Ironhack data analysis bootcamp
Final project : chess explorator

`

Baudouin Dupont
GITHUB
Project Management and ressources

Prolegomena

Being an avid (rather than talented or disciplined) chess player, and chess being such a staple of machine learning research, the choice of a subject for this project was not much of brain pick in my case.

The question rather was, what will I try to show, and how can this help me become a better player? I already have a few ideas how I could further my mastery of the game, my goal is to verify whether my guts are right through rigorous analysis and clear data presentation.

I will therefore run the same analysis on three different chess game datasets, which will all have been built in the same way using Chess.com's public API.
The first dataset will contain games by the top 100 players in the world, the second will be a selection of games of people I have played against (my best idea to come-up with data somewhat representative of the general population) and the third one will of course be the archives of the games I have played myself along the years.

Before we get too far into the technical details, I will say that despite the sheer complexity you can be faced with when dealing with chess from an analytical point of view (the number get oh so big, oh so fast), the resources online are plentiful and the tools at your disposal efficient and easy to access. This clearly made the task a little less daunting, and I was able to take it step by step.

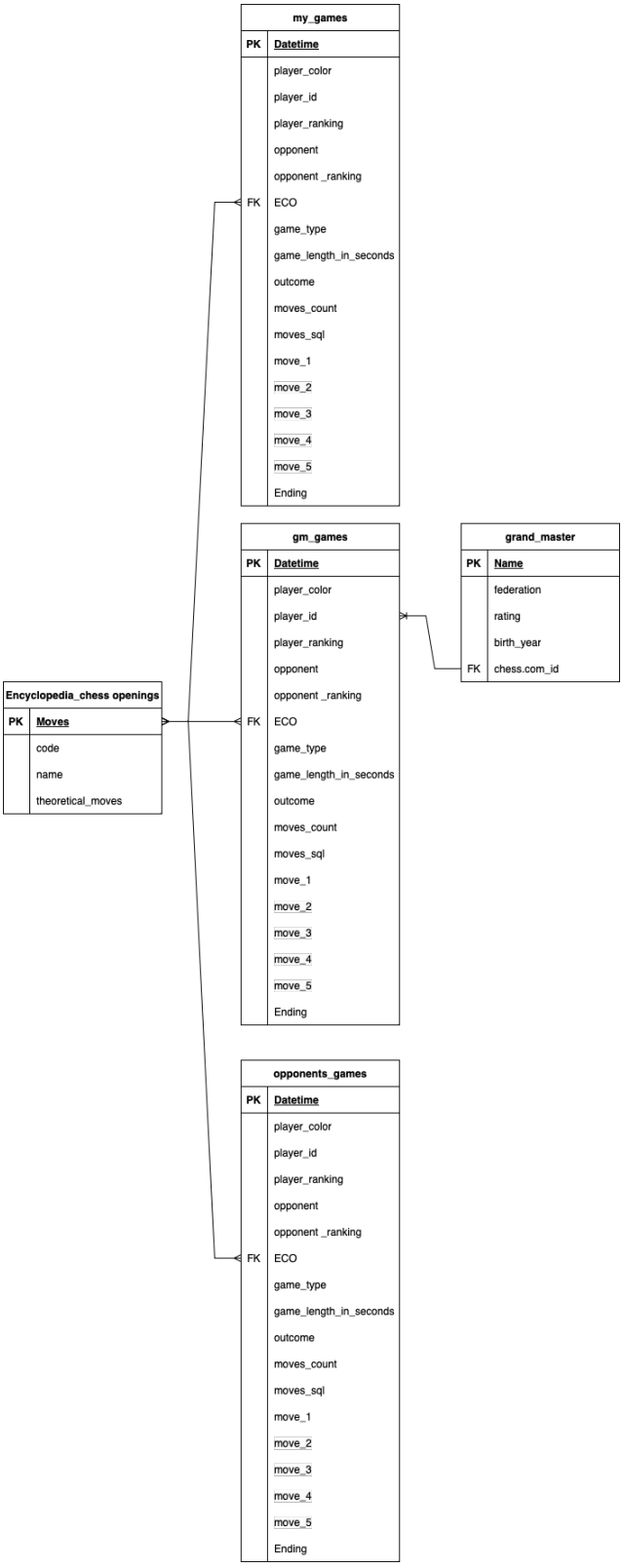
The general idea is twofold: first present statistics of games played by grand masters and then compare them with the same statistics for a (hopefully representative) sample of the population, and then with my own statistics. Then I would like to build a tool that explores the theory a little more by diving into the openings of the games. If I should have enough time, I would then try to adapt a machine learning model to predict the next move for any given position.

The reader can dive a little deeper in this project by reading the project planning wich summarises, all scripts, tables and resources I used or wrote along the way.

Table of Contents

Dataset presentation

This ERD represents the data set I'll be building and using for my analysis. For a more readable version I encourage the reader to visit the project planning page.



Data collection

Data source

The main source of data I am going to use is today's most famous chess website: Chess.com. This website is the main chess hub online, it provides many services among which:

- Chess game server
- Chess news website
- Chess streaming platform
- Chess social network
- Chess database

I have been a long-time user of this platform to play online chess, my 'reservoir' of real life person to play with growing ever thinner as I delved further into the game. Chess.com's rest API is public and has a few different endpoints anyone can tap into:

- Leaderbords
- Individual Players

As will become a trend, I was fortunate enough in the choice of my subject to benefit from the work of many a people before me, and the case of data scrapping was no different. There exist a python wrapper for Chess.com's API that you can just install and import in almost any python script or notebook: chessdotcom.
This wrapper offers different methods to get data from the API. The ones I relied the most heavily upon are the following:

- get_player_game_archives return a list of json files. Each of them holds a month of archives.
- get_titled_players return a list of the players that attained a specific rank.
- get_player_stats returns a list of specific stats
- get_player_profile returns info about player.

Data goal

Armed with this toolkit, I proceeded to create a data scrapping procedure to build the three game database I needed :

- my_games
- gm_games (gm : Grand_Master)
- standard_games

File format

All games are stored in Chess.com's API as PGN files living inside JSON files. Each JSON file corresponds to a month of games for any given player.
PGN is the standard file format for chess games. It consist of to main parts: Headers and Moves. Headers contain a lot of info about the game and the respective players while Moves retrace the entire game in chess standard notation.
In my case, I will need to extract info from both these parts.

Scrapping procedure

The data scrapping procedure exists mainly as a list of functions I created to format the data in a way I could then use. You will find all these functions in the "data_scrapping" notebook on the github repo:

- get_color :
    - this function, given a player_id and a PGN file return the color of the player. It is made necessary by the fact that the PGN files are not player centric, that is they only present the two players by color, ID and ranking.
- get_games : this is by far the most important feature of my data scrapping procedure. Given a player id and a number of months it will return the games of said player for given number of months as a dataframe.

Table construction

I had three tables to build, two were obvious:

- gm_games: I iterated get_games on a list of the 100 best players in the world
- my_games : I applied my_games to my own Chess.com ID

The last one was a little bit more tricky as it is difficult to get Chess.com IDs of representative people. I therefore decided to extract a list of all the players I had ever played against and iterate get_games on this list.

In the end, the three tables all have the same features (more on those in the data cleaning part) but had obviously vastly different number of rows:

- my_games has around 5000 rows
- gm_games has around 175k rows
- opponent_games has around 500k rows

In all these cases, every row is a single game. My dataset therefore holds arounds 680k games. This number could be increased easily but with a lot of patience, as the requests are kind of long to run (3H for the opponent_games). The other problem is that the PGN files are very heavy before data cleaning (opponent_games was a little over a GB just after the initial request and only 200MB after data cleaning and engineering)

Just so you have an idea, here is what a game looks like just after the initial request:

0	2019.04.02	17:26:48	600	White	audouindupor	1036	Pelletammy	1026	C00	ıpont won by	1. e4 { [%clk 0:09:59.9] } 1... e6 { [%clk 0:09:59.5] } 2. Nc3 { [%clk 0:09:52.4] } 2... a6 { [%clk 0:09:57.9] } 3. Bc4 { [%clk 0:09:49.6] } 3... Qf6 { [%clk 0:09:53.9] } 4. d3 { [%clk 0:09:43.4] } 4... Bc5 { [%clk 0:09:51.8] } 5. f4 { [%clk 0:09:38.3] } 5... Qd4 { [%clk 0:09:47.2] } 6. Nh3 { [%clk 0:09:30.4] } 6... Bb4 { [%clk 0:09:36.5] } 7. Bd2 { [%clk 0:09:01.2] } 7... Bxc3 { [%clk 0:09:21.7] } 8. Bxc3 { [%clk 0:08:54.1] } 8... Qe3+ { [%clk 0:09:20.5] } 9. Qe2 { [%clk 0:08:46.9] } 9... Qxe2+ { [%clk 0:09:13.1] } 10. Kxe2 { [%clk 0:08:45.5] } 10... f6 { [%clk 0:09:12.5] } 11. e5 { [%clk 0:08:34.6] } 11... Kf7 { [%clk 0:09:11.3] } 12. exf6 { [%clk 0:08:27.4] } 12... Nxf6 { [%clk 0:09:10] } 13. Raf1 { [%clk 0:08:20.7] } 13... Nc6 { [%clk 0:09:08.6] } 14. Kd1 { [%clk 0:08:15.4] } 14... h6 { [%clk 0:09:02.5] } 15. f5 { [%clk 0:08:10.2] } 15... Re8 { [%clk 0:08:58] } 16. fxe6+ { [%clk 0:08:04.2] } 16... dxe6 { [%clk 0:08:57.2] } 17. Nf4 { [%clk 0:07:49] } 17... b5 { [%clk 0:08:48.6] } 18. Bb3 { [%clk 0:07:45.3] } 18... b4 { [%clk 0:08:42] } 19. Bxf6 { [%clk 0:07:34.7] } 19... gxf6 { [%clk 0:08:40.9] } 20. Nh5 { [%clk 0:07:29.4] } 20... f5 { [%clk 0:08:33.9] } 21. Rxf5+ { [%clk 0:07:24.8] } 21... Kg6 { [%clk 0:08:30.3] } 22. Rc5 { [%clk 0:07:16] } 22... Nd4 { [%clk 0:08:28.1] } 23. Nf4+ { [%clk 0:07:06.7] } 23... Kf6 { [%clk 0:08:20.3] } 24. Bc4 { [%clk 0:06:57.2] } 24... a5 { [%clk 0:08:06.3] } 25. Rf1 { [%clk 0:06:50.4] } 25... Ke7 { [%clk 0:08:02.2] } 26. Rxc7+ { [%clk 0:06:43.3] } 26... Kd8 { [%clk 0:08:00.6] } 27. Rc5 { [%clk 0:06:30.4] } 27... e5 { [%clk 0:07:37.3] } 28. Nd5 { [%clk 0:06:15] } 28... Bg4+ { [%clk 0:07:34.5] } 29. Kd2 { [%clk 0:05:59.2] } 29... Rc8 { [%clk 0:07:10.6] } 30. Rxc8+ { [%clk 0:05:50.2] } 30... Kxc8 { [%clk 0:07:09.5] } 31. Nf6 { [%clk 0:05:33.2] } 31... Be2 { [%clk 0:06:35.8] } 32. Rf2 { [%clk 0:04:50.7] } 32... Rf8 { [%clk 0:06:30.7] } 33. c3 { [%clk 0:04:44.1] } 33... bxc3+ { [%clk 0:06:25.3] } 34. bxc3 { [%clk 0:04:42.2] } 34... Nf3+ { [%clk 0:06:23.7] } 35. gxf3 { [%clk 0:04:39.5] } 35... Rxf6 { [%clk 0:06:22.8] } 36. Kxe2 { [%clk 0:04:34.9] } 36... h5 { [%clk 0:06:19.6] } 37. Rg2 { [%clk 0:04:27.9] } 37... Rf5 { [%clk 0:06:15.8] } 38. Rg8+ { [%clk 0:04:26.1] } 38... Kc7 { [%clk 0:06:14.8] } 39. Rh8 { [%clk 0:04:16.3] } 39... Kc6 { [%clk 0:06:12.6] } 40. Be6 { [%clk 0:04:10.6] } 40... Rg5 { [%clk 0:06:09.4] } 41. Rc8+ { [%clk 0:04:01.3] } 41... Kd6 { [%clk 0:06:08.4] } 42. Bc4 { [%clk 0:03:56.9] } 42... Kd7 { [%clk 0:06:05.5] } 43. Ra8 { [%clk 0:03:53.1] } 43... h4 { [%clk 0:05:59.3] } 44. Kf2 { [%clk 0:03:42.2] } 44... h3 { [%clk 0:05:58.2] } 45. Rxa5 { [%clk 0:03:39.3] } 45... Rg2+ { [%clk 0:05:55.4] } 46. Kf1 { [%clk 0:03:33.4] } 46... Rxh2 { [%clk 0:05:54.1] } 47. Rxe5 { [%clk 0:03:27.9] } 47... Rg2 { [%clk 0:05:49] } 48. Rh5 { [%clk 0:03:20.9] } 48... Rg3 { [%clk 0:05:43.1] } 49. Kf2 { [%clk 0:03:10.2] } 49... Rg2+ { [%clk 0:05:41.9] } 50. Kf1 { [%clk 0:03:04.6] } 50... Rg3 { [%clk 0:05:41.1] } 51. Rh7+ { [%clk 0:03:02.7] } 51... Kd6 { [%clk 0:05:38.6] } 52. f4 { [%clk 0:02:59.8] } 52... Rf3+ { [%clk 0:05:34.9] } 53. Kg1 { [%clk 0:02:57.6] } 53... Rg3+ { [%clk 0:05:33.9] } 54. Kh2 { [%clk 0:02:55.6] } 54... Rf3 { [%clk 0:05:21.2] } 55. Rxh3 { [%clk 0:02:51.3] } 55... Rxf4 { [%clk 0:05:20.4] } 56. a4 { [%clk 0:02:49.4] } 56... Kc6 { [%clk 0:05:18.6] } 57. a5 { [%clk 0:02:47.4] }

There's a lot of information we don't need here so onto:

Data cleaning and feature engineering

For the three main tables, (my_games, gm_games, opponents_games) the process is the same:

1. Drop null values
2. Encode game_type :
    a. blitz bullet
    b. rapid bullet
    c. blitz
    d. rapid
3. Encode outcome:
    a. 1 if player wins
    b. 0 if player loses
    c. 0.5 if draw
4. Encode moves:
    a. Go from PGN moves (very heavy) to a simple list of moves
    b. Join said list to be able to pass it in CSV
5. Encode five first move apart from the rest
6. Rename the end column to ending

For the encyclopedia_chess_openings table, there was very little to do:

1. Encode the moves corresponding to each opening in a consistent way with the previous tables

For the grand_master table:

1. Reformat the names in a consistent way

Importing Data into SQL

I ran into a wall here and got stuck a very long time on how to import my tables. I tried different methods :

1. Via the pymsql and sqlalchemy in a python script:
    - I ran into the same error:BrokenPipeError: [Errno 32] Broken pipe at each of my attempts
    - To the best of my understanding this seemed to be caused by the length of the strings containing game moves.
2. Via MySQL Workbench's import wizard : the import kept on running for hours and I had very few lines total imported in the end
3. Via an INSERT INTO query: same problem as above.
4. Last method I found minutes before despair: a LOAD DATA query. Small problem though:
    - On executing the query I systematically got the following error:
        i. Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on access.
    - In the end I had to reinstall entirely my local SQL server and run it from the terminal like so:
        i. Baudouin$ mysql --local_infile=1 -u root -password database
        ii. After creating the tables manually in MySQL Workbench, I could load the data running the following query:

```
1. LOAD DATA LOCAL INFILE '/Users/Baudouin/Desktop/Chess_exploration
   /dataset/gm_games2.csv'
2.     -> IGNORE INTO TABLE gm_games2
3.     -> FIELDS TERMINATED BY ','
4.     -> LINES TERMINATED BY '\n'
5.     -> IGNORE 1 ROWS;
```

        iii. This worked like a charm in seconds

You will find the SQL script in the github repo. This part of the project also rather simple in my mind before hand was in the end by far the longest step of all. A good lessons I suppose

SQL queries

The goal here is to create a few views that I will be needing to import to Tableau to produce visualisations that will help me verify the few hypotheses I had before starting
You will find the SQL script in the Github repo.

Apart from the queries I used to populate the table, I am now going to create views by joining table so as to cross information from different ones:

One of my first hypotheses is the importance of theory in chess: I therefore want to calculate the average number of theorical moves per grand master, opponent and I and see the effect on the elo rating

Then I would like to bulid a tool to explore a given player's repertoire, I therefore have to create a view counting openings used by each player.

I will also create a view that will enable me to spot most common openings for any given elo range.

Last, I will see if the openings used are significantly different for any given elo ranking, I must therefore create a view that calculate the average ranking of the player playing any given opening.

Visualisations

I will link Tableau to MySql and use the views created to produce some visualisations that will hopefully give me the insight I am looking for. I will share my discoveries during the presentation!