Data 2010 Final Project Report

Tom Rodov, Daniel Paul, Owen Ostermann

April 9, 2023

Abstract

FIFA, or the Fédération Internationale de Football Association, is the governing body of international soccer. It was founded in 1904 and has since grown to include 211 national associations. FIFA is responsible for organizing international tournaments such as the World Cup, which is held every four years and is considered the most prestigious soccer competition in the world. The tournament brings together the best soccer teams from around the globe to compete for the title of world champion.

One of the most exciting aspects of the World Cup is predicting the outcome of matches. Many people enjoy betting on the tournament, but the unpredictable nature of soccer makes it challenging to accurately predict match outcomes. In this research paper, we analyze FIFA World Cup data to develop a model for predicting future match outcomes, specifically the final scores of games.

Our central research question is how to use the available data to accurately predict future matches based on the previous performances of the teams. To achieve this, we use the Kolmogorov–Smirnov test to confirm the distribution of our data and the Elo ranking system to rank the teams and predict the scores of future matches. In this system, we utilize the logistic function, and we optimize the best parameter in the logistic function using the Maximum Likelihood Estimation (MLE) process.

Our findings suggest that our approach provides a more accurate way to predict the outcome of FIFA matches. The implications of our findings are significant as our approach will provide individuals with a more informed way to bet on FIFA matches, increasing their chances of success. We recommend further research to refine the predictive model and expand the dataset to include additional variables such as more matches and player injuries.

In conclusion, Our approach will provide individuals with a more informed way to bet on FIFA matches, and it has the potential to enhance the overall experience of the World Cup for soccer fans worldwide.

The Poisson Distribution

The Poisson distribution is a discrete probability distribution that models the number of rare events that occur within a fixed interval of time and it is especially useful for predicting the number of goals scored in a soccer game, as goals are rare events that occur within the timed interval of the game.

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$$

The Poisson distribution is characterized by a single parameter, λ, which represents the average number of these events (goals scored) occurring in the given interval. By analyzing historical data in the World Cup dataset, we can estimate the λ values of different teams and use the Poisson distribution to calculate the probability of a team scoring a particular number of goals in a soccer game.

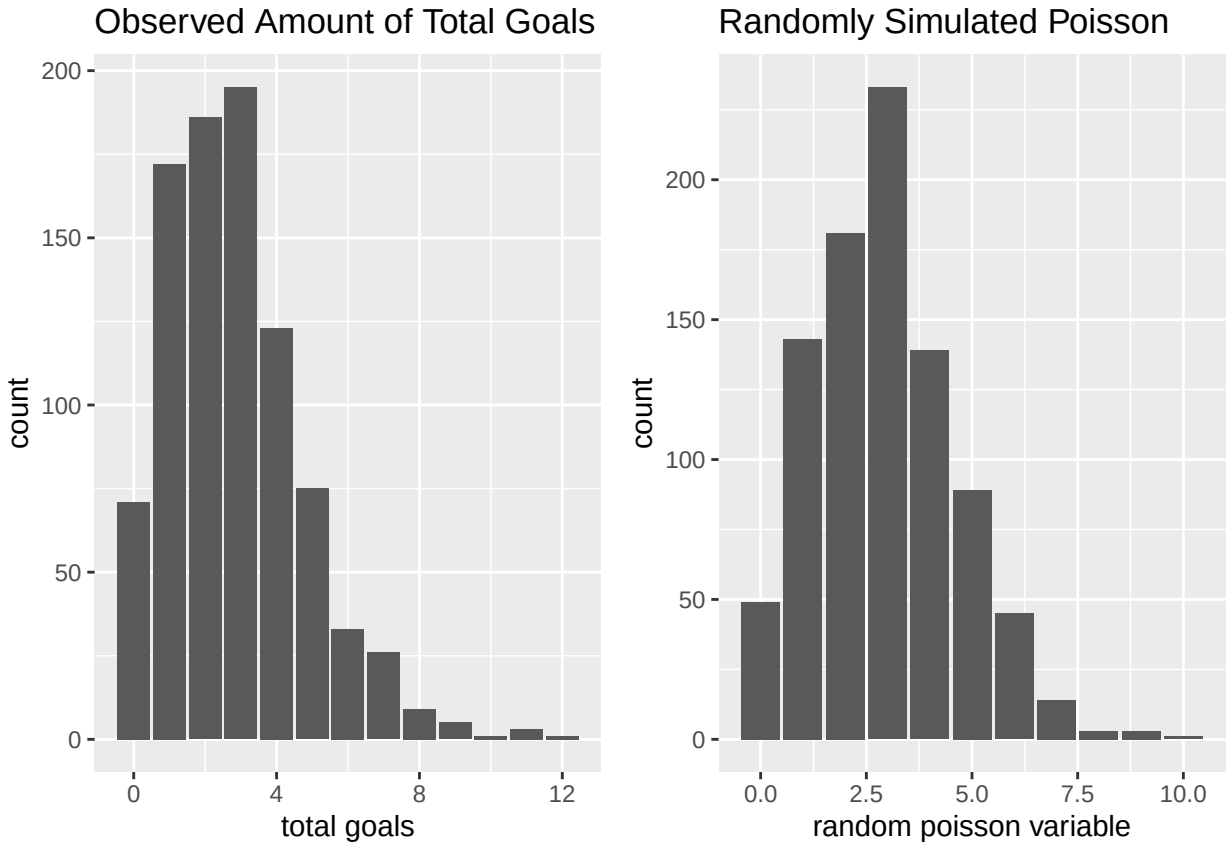
The joint probability function of two discrete random variables X and Y is given by $P(X = x, Y = y)$ and describes the probability that $X = x$ and $Y = y$ simultaneously. In our case X is the home team and Y is the away team, and the amount of goals they score is initially treated as independent. Our model will update the likelihood of the amount of goals in a game given the skill rankings of the two teams playing. The home team, x, takes the values in its range R_x, obtained from the past data of what values of x the team has obtained prior. The away team follows the same strategy.

Testing The Distribution of Goals Scored

Upon importing the dataset, which comprises a vast collection of information regarding World Cup tournaments throughout history, we have identified a few relevant variables for our study. Specifically, we have limited our analysis to the variable that records the number of goals scored in a given game. After applying a filtering process, we have obtained a refined dataset that is suitable for our research objectives.

After plotting the total goals scored in a geom bar plot, it was observed that the distribution appeared to follow a Poisson distribution. This finding is expected, as the Poisson distribution expresses the probability of a given number of events occurring in a fixed interval of time or space, in this case, goals in the amount of time a soccer game takes. Knowing the distribution of a variable can provide valuable insights into its behavior, help us make informed decisions based on this understanding, and guide us in choosing appropriate statistical methods for analysis.

In order to test the hypothesis that the data indeed follows a Poisson distribution, we calculated the mean of the data of total goals scored in a game, which serves as the expected value of the Poisson distribution with lambda equal to 2.831. A random sample of 900 observations was generated from the Poisson distribution with the same lambda, and the resulting distribution was graphed to compare with the original data.



The Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test is a non-parametric statistical test that compares the empirical distribution of a data sample with a known probability distribution. In our case, we use the Kolmogorov-Smirnov test to compare the empirical distribution of the observed data with the Poisson distribution.

$$H_0 \text{ (null hypothesis)} \rightarrow \text{ The two samples are from the same distribution}$$

$$H_A \text{ (alternative)} \rightarrow \text{ The two samples are from different distributions}$$

$$p \text{ - value} = 0.1243 \rightarrow \text{ There is no significant difference in the two samples}$$

The significance level was set at 0.05. If the p-value is greater than 0.05, we fail to reject the null hypothesis, indicating that there is no significant difference between the two samples at the 95% confidence level.

The result of the Kolmogorov-Smirnov test yielded a p-value of 0.1243. Thus, we fail to reject the null hypothesis and conclude that the two samples are from the same distribution. Letting us work with certainty that our data follows the Poisson distribution. Knowing the distribution of a variable can provide valuable insights into its behavior, help us make informed decisions based on this understanding, and guide us in choosing appropriate statistical methods for analysis. For example, if we know that a variable follows a normal distribution, we can make probabilistic statements about the variable, such as the likelihood of a certain value occurring, or the range of values that are expected to occur with a certain probability. (Appendix 1.1)

The same method was applied to analyze the difference in goals scored and the number of total goals scored by France, and it was found that both of these variables also followed a Poisson distribution.

The Elo Ranking System

The Elo ranking system is a method used to rank and rate the relative skill levels of soccer teams. It assigns a numerical ranking to each team that is based on their performance in previous games played. The rating is continuously updated after each game played based on the final outcome of the match and the relative strength of the two teams in the match. Teams with higher rankings are considered stronger and thus more likely to win, whereas teams with a lower rating are considered to be weaker and thus less likely to win.

When two teams with different Elo ratings play each other, the system calculates the expected outcome of the match based on the difference in their ratings. If the higher-rated team wins, they will gain fewer points than if they had won against a team with a similar rating, while the lower-rated team will lose fewer points than if they had lost against a team with a similar rating. The opposite is true if the lower-rated team wins.

Over time, the Elo system can be used to track the progress of teams and to predict the outcomes of future matches, specifically this can help to aid in adjusting the probabilities of the amount of goals scored when two teams play each other. When a team with a high Elo ranking plays a team with a low Elo ranking it is likely that the probability of the better team scoring more goals should be adjusted upwards based on the difference in skill level. This is where the Elo ranking system for the teams in the World Cup dataset will be of vast importance. Overall, the Elo ranking system is a popular method for ranking soccer teams and is used by many professional leagues and tournaments around the world.

Finding The Unique Teams in the World Cup Data

First we filter the results only in the dataset. This is so that when we have the unique teams and we set up the Elo model, we can run through this data frame and update the elo rankings based on the historical results of the World Cup games. (Appendix 2.1)

home_team	home_score	away_team	away_score	winner	home_win	away_win
France	4	Mexico	1	H	1.00	0.00
Belgium	0	United States	3	A	0.00	1.00
Brazil	1	Yugoslavia	2	A	0.00	1.00
Peru	1	Romania	3	A	0.00	1.00

Something to notice that after World War II, Germany was divided into East and West Germany until 1990. Thus between those years Germany is split into two teams. Since the players are still from Germany, we are just going to replace West and East Germany as just Germany. The rankings are based on over 100 years of games. The players are not the same and teams can get better or worse over time depending on the players representing the country at that time. But what we want our ranking to encompass is on average which countries are producing good or bad teams. Thus, in this case players are still from Germany, so we combine the West and East teams. We need to do this for Russia and the Soviet Union as Russia was known as the the Soviet Union from 1922 until 1991. We need to do this for the Republic of Ireland and change it to Ireland as the name was changed from Republic of Ireland to Ireland in 1937. We have to do this for FR Yugoslavia and change it to just Yugoslavia as well, which is now referred to as just Serbia and Montenegro.

Now another problem arises when dealing with the split of Czechoslovakia into Czech Republic and Slovakia. How do we account for the split of the country Czechoslovakia into two new countries in 1992. A solution for this split is to just update both Slovakia and Czech Republics Elo's when the game has Czechoslovakia in it and then just update Slovakia's Elo for games with Slovakia in it and Czech Republic's Elo when its just Czech Republic playing in the match. (Appendix 2.2)

Since we have accounted for the duplicates and same names in the results of matches dataframe. We now need to select only the unique teams, meaning each team in the World Cup dataset should only be displayed once in a new data frame. This data frame is going to include the team name and the Elo ranking and as we simulate through the results data the teams data will be continuously updated with the teams new Elo based on the outcome of the match.

In order for the algorithm to work, we have to pre set a Elo ranking number for each team, this number then gets adjusted based on the outcomes of matches in the games. We initially set each teams Elo ranking to 1500. (Appendix 2.3)

country	ELO
Algeria	1500.00
Angola	1500.00
Argentina	1500.00
Australia	1500.00
Austria	1500.00
Belgium	1500.00
Bolivia	1500.00
Bosnia and Herzegovina	1500.00
Brazil	1500.00
Bulgaria	1500.00

Now our data is prepared and ready to loop through all of the World Cup games given and update each participating teams Elo ranking based on the outcome of the game.

How The Elo Ranking Algorithim Works

The Elo ranking system follows the formula below:

$$r'(A) = r(A) + k(S_A - \mu_A)$$

The value $\mathbf{r(A)}$ represents the previous score for team A. $\mathbf{r'(A)}$ represents the updated score for team A. $\mathbf{S_A}$ is the result of the match. $S_A = 1$ represents a win. $S_A = -1$ represents a loss. $\mu_\mathbf{A}$ is the expected result of the game between the two teams. This is in the form of a probability. $\mathbf{K > 0}$ is a parameter of our choice. This is the score adjustment. A larger value of K will increase the rankings more than a smaller choice of K.

As we stated above, $\mu_\mathbf{A}$, is a probability value, it is defined in the following way:

$$\mu_A = (1) * P(\text{A beats B}) + (-1) * P(\text{B beats A})$$

The probabilities above between team A and B depend on the skill difference between the two teams. In other words $r(A) - r(B)$. In order to transform μA into a probability value we need to use what is known as the **logistic function**.

The logit function is a mathematical function used in statistics and machine learning that uses one or more predictor values, in our case the Elo rankings of the two teams, to predict the probability of an outcome. In the context of soccer and the World Cup dataset, the logistic function can be used to predict the probability of a win, loss, or draw based on the Elo ratings of two teams that are facing each other. The function to convert μ_A into a probability value between 0 and 1 is:

$$\mu_A = \frac{1}{1 + e^{(-C*(r(A)-r(B)))}}$$

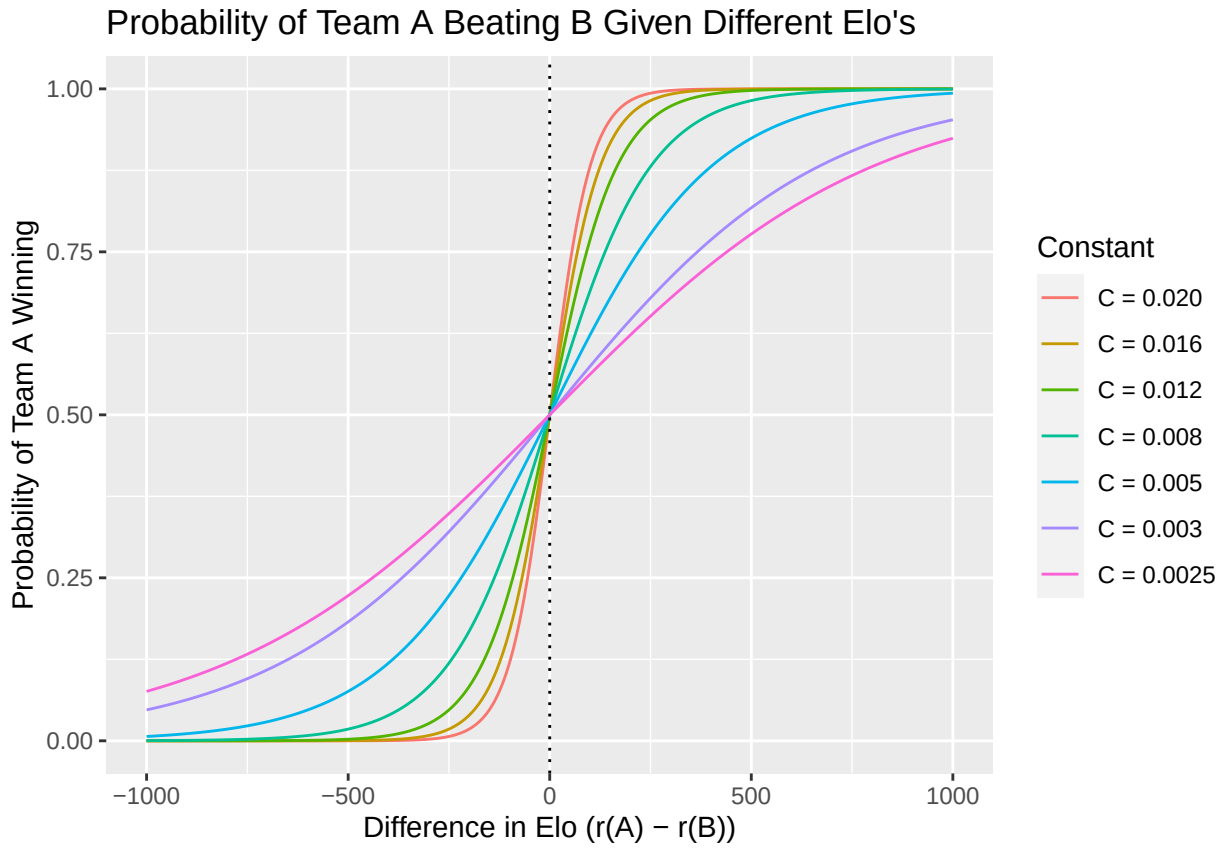
Here C is a constant that can be calibrated by using a training dataset. Some studies have used a value of 0.003 to 0.005 for soccer match predictions, but finding an optimal value for our specific dataset depends on the data itself.

The first step is to find a K value in the formula $r'(A) = r(A) + k(S_A - \mu_A)$. This value determines the score adjustment, in other words how big or small a change in a teams score will be based on the outcome of a game. A larger K value would result in quicker and larger changes in a teams score, this is good if our goal is to quickly figure out and identify the strongest teams on a smaller amount of data. A smaller K value would result in a more gradual change in ratings and would be appropriate if the goal was to provide a more stable and long term ranking system. Here we are going to use a value of $K = 60$, as this is the FIFA convention for games played in the World Cup. The Elo score formula then becomes the following:

$$r'(A) = r(A) + 60(S_A - \mu_A)$$

Estimating Parameters in the Logistic Function

Now that we have the K value from the FIFA standards, the next thing we need to solve is finding the C value in the logistic function. First lets observe a plot of the logistic function with some sample constants. Here we plot the difference in Elo rankings versus the probability of team A winning. We set team B's Elo ranking to a value of 1500 for simplicity. The X-axis ranges from -1000 to 1000, this is the difference in the Elo ranking that team A is from team B. We can observe that for all constants when team A has a negative difference from team B's Elo ranking (they have a lower Elo than team B), their probability of winning a game against team B is low. When team A and team B have no difference in Elo rankings, the probability of winning is even, thus 50% for both teams. All constants centralize to this point. This is because we have $e^0 = 1$, so we end up getting $\mu_A = 0.5$. When team A has a higher Elo ranking than team B, the probability of winning is much higher. In summary, the more negative the difference in Elo, the probability of team A winning converges to 0. The more positive the difference in Elo, the probability of team B winning converges to 1. We can see how the logistical function models this well. Below is a plot of the logistical function with different constant values, the output is the probability that team A wins given that team B's Elo ranking is 1500. Team A's elo ranking ranges from 500 to 2500 in this plot, so we show the different probabilities based on the differences in Elo rankings. (Appendix 2.4)



Now that we have a general idea on the shape of the logistic function we need estimate and calibrate the value of C. We are going to use process of Maximum Likelihood Estimation (MLE). MLE is a method of estimating parameters of a probability distribution, given some observed data. In MLE we maximize a likelihood function so that under the assumed statistical model, the observed data is most likely to occur. The point in the parameter space that maximizes the likelihood function is known as the maximum likelihood estimate. In our case, we need to use some sample results of games and Elo rankings in order to find and estimate a C value in the logistic function. We need to find the log likelihood function and use the *optim* function in R to optimize and find the best C value for the data. Once we have found a C value that is

calibrated and works well with the results of the FIFA soccer games, we then can finally run the simulation on the entire results data set from the FIFA World Cup and get the Elo rankings of the teams over time.

We take the current Elo rankings of countries soccer teams from a publicly open website "World Football Elo Ratings" (http://eloratings.net). This data is in very messy format. We will have to convert it first to a format that we can use. (Appendix 2.5)

Our data looks like this initially:

V1
1
Argentina
2143
2
Brazil
2134
3
France
2082

After cleaning we get it into this form that we can use:

rank	country	elo
1	Argentina	2143.00
2	Brazil	2134.00
3	France	2082.00
4	Netherlands	2073.00
5	Portugal	1999.00

We then join the current Elo rankings on the results dataframe and include the games in which we have Elo rankings from the real world. We combine this with the results rankings to get a data frame that includes the results of the games and the Elo rankings of the teams. We can now start the estimation process. (Appendix 2.6,2.7)

home_team	home_score	away_team	away_score	winner	home_elo	away_elo
France	4	Mexico	1	H	2082.00	1813.00
Peru	1	Romania	3	A	1851.00	1610.00
Argentina	1	France	0	H	2143.00	2082.00
Chile	3	Mexico	0	H	1697.00	1813.00
Uruguay	1	Peru	0	H	1905.00	1851.00

After the optimization is complete we get $C = 0.0028035$. So we have the following formulas that can be used for the ranking system:

(Appendix 2.8)

$$\mu_A = \frac{1}{1 + e^{(-0.0028035 * (r(A) - r(B)))}}$$

$$r'(A) = r(A) + 60(S_A - \mu_A)$$

Because Czechoslovakia was split into Slovakia and Czech Republic, we need to treat Czechoslovakia as the two countries. So we calculate the Elo for Czechoslovakia up until the last occurrence of Czechoslovakia (when the split occurred), then we update both Slovakia and Czech Republic's Elo to what Czechoslovakia's was and then treat them as independent. (Appendix 2.9)

country	ELO
Brazil	1669.98
Germany	1652.37
Netherlands	1584.32
Turkey	1496.79
France	1495.16
Cuba	1471.21
Dutch West Indies	1470.16
Jamaica	1465.66
Spain	1454.21
Bosnia and Herzegovina	1451.44
Ukraine	1435.14
Croatia	1426.02
Wales	1425.20
Haiti	1421.31
Israel	1420.88

Adjusting Lambda Based on Elo

Now since we have the Elo rankings of the teams from our dataset (Appendix 2.10), we need to figure out a way to adjust the probabilities before presenting the updated contingency table of outcomes of goals. We know that each team has a λ value, a mean value of goals scored in a game (follows a poisson distribution) based on past games in the World Cup dataset. When two teams play each other their likelihood of scoring goals changes depending on the relative Elo rankings of the teams playing. For example if Brazil who has a Elo ranking of 1669.98 based on our model is playing South Korea who has an elo ranking of 1046.497. Brazil's likelihood of scoring more goals should go up and South Koreas should go down. We already know the probabilities based on each teams λ and then just multiplying their marginal distribution probabilities together to get a table of probabilities that defines the likelihood of certain final scores in a game. But from the example above, these probabilities need to be slightly changed based on the difference in elo rankings, right now these probabilities are independent of elo rankings. A way to adjust the probabilities is by using the following formula.

$$\lambda_A^{`} = \lambda_A + K(A_{ELO} - B_{ELO})$$

This formula adjusts team A's λ value based on their elo ranking and the other teams elo ranking. It is multiplied by a constant value K, in which we will need to use Maximum Likelihood Estimation to find the optimal value K.

To use MLE to estimate the best value of K, we need to first define the likelihood function that captures the relationship between the observed data and the parameter K we want to estimate. Given that the goal scoring of the two teams playing follows a Poisson distribution, the likelihood function can be defined as:

$$L(\lambda; y) = \Pi(\lambda^{y_i} * e^{-\lambda})/y_i!$$

Where Y_i is the number of goals scored by team i. λ_i is the mean value of goals scored by team i. We then take the logarithm of the likelihood function.

$$logL(\lambda; y) = \sum(y_i * log(\lambda) - \lambda - log(y_i!))$$

Where the summation is over all of the observations, in our case the games played for a team. Lets try to find a K value that is optimal for data on Brazil. (Appendix 2.11)

We get the following formula to adjust Brazil's lambda based on the difference in elo between them and the team they are playing.

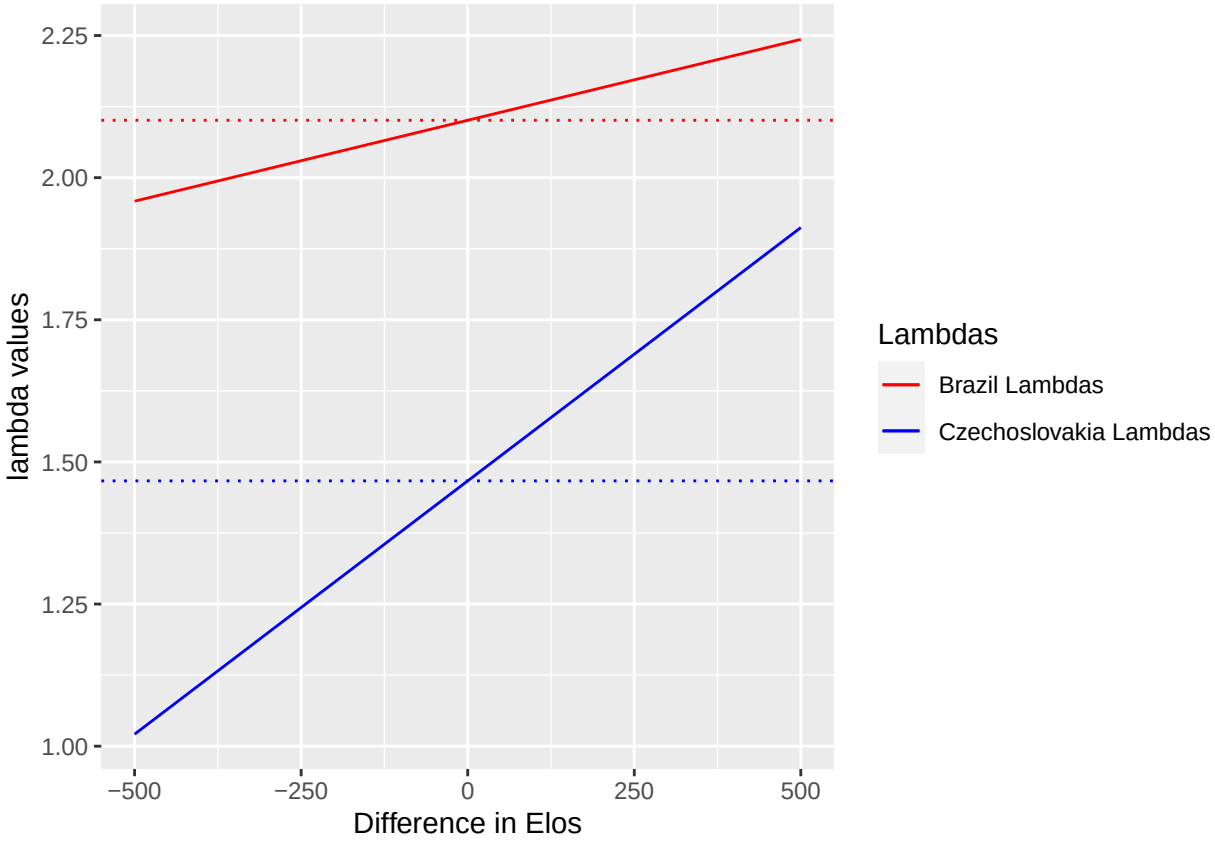
$$\lambda^{'}_{Brazil} = \lambda_{Brazil} + 2.8457642 \times 10^{-4}(Brazil_{ELO} - Opponent_{ELO})$$

We repeat the same process for Czechoslovakia. (Appendix 2.12)

$$\lambda^{'}_{Czechoslovakia} = \lambda_{Czechoslovakia} + 8.9111328 \times 10^{-4}(Czechoslovakia_{ELO} - Opponent_{ELO})$$

When Brazil plays Czechoslovakia , the new λ value is 2.195533 , which has increased from 2.1009174. On the flip side, when Czechoslovakia plays Brazil and wins, the new λ value is 1.1703906 decreased from 1.4666667 meaning they are less likely to score since they are playing against a better team. The plot below shows how the different teams lambda chnages base don the differences in elo values of a given opponent. The line comes from the formula:

$$\lambda^{'}_{A} = \lambda_{A} + K(A_{ELO} - B_{ELO})$$



(Appendinx 2.13)

We can now use these adjusted lambda values in order to construct a probability table that will show the probability of goals occurring in a game. These probabilities will be more accurate than if we were to use the respective teams non-adjusted lambdas.

In this table the left hand row values represent the probability of the amount of goals Brazil will score, and the values on the top, the probability of the amount of goals Czechoslovakia will score, the number in the spot (1,1) for example represents the probability of the game ending with the score of 1-1. (Appendix 3.1)

	0	1	2	3	4	5	6	7
0	0.03	0.08	0.08	0.06	0.03	0.01	0.01	0.00
1	0.04	0.09	0.10	0.07	0.04	0.02	0.01	0.00
2	0.02	0.05	0.06	0.04	0.02	0.01	0.00	0.00
3	0.01	0.02	0.02	0.02	0.01	0.00	0.00	0.00
4	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Validation

The log-likelihood formula is used to measure the fit of a probability distribution to the observed data. In our context, the observed data consists of the 5 games played between Brazil and Czechoslovakia. We specifically chose these two countries because they have played each other the most times, providing us with the highest number of sample points.

To begin, we have a probability distribution, P, that represents all the possible scores that can occur based on the maximum number of goals each team has scored previously. The probabilities of observing the outcomes in the observed data are represented by s1k and s2k, and these probabilities are summed over all observations in the dataset. Our goal is to maximize the log-likelihood, as this output represents the degree to which the probability distribution fits the actual scores. When one score is more positive than another, it indicates that the log-likelihood shows a better fit of the probability distribution.

$$\mathcal{L}(\boldsymbol{P}|\boldsymbol{s}) = \sum_{k=1}^{n} \log P_{s_{1,k},s_{2,k}}$$

The cross-entropy is a measure that quantifies the degree of similarity between the true probability distribution of soccer scores, which is based on the actual frequency of the scores observed in a limited dataset, and the estimated probability distribution generated by the model. The formula accounts for the differences between the two distributions and thus provides a way to evaluate whether the predictions made by the model are a good fit for the observed data.

A lower cross-entropy value indicates that the two probability distributions are closer to each other, meaning that the estimated probability distribution is more similar to the actual frequency distribution of soccer scores. Therefore, a lower cross-entropy value suggests that the model is providing more accurate predictions and is a better fit for the observed data.

$$\mathcal{H}(P,Q) = -\frac{1}{n}\sum_{i=1}^{n} \log Q_{s_{1,i},s_{2,i}}$$

One important point to consider is that neither of these measures provides meaningful information on its own. Rather, two models must be compared to assess the relative effectiveness of each measure. In this context, we compare the probability distribution of lambda values before and after the Elo adjustment. (Appendix 3.2)

Before the elo adjustments the values are:

```
## Log-likelihood: -72.66821
```

```
## Cross-entropy: 14.53364
```

After the adjustments:

```
## Log-likelihood: -69.9949
```

```
## Cross-entropy: 13.99898
```

Observations indicate that the log-likelihood is more positive for the adjusted lambda value distribution, which suggests that the Elo adjustment has resulted in a better fit for the observed data. Similarly, it is observed that the cross-entropy is lower for the adjusted lambda value distribution, which indicates that the Elo adjustment has provided a more accurate value.

Using the model for sports betting

With the probability table we got earlier, we are now poised to make more accurate predictions on future games, the most effective way to use this is to place a point spread bet. Bettors who choose the underdog win their wager when that team either wins the event outright OR loses by an amount less than the point spread.

An example of a point spread bet for a hypothetical Brazil vs Czechoslovakia soccer game:

Point Spread: Brazil -1.5 (+110) vs Czechoslovakia +1.5 (-130)

In this scenario, Brazil is the favorite to win the game by a margin of at least two goals, which means they are given a point spread of -1.5. Czechoslovakia, on the other hand, is the underdog and is given a point spread of +1.5.

If you bet on Brazil to cover the spread, they would need to win the game by two or more goals in order for your bet to be successful. If you bet on Czechoslovakia to cover the spread, they would need to win the game outright or lose by no more than one goal.

The numbers in parentheses indicate the odds for each team to cover the spread. In this case, if you bet $100 on Brazil to cover the spread at odds of +110, you would win $110 if Brazil wins by two or more goals. If you bet $100 on Czechoslovakia to cover the spread at odds of -130, you would win $76.92 if Czechoslovakia wins or loses by no more than one goal. Now with the probability table it is easy to see for the simple layman who has no knowledge of statistics to place a bet on a team, knowing the risks and probabilities he is taking exactly, avoiding predatory behaviors of large corporate betting practices.

Conclusion

In conclusion, our data analysis of FIFA World Cup matches using various statistical methods such as the KS-test, Elo ranking system, MLE, logistic function, and log-likelihood validation has shown promising results in predicting the outcomes of future matches more accurately than just relying on normal data.

This model can be used by coaches, analysts, and fans alike to make more informed decisions when it comes to predicting the results of future FIFA World Cup matches. Our findings demonstrate the value of incorporating advanced statistical techniques into sports analytics.

Appendix

1.1: Plotting Poisson and observed total goals in a game. Doing KS-test to confirm.

```
goals = wcmatches %>%
    mutate(total_goals = home_score + away_score) %>%
    select(total_goals)

ggplot(data = goals, aes(x = total_goals)) + geom_bar() +
    labs(x = "total goals", title = "Observed Amount of Total Goals in a Game") +
    geom_vline(xintercept = mean(goals$total_goals),
        colour = "red", linetype = "dotted", lwd = 0.75) +
    scale_x_discrete(limits = c(0:max(goals$total_goals)))


totals_lambda = mean(goals$total_goals)


totals_points = as.data.frame(rpois(nrow(goals),
    lambda = totals_lambda))
names(totals_points) <- c("random_variables")

ggplot(data = totals_points, aes(x = random_variables)) +
    geom_bar() + labs(x = "random poisson variable",
    title = "Randomly Simulated Poisson Distribution (Lambda = 2.831)") +
    geom_vline(xintercept = totals_lambda, colour = "red",
        linetype = "dotted", lwd = 0.75) + scale_x_discrete(limits = c(0:max(totals_points$random_varial


ks.test(totals_points$random_variables, goals$total_goals)
```

2.1: Cleaning the results dataframe for the estimation later on.

```
results = wcmatches %>%
    select(home_team, home_score, away_team, away_score)

results$home_score = as.integer(results$home_score)
results$away_score = as.integer(results$away_score)

results$winner = ifelse(results$home_score > results$away_score,
    "H", ifelse(results$home_score < results$away_score,
        "A", "D"))

# a draw is considered a loss, we do this to
# have binary outcomes for the logistical
# regression part.
results$home_win = ifelse(results$winner == "H",
    1, 0)
results$away_win = ifelse(results$winner == "A",
    1, 0)
```

2.2: Setting the proper names in the results dataframe.

```r
# Merging West and East Germany
results$home_team[results$home_team == "West Germany"] = "Germany"
results$home_team[results$home_team == "East Germany"] = "Germany"
results$away_team[results$away_team == "West Germany"] = "Germany"
results$away_team[results$away_team == "East Germany"] = "Germany"

# Changing Soviet Union to Russia
results$home_team[results$home_team == "Soviet Union"] = "Russia"
results$away_team[results$away_team == "Soviet Union"] = "Russia"

# Changing Republic of Ireland to Ireland
results$home_team[results$home_team == "Republic of Ireland"] = "Ireland"
results$away_team[results$away_team == "Republic of Ireland"] = "Ireland"

results$home_team[results$home_team == "Northern Ireland"] = "Ireland"
results$away_team[results$away_team == "Northern Ireland"] = "Ireland"

# Changing Yugoslavia to Serbia and
# Montenegro
results$home_team[results$home_team == "FR Yugoslavia"] = "Yugoslavia"
results$away_team[results$away_team == "FR Yugoslavia"] = "Yugoslavia"
results$home_team[results$home_team == "Yugoslavia"] = "Serbia"
results$away_team[results$away_team == "Yugoslavia"] = "Serbia"
results$home_team[results$home_team == "Serbia"] = "Serbia and Montenegro"
results$away_team[results$away_team == "Serbia"] = "Serbia and Montenegro"
```

2.3: Get the unique teams from the results dataset for the Elo dataset.

```r
# stack the two team columns on top of each
# other so we have one long list of the
# teams from the results data frame
teams_list = data.frame(teams = c(results$home_team,
    results$away_team))

# sort the unique teams in alphabetical
# order
teams = data.frame(sort((team_name = unique(teams_list$teams))))
colnames(teams)[1] = "country"

teams$ELO = rep(1500, nrow(teams))
```

2.4: Logistic function graph

```r
# Define logistic function
logistic <- function(teamARating, teamBRating,
    C) {
    1/(1 + exp(-C * (teamARating - teamBRating)))
}

teamA_elo = seq(from = 500, to = 2500, by = 1)
teamB_elo = 1500
probs1 = c()
probs2 = c()
```

```
probs3 = c()
probs4 = c()
probs5 = c()
probs6 = c()
probs7 = c()
for (i in 1:length(teamA_elo)) {
    probs1 = append(probs1, logistic(teamA_elo[i],
        teamB_elo, 0.02))
    probs2 = append(probs2, logistic(teamA_elo[i],
        teamB_elo, 0.016))
    probs3 = append(probs3, logistic(teamA_elo[i],
        teamB_elo, 0.012))
    probs4 = append(probs4, logistic(teamA_elo[i],
        teamB_elo, 0.008))
    probs5 = append(probs5, logistic(teamA_elo[i],
        teamB_elo, 0.005))
    probs6 = append(probs6, logistic(teamA_elo[i],
        teamB_elo, 0.003))
    probs7 = append(probs7, logistic(teamA_elo[i],
        teamB_elo, 0.0025))
}
probs_elo = data.frame(x = teamA_elo, c1 = probs1,
    c2 = probs2, c3 = probs3, c4 = probs4, c5 = probs5,
    c6 = probs6, c7 = probs7)

# Convert data from wide to long format
probs_elo_long = reshape2::melt(probs_elo, id.vars = "x",
    variable.name = "y_var", value.name = "y")

# Plot multiple lines with different colors
ggplot(probs_elo_long, aes(x = x, y = y, color = y_var)) +
    geom_line() + labs(x = "Difference in Elo (r(A) - r(B))",
    y = "Probability of Team A Winning", color = "Constant",
    title = "Probability of Team A Beating B Given Different Elo's") +
    geom_vline(xintercept = 1500, linetype = "dotted") +
    scale_color_discrete(labels = c("C = 0.020",
        "C = 0.016", "C = 0.012", "C = 0.008",
        "C = 0.005", "C = 0.003", "C = 0.0025")) +
    scale_x_continuous(limits = c(500, 2500),
        labels = c("-1000", "-500", "0", "500",
            "1000"))
```

2.5: Cleaning the world elo rankings (current elo ranking system for MLE estimation)

```
# Read in data
data <- read.csv("~/Desktop/DATA 2010/project/World Football Elo Rankings.csv",
    header = FALSE)

# Create new dataframe to store cleaned data
cleaned_data <- data.frame(value = numeric(length = nrow(data)),
    country = character(length = nrow(data)),
    number = numeric(length = nrow(data)))
```

```
# Loop through rows of original data,
# combining rows for countries that span two
# rows
for (i in 1:nrow(data)) {
    if (i%%3 == 1) {
        if (i < nrow(data)) {
            if (nchar(data[i + 1, 1]) > 2) {
                cleaned_data[i, 1] <- data[i,
                  1]
                cleaned_data[i, 2] <- paste(data[i,
                  1], data[i + 1, 1], sep = " ")
                cleaned_data[i, 3] <- as.numeric(data[i +
                  2, 1])
            } else {
                cleaned_data[i, 1] <- data[i,
                  1]
                cleaned_data[i, 2] <- data[i +
                  1, 1]
                cleaned_data[i, 3] <- as.numeric(data[i +
                  2, 1])
            }
        } else {
            cleaned_data[i, 1] <- data[i, 1]
        }
    }
}

# Remove any rows that have NA in the
# country column
cleaned_data = cleaned_data[!is.na(cleaned_data$country),
    ]

# Rename columns
colnames(cleaned_data) = c("rank", "country_rank",
    "elo")

# filter the data so it just includes the
# rows we want and removes the number in the
# country
Elo_Rankings = cleaned_data %>%
    filter(rank != 0) %>%
    separate(country_rank, into = c("rankDelete",
        "country"), sep = " ", remove = FALSE) %>%
    select(rank, country, elo)
```

2.6: Clean furether so now we have data that we can use for MLE.

```
# between the teams and elo ranking (current
# from website) datasets, there is a
# relationship between the country name
# select just the information we need
teams_from_results = teams %>%
    select(country)
```

```
current_teams_elo = Elo_Rankings %>%
    select(country, elo)
# now have the ones that match between the
# current world Elo and the ones in our data
# by a inner join
joined = inner_join(current_teams_elo, teams_from_results,
    by = "country")

# now filter so it has the game results of
# the countries we have in the joined data
filtered_results = results[results$home_team %in%
    joined$country & results$away_team %in% joined$country,
    ]

# now add both teams elo's to the filtered
# results, it worked out that we get about
# 2/3 of the results to do the estimation
# on, this works out well.

results_elo = filtered_results %>%
    left_join(joined, by = c(home_team = "country")) %>%
    rename(home_elo = elo) %>%
    left_join(joined, by = c(away_team = "country")) %>%
    rename(away_elo = elo)

table_to_print_below = results_elo %>%
    select(-c(home_win, away_win))
```

2.7: MLE calibration for the C constant in the logistic function.

```
# MLE information we need
MLE_data = results_elo %>%
    select(home_elo, away_elo, winner)

# 1 for a Home team win and 0 for an away
# team win
MLE_data$winner = ifelse(MLE_data$winner == "H",
    1, 0)

# Define log-likelihood function
ngll = function(C, HElo, AElo, outcome) {
    muA = 1/(1 + exp(-C * (HElo - AElo)))
    -sum(outcome * log(muA) + (1 - outcome) *
        log(1 - muA))
}

# Find maximum likelihood estimate of
# constant using L-BFGS-B optimization BFGS
# determines the descent direction by
# preconditioning the gradient with
# curvature information.  L-BFGS-B is just
# limited memory BFGS optimization
fit <- optim(par = 0, fn = ngll, HElo = MLE_data$home_elo,
```

```
    AElo = MLE_data$away_elo, outcome = MLE_data$winner,
    method = "L-BFGS-B", lower = 0.001, upper = 0.02)
# we use these bounds from the visual graph
# above

constant = fit$par
constant
```

2.8: Functions that are used for the Elo rankings system with calibrated values.

```
# function is the probability value that is
# used for the score adjustment, the first
# parameter is the team that the probability
# of winning is calculated for, team B is
# the team that A is playing so the
# probability of winning depends on team B's
# elo.
muA = function(teamA_elo, teamB_elo) {
    return(1/(1 + exp(-constant * (teamA_elo -
        teamB_elo))))
}

# function is the score adjustment, the
# function takes in teams elo, the outcome
# (1 for win, 0 for loss), and the mu value
# or probability of winning in that game, it
# returns the new elo based on the result of
# the game
scoreAdjust = function(elo, outcome, mu) {
    return(elo + 60 * (outcome - mu))
}
```

2.9: Dealing with split of Czechoslovakia.

```
last_row_home = tail(which(results$home_team ==
    "Czechoslovakia"), 1)
last_row_away = tail(which(results$away_team ==
    "Czechoslovakia"), 1)

# last occurrence is in row 460, so we will
# look to row 460, inclusive, then update
# Slovakia and Czech Republic's Elo and then
# continue the loop
```

2.10: Elo simulation code

```
# we will need to use the results table to
# update the teams data

#-----------------------------------------------------------------
# loop to last occurrence of Czechoslovakia
# home team col
for (i in 1:last_row_away) {
```

```
    # get the elo of both the home and away
    # team
    home_elo = teams$ELO[teams$country == results$home_team[i]]
    away_elo = teams$ELO[teams$country == results$away_team[i]]
    # home probability of winning
    home_prob = muA(home_elo, away_elo)

    # get the outcome
    if (results$home_win[i] == 1) {
        outcome = 1
    } else {
        outcome = 0
    }

    # score adjustment
    new_elo = scoreAdjust(home_elo, outcome, home_prob)

    # update the new elo
    teams$ELO[teams$country == results$home_team[i]] = new_elo
}
#-------------------------------------------------------------

#-------------------------------------------------------------
# loop to last occurrence of Czechoslovakia
# away team col
for (i in 1:last_row_away) {
    # get the elo of both the home and away
    # team
    home_elo = teams$ELO[teams$country == results$home_team[i]]
    away_elo = teams$ELO[teams$country == results$away_team[i]]
    # home probability of winning
    away_prob = muA(away_elo, home_elo)

    # get the outcome
    if (results$away_win[i] == 1) {
        outcome = 1
    } else {
        outcome = 0
    }

    # score adjustment
    new_elo = scoreAdjust(away_elo, outcome, away_prob)

    # update the new elo
    teams$ELO[teams$country == results$away_team[i]] = new_elo
}
#-------------------------------------------------------------

#-------------------------------------------------------------
# now need to update Slovakia and Czech
# Republic ELo
teams$ELO[teams$country == "Slovakia"] = teams$ELO[teams$country ==
    "Czechoslovakia"]
```

```
teams$ELO[teams$country == "Czech Republic"] = teams$ELO[teams$country ==
    "Czechoslovakia"]
#-----------------------------------------------------------------


#-----------------------------------------------------------------
# loop from last occurrence of
# Czechoslovakia to end home team col
for (i in last_row_away:nrow(results)) {
    # get the elo of both the home and away
    # team
    home_elo = teams$ELO[teams$country == results$home_team[i]]
    away_elo = teams$ELO[teams$country == results$away_team[i]]
    # home probability of winning
    home_prob = muA(home_elo, away_elo)

    # get the outcome
    if (results$home_win[i] == 1) {
        outcome = 1
    } else {
        outcome = 0
    }

    # score adjustment
    new_elo = scoreAdjust(home_elo, outcome, home_prob)

    # update the new elo
    teams$ELO[teams$country == results$home_team[i]] = new_elo
}
#-----------------------------------------------------------------

#-----------------------------------------------------------------
# loop to from last occurrence of
# Czechoslovakia to end away team col
for (i in last_row_away:nrow(results)) {
    # get the elo of both the home and away
    # team
    home_elo = teams$ELO[teams$country == results$home_team[i]]
    away_elo = teams$ELO[teams$country == results$away_team[i]]
    # home probability of winning
    away_prob = muA(away_elo, home_elo)

    # get the outcome
    if (results$away_win[i] == 1) {
        outcome = 1
    } else {
        outcome = 0
    }

    # score adjustment
    new_elo = scoreAdjust(away_elo, outcome, away_prob)

    # update the new elo
```

```
    teams$ELO[teams$country == results$away_team[i]] = new_elo
}
#------------------------------------------------------------------
```

2.11: MLE for the lambda adjustment of K for Brazil.

```
observed_games = results %>%
    filter((home_team == "Brazil") | (away_team ==
        "Brazil"))

# brazils elos
brazil_elo = sorted$ELO[sorted$country == "Brazil"]

# brazils goals
Yi = observed_games %>%
    mutate(goals = ifelse(home_team == "Brazil",
        home_score, away_score)) %>%
    filter(home_team == "Brazil" | away_team ==
        "Brazil") %>%
    select(goals)

# difference in elos for each game
di = observed_games %>%
    left_join(sorted %>%
        rename(home_elo = ELO), by = c(home_team = "country")) %>%
    left_join(sorted %>%
        rename(away_elo = ELO), by = c(away_team = "country")) %>%
    mutate(non_brazil = ifelse(home_elo != brazil_elo,
        home_elo, away_elo)) %>%
    mutate(difference = brazil_elo - non_brazil) %>%
    select(difference)

lambda_Brazil = mean(Yi$goals)


# Define the log-likelihood function
loglik <- function(K, elo_diff, goals, lambda) {
    lambda_prime <- lambda + K * elo_diff
    ll <- sum(goals * log(lambda_prime) - lambda_prime -
        lgamma(goals + 1))
    return(-ll)  # return the negative log-likelihood for optimization
}

fit = optim(par = 0, fn = loglik, elo_diff = di,
    goals = Yi, lambda = lambda_Brazil, method = "Nelder-Mead")
K_mle_Brazil = fit$par
```

2.12: MLE for the lambda adjustment of K for Czechoslovakia.

```
observed_games = results %>%
    filter((home_team == "Czechoslovakia") | (away_team ==
        "Czechoslovakia"))
```

```r
# Czechoslovakia elos
czechoslovakia_elo = sorted$ELO[sorted$country ==
    "Czechoslovakia"]

# Czechoslovakia goals
Yi = observed_games %>%
    mutate(goals = ifelse(home_team == "Czechoslovakia",
        home_score, away_score)) %>%
    filter(home_team == "Czechoslovakia" | away_team ==
        "Czechoslovakia") %>%
    select(goals)

# difference in elos for each game
di = observed_games %>%
    left_join(sorted %>%
        rename(home_elo = ELO), by = c(home_team = "country")) %>%
    left_join(sorted %>%
        rename(away_elo = ELO), by = c(away_team = "country")) %>%
    mutate(non_brazil = ifelse(home_elo != czechoslovakia_elo,
        home_elo, away_elo)) %>%
    mutate(difference = czechoslovakia_elo - non_brazil) %>%
    select(difference)

lambda_czechoslovakia = mean(Yi$goals)


# Define the log-likelihood function
loglik <- function(K, elo_diff, goals, lambda) {
    lambda_prime <- lambda + K * elo_diff
    ll <- sum(goals * log(lambda_prime) - lambda_prime -
        lgamma(goals + 1))
    return(-ll)  # return the negative log-likelihood for optimization
}

fit = optim(par = 0, fn = loglik, elo_diff = di,
    goals = Yi, lambda = lambda_czechoslovakia,
    method = "Nelder-Mead")
K_mle_czechoslovakia = fit$par
```

2.13: Plotting the changes in lambda based on Elo

```r
# plot elo difference on x axis plot lambda
# on y axis, vertical horizontal to show
# original lambda

elo_diff_graph = seq(-500, 500, 1)

lambda_function_brazil = function(diff) {
    return(lambda_Brazil + K_mle_Brazil * (diff))
}

lambda_function_CS = function(diff) {
    return(lambda_czechoslovakia + K_mle_czechoslovakia *
```

```
        (diff))
}

resulting_lambdas_brazil = sapply(elo_diff_graph,
    lambda_function_brazil)
resulting_lambdas_CS = sapply(elo_diff_graph,
    lambda_function_CS)


df = data.frame(elo_diff_graph, resulting_lambdas_brazil,
    resulting_lambdas_CS)

ggplot(df, aes(x = elo_diff_graph)) + geom_line(aes(y = resulting_lambdas_brazil,
    color = "Brazil Lambdas")) + geom_line(aes(y = resulting_lambdas_CS,
    color = "Czechoslovakia Lambdas")) + scale_color_manual(values = c("red",
    "blue"), name = "Lambdas", labels = c("Brazil Lambdas",
    "Czechoslovakia Lambdas")) + labs(x = "Difference in Elos",
    y = "lambda values") + geom_hline(yintercept = lambda_Brazil,
    linetype = "dotted", color = "red") + geom_hline(yintercept = lambda_czechoslovakia,
    linetype = "dotted", color = "blue")
```

3.1: Probability contingency matrix code.

```
wcmatches = read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2022/202
worldcups = read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2022/202

poisson_calc <- function(team1, team2) {
    team1_goals <- wcmatches %>%
        mutate(goals = ifelse(home_team == team1,
            home_score, away_score)) %>%
        filter(home_team == team1 | away_team ==
            team1) %>%
        select(goals)

    team1_lambda <- mean(team1_goals$goals)

    team1_gs <- unique(team1_goals$goals)

    team2_goals <- wcmatches %>%
        mutate(goals = ifelse(home_team == team2,
            home_score, away_score)) %>%
        filter(home_team == team2 | away_team ==
            team2) %>%
        select(goals)

    team2_gs <- unique(team2_goals$goals)

    team2_lambda <- mean(team2_goals$goals)
    default_val <- 0

    prob_matrix <- outer(dpois(sort(team2_gs),
        team2_lambda), dpois(sort(team1_gs), team1_lambda),
        "*")
```

```
    prob_table <- prop.table(prob_matrix)
    prob_matrix[is.na(prob_matrix)] <- default_val
    prob_table

}

wc_freq <- table(wcmatches$home_team, wcmatches$away_team)
freq_wc <- as.data.frame(wc_freq)
sorted_wc <- freq_wc[order(-freq_wc$Freq), ]
top_matchup <- sorted_wc[1, ]

top_matchup_final <- subset(wcmatches, (home_team ==
    top_matchup$Var1 & away_team == top_matchup$Var2) |
    (away_team == top_matchup$Var1 & home_team ==
        top_matchup$Var2))
top_matchup_final


poisson_calc_final <- function(team1, team2, team1_lambda_pr,
    team2_lambda_pr) {
    team1_goals <- wcmatches %>%
        mutate(goals = ifelse(home_team == team1,
            home_score, away_score)) %>%
        filter(home_team == team1 | away_team ==
            team1) %>%
        select(goals)

    team1_gs <- unique(team1_goals$goals)

    team2_goals <- wcmatches %>%
        mutate(goals = ifelse(home_team == team2,
            home_score, away_score)) %>%
        filter(home_team == team2 | away_team ==
            team2) %>%
        select(goals)

    team2_gs <- unique(team2_goals$goals)

    default_val <- 0

    prob_matrix <- outer(dpois(sort(team2_gs),
        team2_lambda_pr), dpois(sort(team1_gs),
        team1_lambda_pr), "*")
    prob_table <- prop.table(prob_matrix)
    prob_matrix[is.na(prob_matrix)] <- default_val
    prob_matrix

}

table_elo <- poisson_calc_final("Brazil", "Czechoslovakia",
    2.195533, 1.170391)
rownames(table_elo) <- 0:6
colnames(table_elo) <- 0:7
```

```
xtable_elo <- xtable(table_elo)
print(xtable_elo, type = "latex", include.rownames = TRUE,
    booktabs = TRUE)
```

3.2: Log liklihood and cross entropy calculations

```
# Observed scores
scores <- c("2-1", "1-1", "0-0", "3-1", "4-1")
away_scores <- c(2, 1, 0, 3, 4)
home_scores <- c(1, 1, 0, 1, 1)
score_mat <- (rbind(away_scores, home_scores))

# Probability table, prob_table is the new
# table with the updated lambda's
prob_table <- poisson_calc_final("Brazil", "Czechoslovakia",
    2.195533, 1.170391)
prob_table2 <- poisson_calc("Brazil", "Czechoslovakia")

# log-likelihood of old lambda
log_likelihood <- sum(log(prob_table2[score_mat[2,
    ] + 1, score_mat[1, ] + 1]))

# cross-entropy of old lambda
n <- length(scores)
cross_entropy <- -1/n * sum(log(prob_table2[score_mat[2,
    ] + 1, score_mat[1, ] + 1]))

cat("Log-likelihood:", log_likelihood, "\n")
cat("Cross-entropy:", cross_entropy, "\n")
```

Sources and References

```
# http://eloratings.net/
# https://machinelearningmastery.com/bayes-theorem-for-machine-learning/
# https://www.sbo.net/strategy/football-prediction-model-poisson-distribution/
# https://dashee87.github.io/data%20science/football/r/predicting-football-results-with-statistical-mod
# https://www.kaggle.com/code/sslp23/predicting-fifa-2022-world-cup-with-ml#Data-Analysis
# University of Manitoba DATA-2010 (Ashani
# Wickramasinghe) (Tools and Techniques for
# Data Science) - Course slide material
```