highest outcome probability. Iterating through every match like this gives a mean prediction accuracy of 0.412. This value is higher than $\frac{1}{3}$ which would be the expected accuracy if, for each match, you randomly assigned win, draw or loss with the same probability. Predicting the exact score rather than the outcome is even more difficult and yields a prediction accuracy of 0.176. A detailed table on the exact predictions for every game can be taken from Table 10 in the appendix. Both values are not good prediction scores. This can be explained by the fact that poorly rated teams like Wales and Iceland outperformed in the tournament and won games they were not expected to win. Also the total number of 51 games is sensitive to outliers and random events as one single match with another result than expected already accounts for a 2% drop in prediction accuracy.

4. Conclusion

This paper has given an insight on the Dixon and Coles (1997) method to determine current strengths of football teams using historic data. Implementing 100000 simulation runs of the EURO 2020 based on the estimated parameters has declared Spain to be the most likely winner of the tournament with a chance of 20.652%. Furthermore, validation on the preceding tournament in 2016 has determined a prediction accuracy of 41.2%. This shows the presence of luck and randomness in football. Although it is possible to declare a favourite for a match, underdogs have a fair chance in football. This 'unpredictability' makes such competitions interesting to watch and could be one driving factor for footballs popularity.

References

M. J. Dixon and S. G. Coles. Modelling association football scores and inefficiencies in the football betting market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46(2):265–280, 1997.

M. Jürisoo. International football results from 1872 to 2020, Apr 2021. URL https://www.kaggle.com/martj42/international-football-results-from-1872-to-2017. (Accessed on 2021-04-28).

M. J. Maher. Modelling association football scores. *Statistica Neerlandica*, 36 (3):109–118, 1982.

D. McCarrick, M. Bilalic, N. Neave, and S. Wolfson. Home advantage during the covid-19 pandemic in european football. 2020.

Opisthokonta. The dixon-coles model for predicting football matches in r, Nov 2014. URL https://opisthokonta.net/?p=890. (Accessed on 2021-05-01).

R. Pollard. Home advantage in football: A current review of an unsolved puzzle. *The open sports sciences journal*, 1(1), 2008.

A. Rathke. An examination of expected goals and shot efficiency in soccer. *Journal of Human Sport and Exercise*, 12(2):514–529, 2017.

P. Robberechts and J. Davis. Forecasting the fifa world cup–combining result- and goal-based team ability parameters. In *International Workshop on Machine Learning and Data Mining for Sports Analytics*, pages 16–30. Springer, 2018.

Siddhant. Building a statistical model for predicting uefa euro 2016 results, 2016. URL https://medium.com/@siddhant_08/building-a-statistical-model-for-predicting-euro-2016-results-946ea1561c8a. (Accessed on 2021-04-30).

UEFA. Key information for euro spectators, 2021. URL https://www.uefa.com/uefaeuro-2020/news/025b-0ef33753d7d0-100629325be2-1000--key-information-for-euro-spectators/. (Accessed on 2021-04-26).

Worldfootball.net. Goals per tournament, 2021. URL https://www.worldfootball.net/stats/em/1/. (Accessed on 2021-05-01).

Appendix A: Figures

Table 10
Results vs. Prediction EURO 2016

	T1	T2	Score	Pred_Winner	Pred_Score
1	France	Romania	2 - 1	France	1 - 0
2	Albania	Switzerland	0 - 1	Switzerland	0 - 1
3	Wales	Slovakia	2 - 1	Slovakia	0 - 1
4	England	Russia	1 - 1	England	0 - 0
5	Turkey	Croatia	0 - 1	Croatia	0 - 1
6	Poland	Northern Ireland	1 - 0	Poland	1 - 0
7	Germany	Ukraine	2 - 0	Germany	1 - 1
8	Spain	Czech Republic	1 - 0	Spain	1 - 0
9	Republic of Ireland	Sweden	1 - 1	Sweden	1 - 1
10	Belgium	Italy	0 - 2	Belgium	1 - 0
11	Austria	Hungary	0 - 2	Austria	1 - 1
12	Portugal	Iceland	1 - 1	Portugal	1 - 0
13	Russia	Slovakia	1 - 2	Russia	1 - 0
14	Romania	Switzerland	1 - 1	Switzerland	0 - 1
15	France	Albania	2 - 0	France	1 - 0
16	England	Wales	2 - 1	England	1 - 0
17	Ukraine	Northern Ireland	0 - 2	Ukraine	1 - 0
18	Germany	Poland	0 - 0	Germany	1 - 1
19	Italy	Sweden	1 - 0	Italy	1 - 1
20	Czech Republic	Croatia	2 - 2	Croatia	0 - 1
21	Spain	Turkey	3 - 0	Spain	1 - 0
22	Belgium	Republic of Ireland	3 - 0	Belgium	1 - 0
23	Iceland	Hungary	1 - 1	Hungary	1 - 1
24	Portugal	Austria	0 - 0	Portugal	1 - 0
25	Switzerland	France	0 - 0	France	0 - 1
26	Romania	Albania	0 - 1	Romania	0 - 0
27	Slovakia	England	0 - 0	England	0 - 1
28	Russia	Wales	0 - 3	Russia	1 - 0
29	Northern Ireland	Germany	0 - 1	Germany	0 - 3
30	Ukraine	Poland	0 - 1	Ukraine	1 - 1
31	Croatia	Spain	2 - 1	Spain	0 - 1
32	Czech Republic	Turkey	0 - 2	Turkey	1 - 1
33	Iceland	Austria	2 - 1	Austria	1 - 1
34	Hungary	Portugal	3 - 3	Portugal	0 - 1
35	Sweden	Belgium	0 - 1	Belgium	1 - 1
36	Italy	Republic of Ireland	0 - 1	Italy	1 - 0
37	Switzerland	Poland	1 - 1	Poland	1 - 1
38	Wales	Northern Ireland	1 - 0	Wales	1 - 0
39	Croatia	Portugal	0 - 1	Portugal	0 - 1
40	France	Republic of Ireland	2 - 1	France	1 - 0
41	Germany	Slovakia	3 - 0	Germany	2 - 0
42	Hungary	Belgium	0 - 4	Belgium	0 - 1
43	Italy	Spain	2 - 0	Spain	0 - 1
44	England	Iceland	1 - 2	England	2 - 0
45	Poland	Portugal	1 - 1	Portugal	0 - 1
46	Wales	Belgium	3 - 1	Belgium	0 - 1
47	Germany	Italy	1 - 1	Germany	1 - 0
48	France	Iceland	5 - 2	France	1 - 0
49	Portugal	Wales	2 - 0	Portugal	1 - 0
50	Germany	France	0 - 2	Germany	1 - 1
51	Portugal	France	1 - 0	France	0 - 1

Appendix B: Code

```
library(dplyr)
library(knitr)
library(xtable)

pars_ex <- read.csv("pars_ex.csv")
results <- read.csv("results.csv")
estimation <- load("estimation.rda")



library(kableExtra)



kable(
  tail(results[,1:5]),
  format = "latex",
  booktabs = TRUE,
  linesep = '',
  caption = "Extract of Dataset", label = "data"
) %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position"))



kable(
  tail(results[,6:ncol(results)]),
  format = "latex",
  booktabs = TRUE,
  linesep = '',

) %>%
  kable_styling(latex_options = c("scale_down", "HOLD_position"))




t1 <- rbind(pars_exclusive[
  order(pars_exclusive[,1] + pars_exclusive[,2]*-1, decreasing = TRUE),],
  "Avg. participants" =colMeans(pars_exclusive))

t2 <- rbind(pars24[
  order(pars24[,1] + pars24[,2]*-1, decreasing = TRUE),],
  "Avg. participants" = colMeans(pars24))

t3 <- rbind(pars24_2015[
  order(pars24_2015[,1] + pars24_2015[,2]*-1, decreasing = TRUE),],
  "Avg. participants" = colMeans(pars24_2015))



t1_xtable <- xtable(
  t1,
  digits = 2,
  caption = "Training data 2012-2021 exclusively between EURO 2020 participants",
    label = "parsexclusive"
)


t2_xtable <- xtable(
  t2,
  digits = 2,
  caption = "Training data 2012-2021 between all teams matching criteria",
  label = "pars2012"
)

t3_xtable <- xtable(
  t3,
  digits = 2,
  caption = "Training data 2015-2021",
  label = "pars2015"
)


print(t1_xtable, caption.placement = "top",
  floating=FALSE,
  hline.after=NULL,
  add.to.row=list(pos=list(-1,0, nrow(t1_xtable)),
  command=c('\\toprule\n','\\midrule\n','\\bottomrule\n'))
  )
print(t2_xtable, caption.placement = "top",
  floating=FALSE,
  hline.after=NULL,
  add.to.row=list(pos=list(-1,0, nrow(t2_xtable)),
  command=c('\\toprule\n','\\midrule\n','\\bottomrule\n'))
  )

print(t3_xtable, caption.placement = "top",
  hline.after=NULL,
  add.to.row=list(pos=list(-1,0, nrow(t3_xtable)),
  command=c('\\toprule\n','\\midrule\n','\\bottomrule\n')))

ITATUR <- exp(pars24_2015["Italy", "Attack"]
             + pars24_2015["Turkey", "Defense"] + ha_2015)
ITATUR <- c(ITATUR,
           exp(pars24_2015["Italy", "Attack"]
               + pars24_2015["Turkey", "Defense"]),
           exp(pars24_2015["Turkey", "Attack"]
               + pars24_2015["Italy", "Defense"]))
load("sim_100k.rda")

# Outcomes for Austria
austria_100k <- sort(table(simulations_100k["Austria",])/100000, decreasing = TRUE)

# Winner statistics
winner_100k <- sort(table(rep(rownames(simulations_100k), 100000)
                          [which(grepl(paste0("\\b", "Winner","\\b"),
                                       simulations_100k))])/100000, decreasing = TRUE)

# Groupstage statistics
groupstage_100k <- sort(table(rep(rownames(simulations_100k), 100000)
                              [which(grepl(paste0("\\b", "Group Stage","\\b"),
                                          simulations_100k))])/100000, decreasing = TRUE)


winner_100k_xtable <- xtable(winner_100k,
            digits = 5,
       caption = "Probability of a tournament win",
       label = "win_100k")
names(winner_100k_xtable) <- NULL

gs_100k_xtable <- xtable(
  groupstage_100k,
  digits = 5,
    caption = "Probability of a group stage exit",
  label = "gs_100k")

names(gs_100k_xtable) <- NULL


print(winner_100k_xtable, caption.placement = "top",
  floating=FALSE,
  hline.after=NULL,
  add.to.row=list(pos=list(-1, nrow(winner_100k_xtable)),
  command=c('\\toprule','\\bottomrule\n'))
  )
print(gs_100k_xtable, caption.placement = "top",
  floating=FALSE,
  hline.after=NULL,
  add.to.row=list(pos=list(-1, nrow(gs_100k_xtable)),
  command=c('\\toprule','\\bottomrule\n'))
  )
load("estimation_2016.rda")

t4 <- rbind(pars24_2010[
  order(pars24_2010[,1] + pars24_2010[,2]*-1, decreasing = TRUE),],
  "Average" = colMeans(pars24_2010))

t4_xtable <- xtable(
  t4,
  digits = 2,
  caption = "Estimated parameters for EURO 2016",
  label = "pars2010"
)


print(t4_xtable, caption.placement = "top",
  hline.after=NULL,
  add.to.row=list(pos=list(-1, 0, nrow(t4_xtable)),
  command=c('\\toprule\n','\\midrule\n','\\bottomrule\n')))





load("prob_list.rda")
load("score_probability.rda")
```

```r
res2016 <- read.csv("EURO_2016_results.csv")

# Compare predicted result vs actual result
res2016$Home.Score <- as.numeric(unlist(strsplit(res2016$Result, " - "))[c(TRUE, FALSE)])
res2016$Away.Score <- as.numeric(unlist(strsplit(res2016$Result, " - "))[c(FALSE, TRUE)])
res2016$Score <- res2016$Result

res2016$Result[res2016$Home.Score > res2016$Away.Score] <-
  res2016$Home.Team[res2016$Home.Score > res2016$Away.Score]
res2016$Result[res2016$Home.Score < res2016$Away.Score] <-
  res2016$Away.Team[res2016$Home.Score < res2016$Away.Score]
res2016$Result[res2016$Home.Score == res2016$Away.Score] <-
  "D"

matches2016 <- paste(res2016$Home.Team, "-", res2016$Away.Team)

for(s in 1:length(matches2016)){
  OP <- prob.list.2016[[matches2016[s]]]$`Outcome probability`
  MLR <- prob.list.2016[[matches2016[s]]]$`Most likely result`
  res2016$Prediction[s] <- names(which.max(OP))
  res2016$Prediction_res[s] <- MLR
}

accuracy_2016 <- mean(res2016$Prediction == res2016$Result)
accuracy_result_2016 <- mean(res2016$Score == res2016$Prediction_res)


FRAROM_xtable <- xtable(prob.list.2016$`France - Romania`$`Probability matrix`,
                        digits = 5,
                        caption = "France vs. Romania: goal probabilities",
                        label = "FRAROM")

rownames(FRAROM_xtable) <- c("France: 0", 1:4, ">4")
colnames(FRAROM_xtable) <- c("Romania: 0", 1:4, ">4")

print(FRAROM_xtable, caption.placement = "top")

FRAROM2_xtable <- xtable(prob.list.2016$`France - Romania`$`Outcome probability`,
                         digits = 5,
                         caption = "France vs Romania: outcome probability",
                         label = "FRAROM2")

print(FRAROM2_xtable, caption.placement = "top")

t5 <- res2016[c("Home.Team", "Away.Team", "Score", "Prediction", "Prediction_res")]
colnames(t5) <- c("T1", "T2", "Score", "Pred_Winner", "Pred_Score")

EURO2016_xtable <- xtable(t5,
                        caption = "Results vs. Prediction EURO 2016",
                        label = "EURO2016_pred")




# Inference
#-----------------------------------------------------
```

```r
# read in packages and data
library(dplyr)
setwd("D:\\Surface Dokumente\\Uni\\6. Semester\\Seminar")
results <- read.csv("results.csv")
results <- as.data.frame(results)
results$date <- as.Date(results$date, format = "%Y-%m-%d")

# Define participants of EURO 2020
teams2021 <- c("Italy", "Wales", "Turkey", "Switzerland",
               "Russia", "Belgium", "Denmark", "Finland",
               "Austria", "Ukraine", "Netherlands", "North Macedonia",
               "Czech Republic", "England", "Croatia", "Scotland",
               "Sweden", "Spain", "Poland", "Slovakia",
               "Germany", "France", "Hungary", "Portugal"
)

# Define participants of EURO 2016 for cross validation

teams2016 <- c("France", "Romania", "Albania", "Switzerland",
               "England", "Russia", "Wales", "Slovakia",
               "Germany", "Ukraine", "Poland", "Northern Ireland",
               "Spain", "Czech Republic", "Turkey", "Croatia",
               "Belgium", "Italy", "Republic of Ireland", "Sweden",
               "Portugal", "Iceland", "Austria", "Hungary")


# define threshold dates for inclusion of matches
stichtag2012 <- as.Date("2012-01-01", format = "%Y-%m-%d")
stichtag2015 <- as.Date("2015-01-01", format = "%Y-%m-%d")
# stichtag <- stichtag2012
end <- as.Date("2021-03-31", format = "%Y-%m-%d")
endnow <- Sys.Date()

#for cross validation on EURO 2016
stichtag2010 <- as.Date("2010-01-01", format = "%Y-%m-%d")
end2016 <- as.Date("2016-03-31", format = "%Y-%m-%d")



# Create function for filtering of results df based on threshold date, and teams

filter.results <- function(results, teams, stichtag, endtag){

  results_UEFA_home <- filter(results, home_team %in% teams, date > stichtag, date <= endtag)
  results_UEFA_away <- filter(results, away_team %in% teams, date > stichtag, date <= endtag)
  results_new <- rbind(results_UEFA_away, results_UEFA_home)
  results_new <- unique(results_new)

  results_new <- droplevels(results_new)

  # For later it is important to drop teams which have no away or no home games,
  # so that all teams have home and away games in specified period against UEFA members
  away_nothome <- unique(results_new$away_team)[
    !(unique(results_new$away_team) %in%
        unique(results_new$home_team))]
  home_notaway <- unique(results_new$home_team)[
    !(unique(results_new$home_team) %in%
        unique(results_new$away_team))]

  results_new <- results_new[!(results_new$away_team %in% away_nothome),]
  results_new <- results_new[!(results_new$home_team %in% home_notaway),]
  results_new <- droplevels(results_new)


  #Set new Format for days and include time passed since game took place
  results_new$date <- as.Date(results_new$date, format = "%Y-%m-%d")
  results_new$days <- round(as.numeric(
    difftime(Sys.Date(), results_new$date,units = "weeks")))


  # Additionally only results between 2 participants including transformations as above
  results_exclusive <- filter(results, home_team %in% teams, away_team %in% teams,
                              date > stichtag, date <= endtag)
  results_exclusive$neutral <- !results_exclusive$neutral
  names(results_exclusive)[9] <- "HA"
  results_exclusive <- droplevels(results_exclusive)

  results_exclusive$date <- as.Date(results_exclusive$date, format = "%Y-%m-%d")
  results_exclusive$days <- round(as.numeric(difftime(Sys.Date(), results_exclusive$date, units = "weeks")))

  objects <- list("all" = results_new, "exclusive" = results_exclusive)
  return(objects)
}

# create filtered data frames
results_2012 <- filter.results(results = results, teams = teams2021, stichtag = stichtag2012, endtag = end)
results_2015 <- filter.results(results = results, teams = teams2021, stichtag = stichtag2015, endtag = end)


# implement opisthokonta's algorithm (similar to Siddhant 08)
#-----------------------------------------------------------------------------------

# create Likelihoodfunction
likelihood <- function(h, a, lambda, mu, days, xi = 0.5){
  sum( exp(-xi * days) * (log(dpois(h, lambda)) + log(dpois(a, mu))))
}

# create Data Matrix for home and away team
model_data <- function(df = results_new){

  hm <- model.matrix(~ home_team - 1, data = df, contrasts.arg=list(home_team ='contr.treatment'))
  am <- model.matrix(~ away_team - 1, data = df)

  team_names <- sort(unique(c(df$home_team, df$away_team)))

  return(list(
    homeTeamDM=hm,
    awayTeamDM=am,
    homeGoals=df$home_score,
    awayGoals=df$away_score,
    teams=team_names
  ))
}
```

```r
# Calculate matrices for all training scenarios
model_2012_all <- model_data(results_2012$all)
model_2015_all <- model_data(results_2015$all)
model_2012_exclusive <- model_data(results_2012$exclusive)


# create optimisation function
optimiser <- function(params, model = model_2012_all, days, xi.p = 0){

  home.p <- params[1]

  nteams <- length(model$teams)
  attack.p <- matrix(params[2:(nteams+1)], ncol=1)
  defence.p <- matrix(params[(nteams+2):length(params)], ncol=1)

  lambda <- exp(model$homeTeamDM %*% attack.p + model$awayTeamDM %*% defence.p + home.p)
  mu <- exp(model$awayTeamDM %*% attack.p + model$homeTeamDM %*% defence.p)

  return(
    likelihood(h=model$homeGoals, a=model$awayGoals, lambda, mu, xi.p) * -1
  )
}

## include constraint (mean of alpha_i = 1)

constraint <- function(params, model, ...){
  nteams <- length(model$teams)
  attack.p <- matrix(params[2:(nteams+1)], ncol=1)
  return((sum(attack.p) / nteams) - 1)
}

# initialise estimates for training data with 84 teams (from 2012 onwards)
attack.params <- rep(.9, times=length(unique((results_2012$all$home_team))))
defence.params <- rep(-0.5, times=length(unique((results_2012$all$away_team))))
home.param <- 0.06
par.inits <- c(home.param, attack.params, defence.params)

# initialise estimates for training data with 72 teams (from 2015 onwards)
attack.params_2015 <- rep(.9, times=length(unique((results_2015$all$home_team))))
defence.params_2015 <- rep(-0.5, times=length(unique((results_2015$all$away_team))))
home.param_2015 <- 0.06
par.inits_2015 <- c(home.param_2015, attack.params_2015, defence.params_2015)

# initialise estimates for training data with only 24 participants
attack.params.e <- rep(.9, times=length(unique(results_2012$exclusive$home_team)))
defence.params.e <- rep(-0.5, times=length(unique(results_2012$exclusive$away_team)))
home.param.e <- 0.06
xi <- 0.05
par.inits.e <- c(home.param.e, attack.params.e, defence.params.e)

# name parameters
names(par.inits) <- c('HOME', paste('Attack', model_2012_all$teams, sep='.'),
                      paste('Defence', model_2012_all$teams, sep='.'))
names(par.inits.e) <- c('HOME', paste('Attack', model_2012_exclusive$teams, sep='.'),
                        paste('Defence', model_2012_exclusive$teams, sep='.'))
```

```
names(par.inits_2015) <- c('HOME', paste('Attack', model_2015_all$teams, sep='.'),
                           paste('Defence', model_2015_all$teams, sep='.'))



# # estimate using maximum likelihood
# library(alabama)
# res_2012_all <- auglag(par =par.inits, fn=optimiser, heq=constraint,
#                         model = model_2012_all, days = results_2012£all£days, xi.p = xi)
# res_2012_exclusive <- auglag(par= par.inits.e, fn = optimiser, heq = constraint,
#                               model = model_2012_exclusive, days = results_2012£exclusive£days, xi.p = xi)
# res_2015_all <- auglag(par= par.inits_2015, fn = optimiser, heq = constraint,
#                         model = model_2015_all, days = results_2015£all£days, xi.p = 0.1)
#
# # pars_exclusive contains estimates for data trained on 24 teams
# pars_exclusive <- matrix(res_2012_exclusive£par[2:49], ncol = 2)
# ha_exclusive <- res_2012_exclusive£par[1]
# colnames(pars_exclusive) <- c("Attack", "Defense")
# rownames(pars_exclusive) <- model_2012_exclusive£teams
# #pars_exclusive[order(pars_exclusive[,1]-pars_exclusive[,2],decreasing = TRUE),]
#
#
#
# # pars contains estimates for data trained on 84 teams from 2012 onwards
# pars_all <- matrix(res£par[2:(2*length(model_2012_all£teams)+1)], ncol = 2)
# ha_all <- res_2012_all£par[1]
# colnames(pars_all) <- c("Attack", "Defense")
# rownames(pars_all) <- model_2012_all£teams
#
#
# pars24 <- pars_all[rownames(pars_all) %in% teams2021,]
#
#
# # pars_2015 contains estimates for data trained on 72 teams from 2015 onwards
# pars_2015 <- matrix(res_2015_all£par[2:(2*length(model_2015_all£teams)+1)], ncol = 2)
# ha_2015 <- res_2015_all£par[1]
# colnames(pars_2015) <- c("Attack", "Defense")
# rownames(pars_2015) <- model_2015_all£teams
#
# pars24_2015 <- pars_2015[rownames(pars_2015) %in% teams2021,]
#
#
#
#






# Cross validation on EURO 2016
#-------------------------------------------------

results_2010 <- filter.results(results = results, teams = teams2016, stichtag = stichtag2010, endtag = end2016)
model_2010_all <- model_data(results_2010$all)
model_2010_exclusive <- model_data(results_2010$exclusive)

# initialise estimates for training data with 83 teams (from 2010 onwards)
```

```
attack.params_2010 <- rep(.9, times=length(unique((results_2010$all$home_team))))
defence.params_2010 <- rep(-0.5, times=length(unique((results_2010$all$away_team))))
home.param_2010 <- 0.06
par.inits_2010 <- c(home.param_2010, attack.params_2010, defence.params_2010)

# initialise estimates for training data with only 24 participants
attack.params.e_2010 <- rep(.9, times=length(unique(results_2010$exclusive$home_team)))
defence.params.e_2010 <- rep(-0.5, times=length(unique(results_2010$exclusive$away_team)))
home.param.e_2010 <- 0.06
xi <- 0.05
par.inits.e_2010 <- c(home.param.e_2010, attack.params.e_2010, defence.params.e_2010)


# rename
names(par.inits_2010) <- c('HOME', paste('Attack', model_2010_all$teams, sep='.'),
                           paste('Defence', model_2010_all$teams, sep='.'))

names(par.inits.e_2010) <- c('HOME', paste('Attack', model_2010_exclusive$teams, sep='.'),
                             paste('Defence', model_2010_exclusive$teams, sep='.'))

# # estimate
#
# res_2010_all <- auglag(par= par.inits_2010, fn = optimiser, heq = constraint,
#                        model = model_2010_all, days = results_2010£all£days, xi.p = 0.1)
# res_2010_exclusive <- auglag(par= par.inits.e_2010, fn = optimiser, heq = constraint,
#                              model = model_2010_exclusive, days = results_2010£exclusive£days, xi.p = 0.1)
#
#
# # pars_2010 contains estimates for data trained on 83 teams from 2010 onwards
# pars_2010 <- matrix(res_2010_all£par[2:(2*length(model_2010_all£teams)+1)], ncol = 2)
# ha_2010 <- res_2010_all£par[1]
# colnames(pars_2010) <- c("Attack", "Defense")
# rownames(pars_2010) <- model_2010_all£teams
#
# pars24_2010 <- pars_2010[rownames(pars_2010) %in% teams2016,]
#
# # pars_2010_exclusive contains estimates for data trained on 24 teams from 2010 onwards
# pars_2010_exclusive <- matrix(res_2010_exclusive£par[2:(2*length(model_2010_exclusive£teams)+1)], ncol = 2)
# ha_2010_exclusive <- res_2010_exclusive£par[1]
# colnames(pars_2010_exclusive) <- c("Attack", "Defense")
# rownames(pars_2010_exclusive) <- model_2010_exclusive£teams
#
#
#
# # create matchvalue matrix for every possible game
#
# matchvals_2021 <- matchvals_2016 <- matrix(0,24,24)
# rownames(matchvals_2021) <- colnames(matchvals_2021) <- sort(teams2021)
# for(i in rownames(matchvals_2021)){
#   for(j in colnames(matchvals_2021)){
#     if(i != j){
#       matchvals_2021[i, j] <- exp(pars24_2015[i, "Attack"] +
#                                    pars24_2015[j, "Defense"])
#     }
#   }
# }
#
```

```
# rownames(matchvals_2016) <- colnames(matchvals_2016) <- sort(teams2016)
# for(i in rownames(matchvals_2016)){
#   for(j in colnames(matchvals_2016)){
#     if(i != j){
#       matchvals_2016[i, j] <- exp(pars24_2010[i, "Attack"] +
#                                     pars24_2010[j, "Defense"])
#     }
#   }
# }
#
#
# #-------------------------------------------------
# # Save results to be read in externally
# write.csv(pars_exclusive, "D:\\Surface Dokumente\\Uni\\6. Semester\\Seminar\\pars_ex_new.csv")
# save(pars_exclusive, pars_all, pars24, pars_2015, pars24_2015,
#      ha_2015, ha_exclusive, ha_all, file = "estimation.rda")
# save(pars_2010_exclusive, pars_2010, pars24_2010, ha_2010_exclusive, ha_2010, file = "estimation_2016.rda")
#
# save(matchvals_2016, matchvals_2021, file = "matchvals_new.rda")
# #-------------------------------------------------


# Prediction
#---------------------------------------------------------------------------
teams2021 <- c("Italy", "Wales", "Turkey", "Switzerland",
               "Russia", "Belgium", "Denmark", "Finland",
               "Austria", "Ukraine", "Netherlands", "North Macedonia",
               "Czech Republic", "England", "Croatia", "Scotland",
               "Sweden", "Spain", "Poland", "Slovakia",
               "Germany", "France", "Hungary", "Portugal"
)

teams2016 <- c("France", "Romania", "Albania", "Switzerland",
               "England", "Russia", "Wales", "Slovakia",
               "Germany", "Ukraine", "Poland", "Northern Ireland",
               "Spain", "Czech Republic", "Turkey", "Croatia",
               "Belgium", "Italy", "Republic of Ireland", "Sweden",
               "Portugal", "Iceland", "Austria", "Hungary")


teams <- teams2021

load("estimation.rda")
load("estimation_2016.rda")
load("matchvals_new.rda")






# for home advantage
# levels(spielplan£Location) <- list(Germany = "Fußball Arena München",
# Azerbaijan = "Baki Olimpiya Stadionu", Spain = "Estadio La Cartuja", Denmark = "Parken",
# Russia = "Saint Petersburg Stadium", England = "Wembley Stadium",
# Netherlands = "Johan Cruijff ArenA", Scotland = "Hampden Park",
# Italy = "Olimpico in Rome", Hungary = "Puskás Aréna", Ukraine = "Arena Nationala")
```

```r
# sim single euro
#-----------------------------------------------------

sim_euro <- function(year = 2020, output = "detailed"){

  ## output can either be "stage", "stage_sorted", "detailed" or "table_3rd"
  # stage will be needed for multiple simulations

  #reset data

  teams <- c("Italy", "Wales", "Turkey", "Switzerland",
             "Russia", "Belgium", "Denmark", "Finland",
             "Austria", "Ukraine", "Netherlands", "North Macedonia",
             "Czech Republic", "England", "Croatia", "Scotland",
             "Sweden", "Spain", "Poland", "Slovakia",
             "Germany", "France", "Hungary", "Portugal"
  )

  if(year != 2020){
    teams <- teams2016
  }

  # initialise groups
  groups <- list("A" = list("Teams" = teams[1:4],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))),
                 "B" = list("Teams" = teams[5:8],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))),
                 "C" = list("Teams" = teams[9:12],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))),
                 "D" = list("Teams" = teams[13:16],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))),
                 "E" = list("Teams" = teams[17:20],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))),
                 "F" = list("Teams" = teams[21:24],
                            "Table" = data.frame("Points" = rep(0,4),
                                                 "Goals" = rep(0,4), "GA" = rep(0,4))))

  for(i in 1:length(groups)){
    rownames(groups[[i]]$Table) <- teams[4*(i-1)+(1:4)]
  }

  if(year == 2020){
    matchvals <- matchvals_2021
  } else {
    matchvals <- matchvals_2016
  }
  #initialise match plan

  # initialise match plan

  teams_g <- data.frame("Team" = teams,"Groups" = paste("Group", rep(LETTERS[1:6], each = 4)))
```

```r
if(year == 2020){
  spielplan <- read.csv("EURO_2020_matches.csv", encoding = "UTF-8")
} else{
  spielplan <- read.csv("EURO_2016_matches.csv", encoding = "UTF-8")
}
spielplan$Away.Score <- spielplan$Home.Score <- numeric(51)
spielplan$Home.Team <- as.character(spielplan$Home.Team)
spielplan$Away.Team <- as.character(spielplan$Away.Team)
spielplan$Result <- NA


{# Simulate Group Stage

  # simulate matches
  for(i in 1:36){
    spielplan$Home.Score[i] <- rpois(1, matchvals[spielplan$Home.Team[i], spielplan$Away.Team[i]])
    spielplan$Away.Score[i] <- rpois(1, matchvals[spielplan$Away.Team[i], spielplan$Home.Team[i]])

    if(spielplan$Home.Score[i] > spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Home.Team[i]
    } else if(spielplan$Home.Score[i] < spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Away.Team[i]
    } else {
      spielplan$Result[i] <- "D"
    }
  }

  # update Tables
  for(i in 1:length(groups)){
    for(j in groups[[i]]$Teams){

      groups[[i]]$Table[j, "Goals"] <-
        sum(spielplan$Home.Score[spielplan$Home.Team == j], spielplan$Away.Score[spielplan$Away.Team == j])
      groups[[i]]$Table[j, "GA"] <-
        sum(spielplan$Away.Score[spielplan$Home.Team == j], spielplan$Home.Score[spielplan$Away.Team == j])
      groups[[i]]$Table[j, "GD"] <-
        groups[[i]]$Table[j, "Goals"] - groups[[i]]$Table[j, "GA"]

      games_team <- filter(spielplan[c("Home.Team", "Away.Team", "Home.Score", "Away.Score", "Result")],
                           Home.Team == j | Away.Team == j)
      groups[[i]]$Table[j, "Points"] <- sum((games_team$Result == j)*3, games_team$Result == "D")
    }

    #ordering Table (points, GD, Goals)
    groups[[i]]$Table <- groups[[i]]$Table[order(
      -groups[[i]]$Table$Points, -groups[[i]]$Table$GD, -groups[[i]]$Table$Goals),]

  }
}

# table of 3rd placed teams
table_3 <- data.frame()
for(i in 1:length(groups)){
  table_3 <- rbind(table_3,cbind(groups[[i]]$Table[3,], "Group" = names(groups[i])))
}
```

```
table_3 <- table_3[order(-table_3$Points, -table_3$GD, -table_3$Goals),]
groups_3rd <- sample(table_3$Group[1:4])


#-------------------------------------------------------------------------------------------------
# Different Round of 16 constellations (not implemented because it would not change that much)
#--------------------------------------------------------------------------------------------------
# if(all(groups_3rd %in% c("A", "B", "C", "D")) == TRUE){
#
#
# } else if(all(groups_3rd %in% c("A", "B", "C", "E")) == TRUE){
#
# } ......... (15 different constellations)

#---------------------------------------------------------------------------------------------
# Random Round of 16 allocation of 3rd placed teams
#-----------------------------------------------------------------------------------------------
{#Simulate round of 16
  { # create game plan

    spielplan[37, "Home.Team"] <- rownames(groups$A$Table)[2]
    spielplan[37, "Away.Team"] <- rownames(groups$B$Table)[2]

    spielplan[38, "Home.Team"] <- rownames(groups$A$Table)[1]
    spielplan[38, "Away.Team"] <- rownames(groups$C$Table)[2]

    spielplan[39, "Home.Team"] <- rownames(groups$C$Table)[1]
    spielplan[39, "Away.Team"] <- rownames(groups[[groups_3rd[1]]]$Table)[3]

    spielplan[40, "Home.Team"] <- rownames(groups$B$Table)[1]
    spielplan[40, "Away.Team"] <- rownames(groups[[groups_3rd[2]]]$Table)[3]

    spielplan[41, "Home.Team"] <- rownames(groups$D$Table)[2]
    spielplan[41, "Away.Team"] <- rownames(groups$E$Table)[2]

    spielplan[42, "Home.Team"] <- rownames(groups$F$Table)[1]
    spielplan[42, "Away.Team"] <- rownames(groups[[groups_3rd[3]]]$Table)[3]

    spielplan[43, "Home.Team"] <- rownames(groups$D$Table)[1]
    spielplan[43, "Away.Team"] <- rownames(groups$F$Table)[2]

    spielplan[44, "Home.Team"] <- rownames(groups$E$Table)[1]
    spielplan[44, "Away.Team"] <- rownames(groups[[groups_3rd[4]]]$Table)[3]
  }


  # simulate matches
  for(i in 37:44){
    spielplan$Home.Score[i] <- rpois(1, matchvals[spielplan$Home.Team[i], spielplan$Away.Team[i]])
    spielplan$Away.Score[i] <- rpois(1, matchvals[spielplan$Away.Team[i], spielplan$Home.Team[i]])

    if(spielplan$Home.Score[i] > spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Home.Team[i]
    } else if(spielplan$Home.Score[i] < spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Away.Team[i]
    } else {
      spielplan$Result[i] <- sample(c(spielplan$Home.Team[i], spielplan$Away.Team[i]), 1,
                                   prob = c(matchvals[spielplan$Home.Team[i],
                                                      spielplan$Away.Team[i]]/
                                              (sum(matchvals[spielplan$Home.Team[i],
                                                             spielplan$Away.Team[i]],
                                                   matchvals[spielplan$Away.Team[i],
                                                             spielplan$Home.Team[i]])),

                                            matchvals[spielplan$Away.Team[i],
                                                      spielplan$Home.Team[i]]/
                                              (sum(matchvals[spielplan$Home.Team[i],
                                                             spielplan$Away.Team[i]],
                                                   matchvals[spielplan$Away.Team[i],
                                                             spielplan$Home.Team[i]]))))
    }
  }
}


{
  #simulate quarter finals

  { # create game plan

    spielplan[45, "Home.Team"] <- spielplan[39, "Result"]
    spielplan[45, "Away.Team"] <- spielplan[37, "Result"]

    spielplan[46, "Home.Team"] <- spielplan[41, "Result"]
    spielplan[46, "Away.Team"] <- spielplan[42, "Result"]

    spielplan[47, "Home.Team"] <- spielplan[43, "Result"]
    spielplan[47, "Away.Team"] <- spielplan[44, "Result"]

    spielplan[48, "Home.Team"] <- spielplan[40, "Result"]
    spielplan[48, "Away.Team"] <- spielplan[38, "Result"]



  }

  # simulate games

  for(i in 45:48){
    spielplan$Home.Score[i] <- rpois(1, matchvals[spielplan$Home.Team[i], spielplan$Away.Team[i]])
    spielplan$Away.Score[i] <- rpois(1, matchvals[spielplan$Away.Team[i], spielplan$Home.Team[i]])

    if(spielplan$Home.Score[i] > spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Home.Team[i]
    } else if(spielplan$Home.Score[i] < spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Away.Team[i]
    } else {
      spielplan$Result[i] <- sample(c(spielplan$Home.Team[i], spielplan$Away.Team[i]), 1,
                                   prob = c(matchvals[spielplan$Home.Team[i],
                                                      spielplan$Away.Team[i]]/
                                              (sum(matchvals[spielplan$Home.Team[i],
                                                             spielplan$Away.Team[i]],
                                                   matchvals[spielplan$Away.Team[i],
                                                             spielplan$Home.Team[i]])),
                                             matchvals[spielplan$Away.Team[i],
                                                       spielplan$Home.Team[i]]/
                                               (sum(matchvals[spielplan$Home.Team[i],
                                                              spielplan$Away.Team[i]],
                                                    matchvals[spielplan$Away.Team[i],
                                                              spielplan$Home.Team[i]]))))
    }
  }



}

{
  #simulate semi finals

  { # create game plan

    spielplan[49, "Home.Team"] <- spielplan[46, "Result"]
    spielplan[49, "Away.Team"] <- spielplan[45, "Result"]

    spielplan[50, "Home.Team"] <- spielplan[48, "Result"]
    spielplan[50, "Away.Team"] <- spielplan[47, "Result"]

  }

  # simulate games

  for(i in 49:50){
    spielplan$Home.Score[i] <- rpois(1, matchvals[spielplan$Home.Team[i], spielplan$Away.Team[i]])
    spielplan$Away.Score[i] <- rpois(1, matchvals[spielplan$Away.Team[i], spielplan$Home.Team[i]])

    if(spielplan$Home.Score[i] > spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Home.Team[i]
    } else if(spielplan$Home.Score[i] < spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Away.Team[i]
    } else {
      spielplan$Result[i] <- sample(c(spielplan$Home.Team[i], spielplan$Away.Team[i]), 1,
                                    prob = c(matchvals[spielplan$Home.Team[i],
                                                       spielplan$Away.Team[i]]/
                                               (sum(matchvals[spielplan$Home.Team[i],
                                                              spielplan$Away.Team[i]],
                                                    matchvals[spielplan$Away.Team[i],
                                                              spielplan$Home.Team[i]])),

                                             matchvals[spielplan$Away.Team[i],
                                                       spielplan$Home.Team[i]]/
                                               (sum(matchvals[spielplan$Home.Team[i],
                                                              spielplan$Away.Team[i]],
                                                    matchvals[spielplan$Away.Team[i],
                                                              spielplan$Home.Team[i]]))))
    }
  }
```

```
}

{
  #simulate final

  { # create game plan

    spielplan[51, "Home.Team"] <- spielplan[49, "Result"]
    spielplan[51, "Away.Team"] <- spielplan[50, "Result"]

  }

  # simulate games

  for(i in 51){
    spielplan$Home.Score[i] <- rpois(1, matchvals[spielplan$Home.Team[i], spielplan$Away.Team[i]])
    spielplan$Away.Score[i] <- rpois(1, matchvals[spielplan$Away.Team[i], spielplan$Home.Team[i]])

    if(spielplan$Home.Score[i] > spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Home.Team[i]
    } else if(spielplan$Home.Score[i] < spielplan$Away.Score[i]){
      spielplan$Result[i] <- spielplan$Away.Team[i]
    } else {
      spielplan$Result[i] <- sample(c(spielplan$Home.Team[i], spielplan$Away.Team[i]), 1,
                                    prob = c(matchvals[spielplan$Home.Team[i],
                                                       spielplan$Away.Team[i]]/
                                               (sum(matchvals[spielplan$Home.Team[i],
                                                              spielplan$Away.Team[i]],
                                                    matchvals[spielplan$Away.Team[i],
                                                              spielplan$Home.Team[i]])),

                                             matchvals[spielplan$Away.Team[i],
                                                       spielplan$Home.Team[i]]/
                                               (sum(matchvals[spielplan$Home.Team[i],
                                                              spielplan$Away.Team[i]],
                                                    matchvals[spielplan$Away.Team[i],
                                                              spielplan$Home.Team[i]]))))
    }
  }



}

stage <- character(length(teams))
names(stage) <- teams

for(s in teams){
  max_game <- max(which(s == spielplan[c("Home.Team", "Away.Team")], arr.ind = TRUE)[,"row"])
  stage[s] <- as.character(spielplan$Round.Number[max_game])
}

stage[spielplan[51, "Result"]] <- "Winner"
stage[stage == "3"] <- "Group Stage"

sorting <- c("Winner", "Final", "Semi Finals", "Quarter Finals", "Round of 16", "Group Stage")
```

```
  stage_sorted <- stage[order(match(stage, sorting))]




  detailed <- list("Group Stage" = list("Games" = spielplan[1:36, ],
                                        "Table" = groups),
                   "Final Rounds" = list("Games" = spielplan[37:51, ],
                                         "End Result" = stage_sorted),
                   "Table 3rd" = table_3
  )

  return(get(output))


}



sim_multiple_euro <- function(year = 2020,n_sim = 100){

  if(year != 2020){
    teams <- teams2016
  }

  results_teams <- matrix(NA, nrow = length(teams), ncol = n_sim)
  rownames(results_teams) <- teams
  colnames(results_teams) <- 1:n_sim


  for(k in 1:n_sim){
    results_teams[,k] <- sim_euro(year, "stage")
  }

  return(results_teams)
}

# simulations_100k <- sim_multiple_euro(2020,n_sim = 100000)


# save(simulations_100k ,  file = "sim_100k.rda")
# save(sim_euro, file = "sim_euro.rda")
# save(sim_multiple_euro, file = "sim_multiple_euro.rda")

# Validation
#--------------------------------------

# Create probability matrix of results
score_probability <- function(T1 = "Austria", T2 = "Germany", year = 2020){

  if(year == 2016){
    matchvals <- matchvals_2016
  } else {
    matchvals <- matchvals_2021
  }
  mat <- matrix(0, 6, 6)
  colnames(mat) <- rownames(mat) <- c(0:4, ">4")
  for(i in 0:5){
    for(j in 0:5){
      # last row and column is probability of scoring > 4 goals, therefore upper tail of distribution function
      if(i == 5){
        mat[i+1, j+1] <- ppois(i-1, matchvals[T1, T2], lower.tail = FALSE) * dpois(j, matchvals[T2,T1])
      } else if(j == 5){
        mat[i+1, j+1]<- dpois(i, matchvals[T1, T2]) * ppois(j-1, matchvals[T2,T1], lower.tail = FALSE)
      } else { # remaining entries density
        mat[i+1, j+1] <- dpois(i, matchvals[T1, T2])*dpois(j, matchvals[T2, T1])
      }
      # very last entry
      mat[6, 6] <- ppois(4, matchvals[T1, T2], lower.tail = FALSE) *
        ppois(4, matchvals[T2,T1], lower.tail = FALSE)
    }
  }

  draw <- sum(diag(mat))
  p1 <- sum(mat[lower.tri(mat)])
  p2 <- sum(mat[upper.tri(mat)])

  df <- data.frame(T1 = p1,
                   "D" = draw,
                   T2 = p2)
  colnames(df) <- c(T1, "D", T2)

  max.ind <- which(mat == max(mat), arr.ind = TRUE)
  score.max.prob <- paste0(max.ind[1]-1, " - ", max.ind[2]-1)
  max.prob <- mat[max.ind[1], max.ind[2]]

  return(list("Probability matrix" = mat,
              "Outcome probability" = df,
              "Most likely result" = score.max.prob
  ))



}

# Create list with prob matrix and most likely result and outcome probabilities
prob.list.2016 <- list()
for(nation1 in teams2016){
  for(nation2 in teams2016){
    if(nation1 == nation2){
      next
    }
    prob.list.2016[[paste(nation1, "-", nation2)]] <- score_probability(nation1, nation2, year = 2016)
  }
}

prob.list.2020 <- list()
for(nation1 in teams2021){
  for(nation2 in teams2021){
    if(nation1 == nation2){
      next
    }
    prob.list.2020[[paste(nation1, "-", nation2)]] <- score_probability(nation1, nation2)
  }
```

```
}

# kable(
#   t5,
#   format = "latex",
#   booktabs = TRUE,
#   linesep = '',
#   caption = "Results vs. Prediction EURO 2016", label = "EURO2016_pred"
# ) #%>%
 # kable_styling(latex_options = c("scale_down"))
print(EURO2016_xtable, caption.placement = "top", table.placement = "H",
  hline.after=NULL,
  add.to.row=list(pos=list(-1, 0, nrow(EURO2016_xtable)),
  command=c('\\toprule\n','\\midrule\n','\\bottomrule\n')))
```