Finding Elo: Restructuring Chess Ratings

Background

In 6th grade, I decided to join the Chess club at my school. The night before our first meeting at 12:30am (a record-breaking time for me to be awake till at the time), I woke up my dad and asked him to teach me how to play chess. We went over the basic rules of how pieces move, and I watched a few YouTube videos on basic openings. With my newfound knowledge, I woke up the next morning and was ready to test my skills against others… Surprisingly, I didn't win a single game.
But from that night onwards, I was hooked. I studied chess openings, obsessed over increasing my chess.com rating, and constantly got click-baited by videos online that would *supposedly* make me a Grandmaster overnight. To my dismay, that wasn't the case—but, I found a new passion. By 9th grade, I was already playing my second inter-school tournament and our team won a bronze medal! Here are my results as depicted in *Figure 1* and a photo from my first tournament in *Image 1*!

Image 1: My First Tournament

But there was a point where I completely stopped playing chess for 2 years. High School, tell me about it. One day, I went on chess.com and played a game for fun—I got checkmated in less than 20 moves. That's analogous to the outcome of me playing a 1v1 against LeBron James. My *true* Elo rating decreased dramatically in those 2 years, but I only lost a few rating points!

Cr.	Masa	stno	Name	Rtg	FED	Club/Grad	T.	Res.
1	14	140	Muhammad Faizan	0	SHP	St Hilda's Pr	5	■ 0
2	74	139	Muhammad S/o Mujib Rahman Aqeel	0	NAP	Ngee Ann Pr	4	□ 1
3	37	137	Michael Fuma	0	SAM	Singapore American Sch	3.5	■ 1
4	22	138	Mohanty Swayam	0	OFS	Overseas Family Sch	5	□ 1
5	15	128	Long Kai Xi Dylan	0	NLP	Northland Pr	6.5	■ 0
6	33	136	Malani Arnav	0	LHP	Lianhua Pro	5	□ 0
7	52	135	Mak Wai Yan Isaiah	0	RLP	Rulang Pr	5	■ 1
8	37	151	Ng Wei An Ryan	0	SWP	Sembawang Pr	5	□ 0
9	52	248	Yeo Jay Yoong Ryan	0	NYP	Nanyang Pr	4.5	■ 1

Figure 1: My First Tournament Results; 5 W/4 L

Thinking about my own experiences made me interested in how a player's time off from playing affects how their rating should increase or decrease when they play again. The current Elo system does not take this into account. Through this paper, I will explore the Chess rating system through a probabilistic lens—incorporating concepts of gaussian distributions, bayesian inference, scaling factors, and code-based simulations. I have created a new rating system that accounts for a player's time since their last game using concepts learnt in CS109.

What is the Elo System?

The Elo system was invented by Árpád Élő, a Hungarian-American physics professor, and was adopted by the US Chess Federation in 1960. It forms the basis of how players are compared to one another (using a numeric scale) in games such as Chess, Basketball, Football, etc. as depicted in Figures 2 and 3:

#	Name	Classic
1	Carlsen, Magnus	2852.6
2	Nepomniachtchi, Ian	2795.0
3	Ding, Liren	2788.0
4	Firouzja, Alireza	2785.0
5	Giri, Anish	2768.0
6	Nakamura, Hikaru	2768.0
7	Caruana, Fabiano	2766.0
8	So, Wesley	2761.0
9	Anand, Viswanathan	2754.0
10	Radjabov, Teimour	2747.0

Figure 2: World Chess Elo Rankings

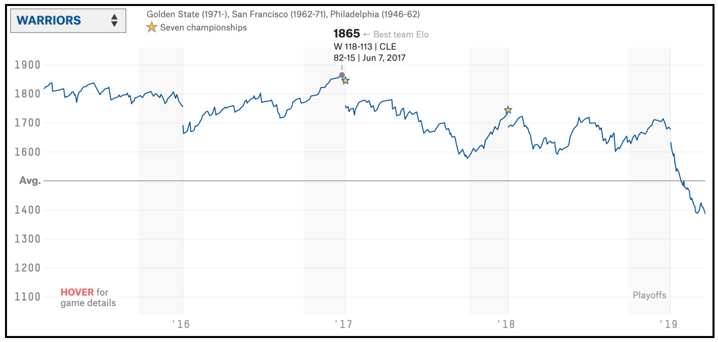


Figure 3: Warriors Elo Graph 2016-19 Playoffs

How the Elo Rating System Works

Probabilities of Each Outcome

R_A = Rating of Player A.

R_B = Rating of Player B.

Let's break this down into 3 cases where both players plays each other 100 times:

1. **If $R_A > R_B$:** We expect Player A to have more points than player B i.e > 50 points.
2. **If $R_A = R_B$:** We expect Player A and Player B to have approximately the same number of points ~50
3. **If $R_A < R_B$:** We expect Player A to have less points than player B i.e < 50 points.

The Elo formula for calculating Player A's expected score against Player B is modelled using a logistic regression. Here is the derivation:

$$\frac{P(A\ wins)}{P(B\ wins)} = 10^{\frac{R_B - R_A}{400}} \text{ (explained later)}$$

$$P(A\ wins) = 10^{\frac{R_B - R_A}{400}} \cdot P(B\ wins)$$

$$P(A\ wins) = 10^{\frac{R_B - R_A}{400}} \cdot (1 - P(A\ wins))$$

$$P(A\ wins) = 10^{\frac{R_B - R_A}{400}} - 10^{\frac{R_B - R_A}{400}} P(A\ wins)$$

$$P(A\ wins) + 10^{\frac{R_B - R_A}{400}} P(A\ wins) = 10^{\frac{R_B - R_A}{400}}$$

$$= P(A\ wins)(1 + 10^{\frac{R_B - R_A}{400}}) = 10^{\frac{R_B - R_A}{400}}$$

$$P(A\ wins) = \frac{10^{\frac{R_B - R_A}{400}}}{1 + 10^{\frac{R_B - R_A}{400}}}$$

Using a special definition of a logistic curve (base10) from the Bernoulli Differential Equation: (a complex proof that is not the focus of this paper):

$$P(A\ wins) = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}$$

$P(B\ wins)$ uses the same principle but has R_A - R_B as we compare Player B's strength relative to Player A:

$$P(B\ wins) = \frac{1}{1 + 10^{\frac{R_A - R_B}{400}}}$$

Now, we can extend *Axiom 3 of Probability*(as discussed in Lecture 3: Probability, Slide 18) that states:

$$"If\ E\ and\ F\ are\ mutually\ exclusive\ (i.e\ E \cap F = \emptyset),\ then\ P(E \cup F) = P(E) + P(F)"$$

Our sample space, S, is the probability of Player A winning, $P(A)$, Player B winning, $P(B)$. If both of these are equal to 0.5 then expected outcome is a draw! $P(A)$ and $P(B)$ cannot occur simultaneously—they are mutually exclusive. Thus, we find that:

$$P(A \cup B) = P(A) + P(B)$$
$$P(S) = P(A) + P(B)$$

By *Axiom 2 of Probability,* $P(S) = 1$ and by *Axiom 1 of Probability* any probability must range from [0, 1]

$$1 = P(A) + P(B)$$
$$\therefore P(B) = 1 - P(A)$$

Expected Score

$P(A\ wins)$ is also known as the **Expected Score, $E(A)$,** of player A. That is, the expected number of points they will score against a player of a certain rating. Recall that point range from 0 for a loss, 0.5 for a draw, and 1 for a win. However, while the outcomes may be fixed numbers, the expected value does not—it need not be either 0, 0.5, or 1—it can take on any value from 0 to 1, inclusive. The way the logistic regression works is: if player A has a rating 400 points higher than player B, they are 10 times more likely to win against player B; the base value of 400 was chosen based on the scale developed uniquely for the Elo system. We can show this concept using the formula we derived above:

$$E(A) = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}} = \frac{1}{1 + 10^{\frac{-400}{400}}} = \frac{1}{1.1} = 0.9 \Rightarrow 90\%\ chance\ of\ winning$$
$$E(B) = \frac{1}{1 + 10^{\frac{R_A - R_B}{400}}} = \frac{1}{1 + 10^{\frac{400}{400}}} = \frac{1}{11} = 0.09 \Rightarrow 9\%\ chance\ of\ winning$$

$$\frac{E(A)}{E(B)} = \frac{0.9}{0.09} = 10$$ $\Rightarrow$ player A is 10 times more likely to win against player B

Updating a Player's Elo Rating

We now know how the Elo system calculates a player's expected score, but how does it update their rating?

Let:

R_A = player A's current rating

R'_A = player A's new rating after a result

E_A = the expected value of player A

S_A = player A's actual score i.e 1 for win, 0 for loss, 0.5 for draw

K = scaling factor (higher s.f corresponds to a lower rating/less experience. 40 is the highest in Elo)

$$R'_A = R_A + K(S_A - E_A)$$

Let's do an example, but I'm tired of arbitrary players A and B. In CS109, one of the things I always look forward to is Jerry's *"The Day's CS109 Briefing"* email. I particularly enjoy the sections about Doris. Here are two snippets that really caught my attention:

<u>Date: Mon 1/30/2023</u>

just as we were leaving, Doris met another dog I'm convinced is The One. This new crush is an English bulldog named 😍 Dirk 😍 who lives with his humans about 500 feet from us. The Mrs. Bennett in me is overjoyed.

<u>Date: Wed 2/1/2023</u>

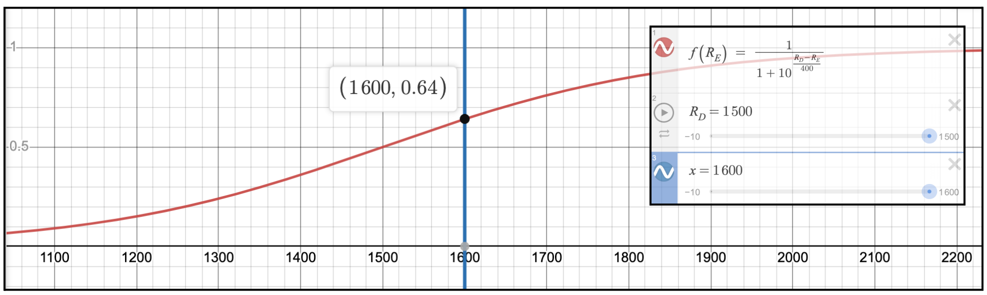
Oh, one thing: Doris dumped Dirk. She's likes Evan now.

Dirk is upset—he was dumped in 3 days?! He challenges Evan to a game of (dog) Chess in hopes to win Doris over…after a grueling match, Dirk wins! Let's see what their new ratings are when given their initial ratings: $R_D = 1600$ and $R_E = 1500$ (let K = 40). We need to first calculate their expected scores:

$$E_D = \frac{1}{1+10^{\frac{R_E - R_D}{400}}} = \frac{1}{1+10^{\frac{1500-1600}{400}}} = \frac{1}{1+10^{-0.25}} = 0.64$$

$$E_E = \frac{1}{1+10^{\frac{R_D - R_E}{400}}} = \frac{1}{1+10^{\frac{1600-1500}{400}}} = \frac{1}{1+10^{0.25}} = 0.36 \ (= 1 - E_D)$$

We can use a graphical approach to visualize the logistic regression and Dirk's expected number of points:



Graph 1: x-axis measures a player's rating and the x = 1600 line represents Dirk's rating. The y-axis is the expected points when playing against Evan whose rating is 1500. This yields 0.64. Play around with it [here]!

<u>Dirk's New Rating</u>

$R'_D = R_D + K(S_D - E_D)$
$= R_D + K(S_D - E_D)$
$= 1600 + 40(1 - 0.64)$
$= 1600 + 40(0.32)$
$= 1600 + 14.4$
$R'_D = 1614.4$

<u>Evan's New Rating</u>

$R'_E = R_E + K(S_E - E_E)$
$= R_D + K(S_D - E_D)$
$= 1600 + 40(0 - 0.36)$
$= 1500 + 40(-0.36)$
$= 1600 - 14.4$
$R'_E = 1585.6$

While Dirk won the Chess game, he didn't win Doris back; I guess she's not interested after their antics:

both Evan and Dirk along with their doggy moms and dads. Initially, E and D are circling around each other in that way that dogs do, but then Dirk sees Doris and Evan sees her about three seconds later. I fully expected Doris to be shaken, but she delivered the best side eye I've ever seen out her (note: she's

Issues with the Elo System and Focus of this Paper

While the Elo system works very well, there are 2 primary flaws I noticed:

Critique	Explanation
Does not account for *how* a player wins, draws, or loses	If Dirk had beat Evan simply because Evan ran out of time despite Evan crushing Dirk in the game, does that actually make Dirk a better player? Why should Evan lose 14.4 points for moving his pieces too slowly? It would be fairer for him to only lose half of those points. On the other hand, if Dirk had won because Evan resigned and accepted defeat, I feel that is a more legitimate method of being a better chess player. Though, this depends on the definition of what makes someone a "better" chess player. These are only a couple of examples. However, I feel that these concerns are not practical for chess rating systems to incorporate. It may even require human judgment, and keeping track of many factors becomes difficult, especially for chess sites that have millions of users.
Does not take into account the time since a player has last played a game	Imagine Evan has been playing lots and lots of chess for the last 6 months. His rating increased from 800 to 1500 (where it is now). Whereas Dirk, on the other hand, hasn't played chess for over 1 year! His rating is 1600 which is where he left off last year. In a scenario where you would want to predict who has the upper hand in this match, most people would pick Evan. Even though Evan has an objectively lower rating (by 100 points), he has more practice and has been playing recently so you would not write him off so easily. The Elo system does not take this into account. It would still predict that Dirk has a 64% chance of winning.

The issue I have chosen to attempt to address is that the Elo system does not take into account the time since a player has last played a game.

My New Approach to Addressing the Elo Time Issue

Expected Score in My System

The Elo system uses logistic regression to model the expected score of a player against their opponent. However, after learning more about the Gaussian (Normal) distribution and its applicability to umpteen different scenarios, I thought more about how I can apply what we have learned in class to the Elo system. The logistic regression is used as a PMF to calculate probability of a player beating another. I wondered, why not use a normal distribution instead to model a player's expected score? To start off, I created a normal that represents the difference in rating between the two players using the sum of two normals, X and Y. Let X represent *your* rating and Y represent *your opponent's* rating, both as normal distributions.

$$Z = Y - X$$

$$Y \sim N(\mu_Y, \sigma_Y^2)$$

$$X \sim N(\mu_X, \sigma_X^2)$$

$$\therefore Z \sim N(\mu_Y - \mu_X, \sigma_Y^2 + \sigma_X^2)$$

The mean is the difference of both players' current ratings. Now, how do we determine the variance? The purpose of this system is to include the time since your last game for both you and your opponent as that is the largest contributor to the uncertainty in your rating. Thus, the variance is just the days since the players' last rated Chess game. This gives me the parameters for Z's distribution. I used numpy to write code for this:

```
def expected_score(player_rating, opponent_rating, opponent_scaling_factor, my_scaling_factor):
    mu = opponent_rating - player_rating
    sigma = np.sqrt((opponent_scaling_factor) ** 2 + (my_scaling_factor) ** 2)
```

Then, I created a Normal Distribution using these parameters. This allows me to calculate my expected value which is a value between 0 and 1. Now, I return the CDF of 0 as that gives me everything to the left of the curve which is analogous to the expected value of beating your opponent. Imagine your ratings are equal, the CDF would return 0.5 since it is centered around zero. If you are higher rated it would be a above 0.5 and if you are lower rated, below 0.5. This method is mentioned in *Lecture 10: Normal Guassian, Slide 28*:

```
def expected_score(player_rating, opponent_rating, opponent_scaling_factor, my_scaling_factor):
    mu = opponent_rating - player_rating
    sigma = np.sqrt((opponent_scaling_factor) ** 2 + (my_scaling_factor) ** 2)
    normal = stats.norm(mu, sigma)
    return normal.cdf(0)
```

Now, I have a function that gets me the expected score of a player using a Normal Distribution as opposed to the Logistic Regression used by the Elo system.

Updating a Player's Rating in My System

The general mechanism is similar to Bayesian Inference as discussed in class. Take player A, for example:

$$R'_A = R_A + \Delta R_A$$

Where R'_A is the posterior rating after a result, R_A is the prior rating before the game, and ΔR_A represents a player's change in rating after the observed outcome of a game. Since I am interested in the time since the last game, I can include a new factor called the scaling factor as mentioned above. From now onwards, the scaling factor wil represent $\frac{1}{days\ since\ last\ game}$ as I found it much easier to work with to get more significant ΔR_A in my trials. K is set to a value depending on a player's skill. Imagine Player A beats Player B:

Player A wins - Positive change

$$\Delta R_A = \frac{E(A)}{scaling\ factor \cdot K}$$

Player B Loses - Negative change

$$\Delta R_B = \frac{-1 \cdot E(B)}{scaling\ factor \cdot K}$$

In my code, I kept an array of player ratings and updated that array after a result. Translating this to code:

```
delta = (expected_score1) / (player1_sf * K_factor)
player_ratings[index_1] += delta

delta = (-1 * expected_score2) / (player2_sf * K_factor)
player_ratings[index_2] += delta
```

Testing and Results

I spent a while considering how I can actually **test** my ideas. It is infeasible to start from the beginning of rated Chess and parse through all the data from dozens of decades and rate every player in history based on my system. However, what I could do is a controlled simulation. For example, given a series of results in a tournament, what is the Elo rating and rankings for the world's top 10 players versus the same players' ratings and rankings using my system? *Figure 4* depicts this data; I found "days since last game" using 2700chess.com. I made random pairings and results for 20 games amongst the top 10 players. The reason I chose to do 20 games because it is a manageable number of games to keep track of and also a reasonable sample to test out.

World Top 10 Chess Players (Classical)			
Ranking	**Name**	**Rating**	**Days since last game**
1	Magnus	2852.6	42
2	Ian	2795	15
3	Ding	2788	49
4	Alireza	2785	181
5	Anish	2768	15
6	Hikaru	2768	342
7	Wesley	2766	42
8	Fabi	2761	15
9	Anand	2754	154
10	Radjabov	2747	251

Figure 4: Top 10 Chess Players in Classical Chess, Elo rating, and days since last game

Here is a link to a spreadsheet containing all my data and calculations explained below.

Here is my code set up for the data illustrated in *Figure 4*:

```
names = np.array(['Magnus', 'Ian', 'Ding', 'Alireza', 'Anish', 'Hikaru', 'Wesley', 'Fabiano', 'Anand', 'Radjabov']
player_ratings = np.array([2852.6, 2795.0, 2788.0, 2785.0, 2768.0, 2768.0, 2766.0, 2761.0, 2754.0, 2747.0])
scaling_factors = np.array([42, 15, 49, 181, 15, 342, 42, 15, 154, 251])
scaling_factors = 1/scaling_factors
```

Here are the 20 pairings and results I created:

All Results			
Game	**Result**	**Game**	**Result**
1	Magnus beats Ian	11	Hikaru draws Fabiano
2	Ian beats Radjabov	12	Wesley draws Ding
3	Ding draws Alireza	13	Anand beats Radjabov
4	Anish beats Hikaru	14	Magnus draws Ding
5	Wesley beats Anand	15	Ian beats Alireza
6	Anish draws Fabiano	16	Anish draws Alireza
7	Radjabov draws Hikaru	17	Ian beats Fabiano
8	Ding beats Anand	18	Magnus beats Anish
9	Magnus beats Wesley	19	Wesley draws Hikaru
10	Alireza beats Radjabov	20	Fabiano beats Alireza

Player Scores by Game
Magnus (2852.6): 1, 1, 0.5, 1
Ian (2795): 0, 1, 1, 1
Ding (2788): 0.5 , 1, 0.5, 0.5
Alireza (2785): 0.5, 1, 0, 0.5, 0
Anish (2768): 1, 0.5, 0.5, 0
Hikaru (2768): 0, 0.5, 0.5, 0.5
Fabiano (2766): 0.5, 0.5, 0, 1
Wesley (2761): 1, 0, 0.5, 0.5
Anand (2754): 0, 0, 0.5, 1
Radjabov (2747): 0, 0.5, 0, 0

20 Pairings and Results (1 - 0 indicates player's name that is in the relevant column won)										
	Magnus	Ian	Ding	Alireza	Anish	Hikaru	Wesley	Fabiano	Anand	Radjabov
Magnus										
Ian	1 - 0									
Ding	0.5 - 0.5									
Alireza		1 - 0	0.5 - 0.5							
Anish	1 - 0			0.5 - 0.5						
Hikaru					1 - 0					
Wesley	1 - 0		0.5 - 0.5			0.5 - 0.5				
Fabiano		1 - 0		0 - 1	0.5 - 0.5	0.5 - 0.5				
Anand			1 - 0				1 - 0			
Radjabov		1 - 0		1 - 0		0.5 - 0.5			1 - 0	

Elo Rating System Results

I computed the new Elo ratings for each player using spreadsheet formulas and the International Chess Federation (FIDE) official [Elo rating calculator](). Here are the full set of results:

Game 1: Magnus beats Ian		
Name	Old	New
Magnus	2852.6	2856.7
Ian	2795	2790.9

Game 2: Ian beats Radjabov		
Name	Old	New
Ian	2790.9	2795.3
Radjabov	2747	2742.6

Game 3: Ding draws Alireza		
Name	Old	New
Ding	2788	2788
Alireza	2785	2785

Game 4: Anish beats Hikaru		
Name	Old	New
Anish	2768	2773
Hikaru	2768	2763

Game 5: Wesley beats Anand		
Name	Old	New
Wesley	2761	2765.9
Anand	2754	2749.1

Game 6: Anish draws Fabiano		
Name	Old	New
Anish	2773	2772.7
Fabiano	2754	2754.3

Game 7: Radjabov draws Hikaru		
Name	Old	New
Radjabov	2747	2747.2
Hikaru	2763	2762.8

Game 8: Ding beats Anand		
Name	Old	New
Ding	2788	2792.5
Anand	2749.1	2744.6

Game 9: Magnus beats Wesley		
Name	Old	New
Magnus	2856.7	2860.5
Wesley	2765.9	2762.1

Game 10: Alireza beats Radjabov		
Name	Old	New
Alireza	2785	2789.5
Radjabov	2747.2	2742.7

Game 11: Hikaru draws Fabiano		
Name	Old	New
Hikaru	2762.8	2762.7
Fabiano	2754.3	2754.4

Game 12: Wesley draws Ding		
Name	Old	New
Wesley	2762.1	2762.5
Ding	2792.5	2792.1

Game 13: Anand beats Radjabov		
Name	Old	New
Anand	2744.6	2749.6
Radjabov	2742.7	2737.7

Game 14: Magnus draws Ding		
Name	Old	New
Magnus	2860.5	2859.5
Ding	2792.1	2793.1

Game 15: Ian beats Alireza		
Name	Old	New
Ian	2795.3	2800.2
Alireza	2789.5	2784.6

Game 16: Anish draws Alireza		
Name	Old	New
Anish	2772.7	2772.9
Alireza	2784.6	2784.4

Game 17: Ian beats Fabiano		
Name	Old	New
Ian	2800.2	2804.6
Fabiano	2754.4	2750

Game 18: Magnus beats Anish		
Name	Old	New
Magnus	2859.5	2863.3
Anish	2772.9	2769.1

Game 19: Hikaru draws Wesley		
Name	Old	New
Wesley	2762.5	2758.7
Hikaru	2762.7	2758.9

Game 20: Fabiano beats Alireza		
Name	Old	New
Fabiano	2750	2755.5
Alireza	2784.4	2778.9

Notice that in Game 1, Ian loses—his initial rating was 2795 and he loses 4.1 points to bring his new rating to 2790.9. In game 2, Ian's old rating is now 2790.9. This updating happens for every player after each game until the end of the tournament. Here are the final results for the Elo system:

Elo System					
Name	Old Rank	New Rank	Old Rating	New Rating	Change
Magnus	1	1	2852.6	2863.3	10.7
Ian	2	2	2795	2804.6	9.6
Ding	3	3	2788	2793.1	5.1
Alireza	4	4	2785	2778.9	-6.1
Anish	5	5	2768	2769.1	1.1
Hikaru	6	6	2768	2758.9	-9.1
Wesley	7	7	2766	2758.7	-7.3
Fabiano	8	8	2761	2755.5	-5.5
Anand	9	9	2754	2749.6	-4.4
Radjabov	10	10	2747	2737.7	-9.3

Figure 5: Elo Rating System Tournament Results

Based on the Elo System, the ratings of the world's top 10 players did change a decent amount as seen by the green and red arrows. However, their rankings stayed the same as indicated by the grey arrows.

My Rating System Results

For a more in-depth breakdown of my code and how it works, refer to my [YouTube video](). In my tournament simulation code, I had each player play one another based on the games and results I predetermined. After dozens of iterations and tweaking, I realised two things: 1) I need additional scaling factors to ensure rating increases/decreases are more easily noticeable and interpreted by humans 2) I need some extra cases to break down how much a player's rating should actually change depending on the time since their last game (scaling factor, S.F). Here is how I did this:

<u>Additional Scaling Factors</u>

```
# additional scaling factors
days_sf = 1.5
more = 0.4
medium = 0.6
less = 0.8
```

<u>Y = more/less/medium</u>

<u>Win:</u>	**<u>Loss:</u>**
$\Delta R_A = \frac{E(A)}{S.F \cdot Y \cdot K}$	$\Delta R_B = \frac{-E(B)}{S.F \cdot Y \cdot K}$

Based on the special case (below), ther is either a more, medium, or less change in a player's rating. I decided on these values upon dozens of iterations; these values showed an interpretable change in rating and were reasonable as players did not lose/gain disproportionately.

My new formula (on the left) includes the additional scaling factors. The days_sf is used to compare two players' days since last game in the special cases. For example, player 1 not playing for 30 days is pretty much the same time off as player 2 who hasn't played in 40 days so we treat them as the same. But if someone has not played in 200 days versus their opponent who hasn't played in 15 days, then we enter the special cases mentioned as 15 * 1.5 < 200.

<u>Special Cases</u>

Note 1: "days" = days since last game; analogous to S.F used in code
Note 2: These special cases are explained in more depth in my video

<u>Draw:</u>

1. **Same rating and same days**
    a. No change in rating for either
2. **Same rating, different days**
    a. Higher days gains <u>less</u>, lower days loses <u>less</u>
3. **Different rating**
    a. Lower rated gains <u>medium</u>, higher rated loses <u>less</u>

<u>Win:</u>

1. **Both players playing after a while (90 days)**
    a. <u>Winner</u> gains less, loser loses <u>less</u>
2. **Either of the two players playing after > 50 days**
    a. winner rating >= losers rating ; winner days > days * loser days
        i. Winner gains <u>more</u>, loses loses <u>medium</u>
    b. winner rating >= losers rating ; loser days > days * winner days
        i. Winner gains <u>less</u>, loser loses <u>medium</u>
    c. winner rating <= losers rating ; winner days > days * loser days
        i. Winner gains <u>more</u>, loser loses <u>more</u>
    d. winner rating <= losers rating ; loser days > days * winner days
        i. Winner gains <u>medium</u>, loser loses <u>more</u>
3. **None of the above**
    a. Winner gains <u>medium</u>, loser loses <u>medium</u>

Here are 2 examples of the cases above (I've explained more of them in my video!) These are direct outputs from my code for 1 win case and 1 draw case:

<u>Win Case #2d</u>	**<u>Draw Case #3 - None</u>**

```
GAME: 15
WIN

Ian beats Alireza

Ian expected score: 0.462
Alireza expected score: 0.538

Days since Ian has played: 15.0
Days since Alireza has played: 181.0

Ian old rating: 2795.833
Alireza old rating: 2813.056

Ian new rating: 2796.989
Alireza new rating: 2788.721

Ian change in rating: 1.156
Alireza change in rating: -24.334
```

Alireza has not played in quite a while, but his prior rating is very much higher than Ian's (~20 pts). Alireza's expected chance of winning is 0.54 (54%) taking into account he has not played in about 6 months. After losing to Ian, however, Alireza loses about 24 pts which is a lot! That is because 1) he lost to a lower-rated player, and 2) he hasn't played in a long time so his rating is more volatile than Ian's. Ian only gained 1.2 pts despite beating a higher-rated player because the player he beat had not played in a long time.

```
GAME: 11
DRAW

Hikaru draws Fabiano

Hikaru expected score: 0.448
Fabiano expected score: 0.552

Days since Hikaru has played: 342.0
Days since Fabiano has played: 15.0

Hikaru old rating: 2717.712
Fabiano old rating: 2762.615

Hikaru new rating: 2743.238
Fabiano new rating: 2761.579

Hikaru change in rating: 25.526
Fabiano change in rating: -1.035
```

Hikaru has a much lower initial rating than Fabiano—a difference of almost 45 points! On top of that, Hikaru has not played for almost an entire year. Thus, it is impressive that he is even able to compete with players like Fabiano who have been playing very regularly. Therefore, when they draw, Hikaru gains quite a few points. Fabiano only loses a very few points since it is a draw and he has been playing regularly so his rating is likely very close to his current rating and he has been playing quite regularly.

Here are the full table of results after all 20 games are played:

My System					
Name	**Old Rank**	**New Rank**	**Old Rating**	**New Rating**	**Change**
Magnus	1	▶ 1	2852.6	▲ 2871.8	19.2
Ian	2	▶ 2	2795	▲ 2799.4	4.4
Ding	3	▶ 3	2788	▲ 2792.2	4.2
Alireza	4	▼ 8	2785	▼ 2752.3	-32.7
Anish	5	▶ 5	2768	▲ 2769.4	1.4
Hikaru	6	▲ 4	2768	▲ 2769.9	1.9
Wesley	7	▲ 6	2766	▲ 2768.2	1.2
Fabiano	8	▲ 7	2761	▲ 2762.6	1.6
Anand	9	▶ 9	2754	▼ 2741.3	-12.7
Radjabov	10	▶ 10	2747	▼ 2720.2	-26.8

Figure 5: My Rating System Tournament Results

The key difference between my system's results after the tournament and the Elo system's is that my system produced a change in the original rankings. Alireza is much lower ranked now than he was earlier and it is likely that Anand and Radjabov are no longer in the world's top 10 players after their rating drop! All three of these players had been out of the game for quite a while and did not perform well in the tournament hence their drop in rating and the changes in rankings. Hikaru, on the other hand, performed decently with 3 draws and 1 loss after coming back to the game in nearly a year.

Conclusion and Next Steps

Including a player's time off from the game is crucial in determining their true skill level. Players who haven't played in a year are definitely not clear favorites against those who have been playing more regularly (even if their ratings are different). There are players in the world that are inactive and thus when coming back to playing a tournament, their current rating is not representative of their true rating—as a result, their ratings should fluctuate more than those playing regularly. This is the exact problem I noticed applied to my own life, and hence, the reason behind choosing this project. Including this factor does not just apply to Chess, but other sports as well that use Elo such as Basketball, Football, Baseball, etc.

I feel that my system is feasible to implement but keeping track of a player's last game is an additional data point that should be taken into consideration. This may be difficult for large chess sites such as chess.com and lichess.org to implement as they have millions of users. However, I feel this is should be included at least in elite-level chess where players' rankings and ratings matter very much. In fact, it could be the difference between them winning and losing a tournament. Moreover, it could promote players to be more active and participate in more tournaments rather than maintaining their rating and ranking by choosing not to play select tournaments. Implementing an entire new system for a well-established game such as Chess can be challenging, but I feel it is an interesting avenue to explore.

There is definitely room for improvement in my system. More iterations and trial and error can be completed in order to choose the most appropriate additional scaling factors (days_sf, more, medium, less). Additionally, my code can be further optimised—something that was not my focus given the scope of the project. I can also explore different variations of the formula I used to calculate the expected score e.g adjusting how the variance is calculated.

Overall, I really enjoyed this project especially since it is centered around a game that I love: Chess. In this project, I used concepts learned in CS109 such as a normal distribution, expected scores, logistic regressions, CDF of a gaussian, axioms of probability, and coding simulations in Python using numpy and scipy. I learned about how fundamental probability concepts can be applied to complex systems and used in the real world!

Bibliography

1. https://2700chess.com/
2. https://ratings.fide.com/calc.phtml?page=change
3. https://docs.scipy.org/doc/scipy/index.html
4. https://www.youtube.com/watch?v=AsYfbmp0To0
5. https://www.chess.com/insights/aryansahai#overview
6. http://angrystatistician.blogspot.com/2015/10/elos-rating-system-as-forgetful.html
7. https://en.m.wikipedia.org/wiki/Elo_rating_system#Selective_pairing
8. https://projects.fivethirtyeight.com/complete-history-of-the-nba/#warriors
9. https://fivethirtyeight.com/features/how-we-calculate-nba-elo-ratings/
10. https://www.youtube.com/watch?v=GTaAWtuLHuo
11. https://ratings.fide.com/calc.phtml?page=change
12. https://www.w3schools.com/python/ref_func_round.asp
13. https://www.freecodecamp.org/news/2f-in-python-what-does-it-mean/#:~:text=The%20%25f%20formatter%20is%20specifically,point%20number%20is%20rounded%20up.
14. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html
15. https://en.wikipedia.org/wiki/Logistic_function#Logistic_differential_equation
16. https://mattmazzola.medium.com/understanding-the-elo-rating-system-264572c7a2b4
17. https://www.flaticon.com/free-icon/chess-clock_79851
18. https://www.chess.com/forum/view/community/update-to-badges-should-include-ratings-tiers-icons-for-non-titled
19. https://www.freepnglogos.com/images/chess-39290.html