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.

```
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)
```

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.

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.

```
# 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"
```

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.

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

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:

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

- $\mathbf{r(A)}$ 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.

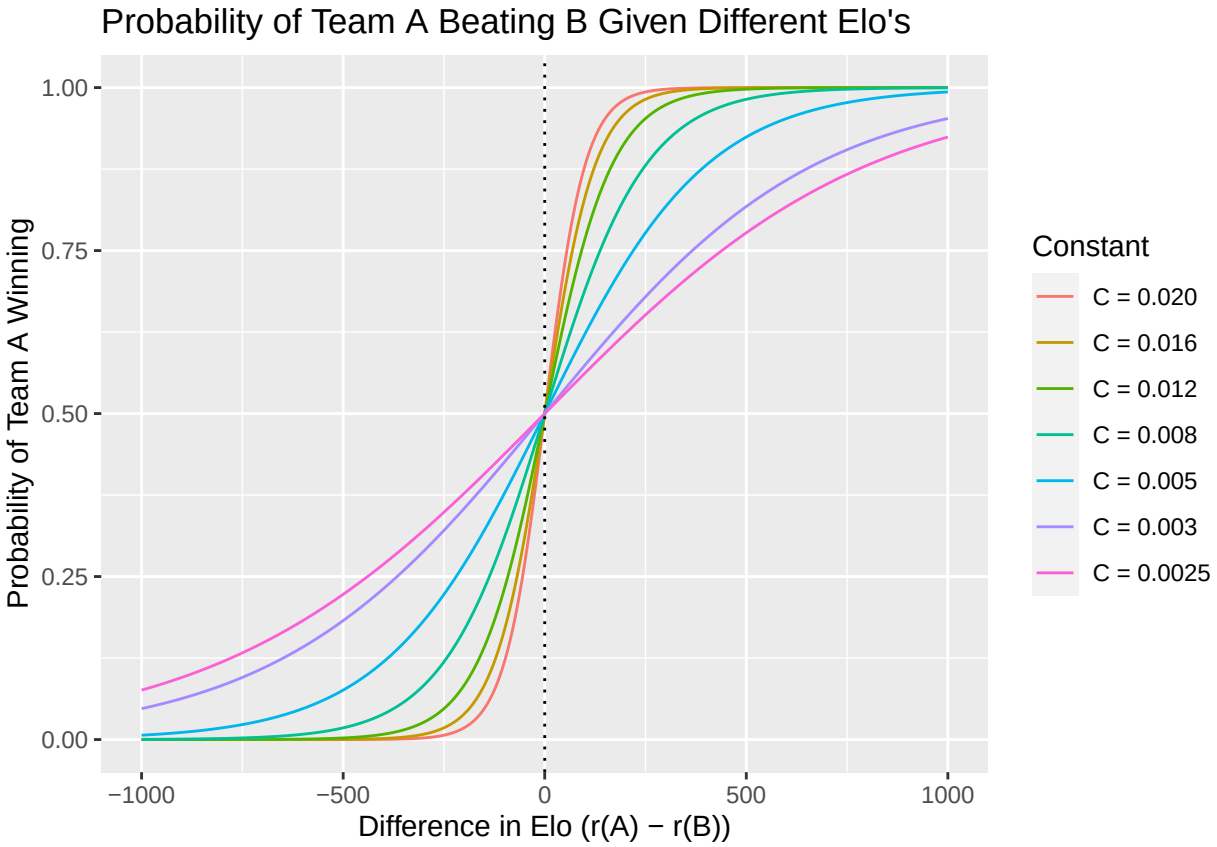
The first step is to find a K value in the formula $r'(A) = r(A) + k(S_A - \mu_A)$. This value determines the score adjustment, in other words how big or small a change in a teams score will be based on the outcome of a game. A larger K value would result in quicker and larger changes in a teams score, 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.



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.

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.

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

```
# 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
```

```
## [1] 0.002803505
```

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

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

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

```
# 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))
}
```

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.

```
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
```

```
# 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
}
#------------------------------------------------------------
```

```
sorted = teams[order(-teams$ELO), ]
rownames(sorted) = 1:nrow(sorted)
```

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, 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.

```
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
K_mle_Brazil
```

```
## [1] 0.0002845764
```

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.

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

# 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
K_mle_czechoslovakia
```

```
## [1] 0.0008911133
```

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

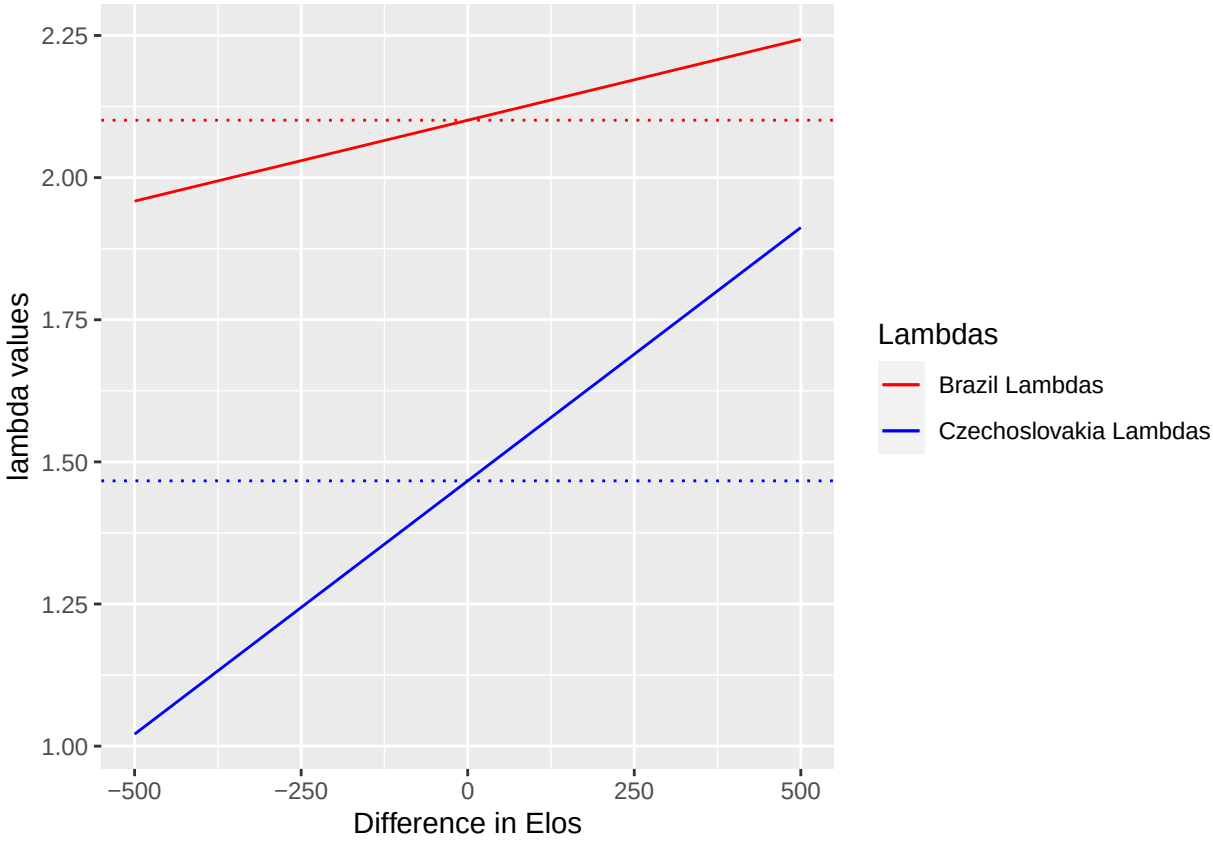
```r
new_brazil = lambda_Brazil + K_mle_Brazil * (brazil_elo -
    czechoslovakia_elo)
new_czechoslovakia = lambda_czechoslovakia + K_mle_czechoslovakia *
    (czechoslovakia_elo - brazil_elo)

# When Brazil plays Czechoslovakia, the new elo
c(lambda_Brazil, new_brazil)
```

```
## [1] 2.100917 2.195533
```

```r
# When Czechoslovakia plays Brazil, the new elo
c(lambda_czechoslovakia, new_czechoslovakia)
```

```
## [1] 1.466667 1.170391
```

```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")
```


2.25
2.00
1.75
1.50
1.25
1.00
lambda values
−500
−250
0
250
500
Difference in Elos
Lambdas
Brazil Lambdas
Czechoslovakia Lambdas