SI 330 Final Project:

Average total play times of different video games

Lawrence Zheng

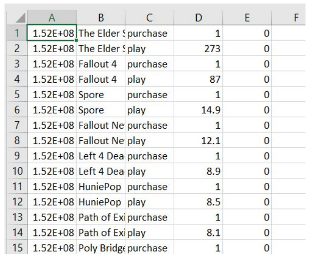
Motivation

As someone who has played video games for a long time, the total amount of time I spend on a video game would vary quite drastically. Sometimes, I would get bored of a game after less than an hour and never touch it again; other times, I could play a game for hours on end without noticing the time go by and keep coming back to the same game over the years.

For my final project, I decided it would be interesting to see how the average total play times varied between different video games and if there were any patterns which could determine which games would be played for longer. My goal was to compare the play times of games to other factors, like the genre of the game and the price, in order to see if there were any meaningful patterns that could be seen.

Data sources

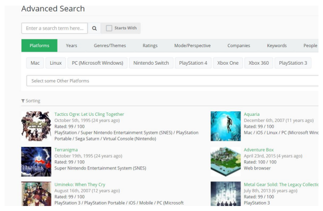
To achieve my goal, I found two data sources that matched what I needed. URLs and details on both these data sources are available as follows:



	A	B	C	D	E	F
1	1.52E+08	The Elder S	purchase	1	0	
2	1.52E+08	The Elder S	play	273	0	
3	1.52E+08	Fallout 4	purchase	1	0	
4	1.52E+08	Fallout 4	play	87	0	
5	1.52E+08	Spore	purchase	1	0	
6	1.52E+08	Spore	play	14.9	0	
7	1.52E+08	Fallout Ne	purchase	1	0	
8	1.52E+08	Fallout Ne	play	12.1	0	
9	1.52E+08	Left 4 Dea	purchase	1	0	
10	1.52E+08	Left 4 Dea	play	8.9	0	
11	1.52E+08	HuniePop	purchase	1	0	
12	1.52E+08	HuniePop	play	8.5	0	
13	1.52E+08	Path of Exi	purchase	1	0	
14	1.52E+08	Path of Exi	play	8.1	0	
15	1.52E+08	Poly Bridge	purchase	1	0	

Data source 1: Kaggle “Steam Video Games” Dataset

https://www.kaggle.com/tamber/steam-video-games/version/3
This provided a CSV file filled with the behavior of users on Steam, a popular video game distribution platform, showing 200,000 records of when a user buys a game or plays a game. The part that is important to me is that when there is a record of the user playing a specific game, the number of hours the user plays the game is also recorded.



Tactics Ogre: Let Us Cling Together
October 5th, 1995 (24 years ago)
Rated: 99 / 100
PlayStation / Super Nintendo Entertainment System (SNES) / PlayStation Portable / Sega Saturn / Virtual Console (Nintendo)

Aquaria
December 6th, 2007 (11 years ago)
Rated: 99 / 100
Mac / iOS / Linux / PC (Microsoft Wind

Terranigma
October 19th, 1995 (24 years ago)
Rated: 99 / 100
Super Nintendo Entertainment System (SNES)

Adventure Box
April 23rd, 2015 (4 years ago)
Rated: 100 / 100
Web browser

Umineko: When They Cry
August 16th, 2007 (12 years ago)
Rated: 99 / 100
PlayStation 3 / PlayStation Portable / iOS / Mobile / PC (Microsoft

Metal Gear Solid: The Legacy Collectio
July 8th, 2013 (6 years ago)
Rated: 99 / 100
PlayStation 3

Data source 2: IGDB API

https://api.igdb.com/
IGDB provides a comprehensive database with details on specific video games. I used its API to query specific details of video games, which returned JSON objects. I ended up querying the API 3600 times in my final program, with each query being a different game I wanted to look at the details for.

Data manipulation

In order for my plan to work, I specifically needed to get a list of games and the corresponding average total play time from the Kaggle dataset. Then, I would expand this list by querying the names of the games in the list through the IGDB API and joining the old play time data with new factors like the genre of the game and the release date which weren't present in the Kaggle dataset. In this way, I could get a more complete dataset suitable for my needs.

For missing data, I found that some of the factors that I originally wanted to look at, such as the price of the game, was not available through the IGDB API, so I had to adjust my plan slightly. A lot of data from the IGDB API ended up having the None label attached to it since not all fields were filled out for every video game. I started out by applying a dropna function to each column of my Pandas dataframe, but I eventually found I only really needed it for my Release Date column to avoid errors. The Kaggle dataset did not have column names for some reason, so I had to figure out what each column meant from context and apply these to my dataframe after importing - luckily, this was not too difficult.

The Kaggle dataset was easy to convert to a Pandas dataframe through a simple read_csv function. Conveniently, the dataset had a column indicating whether the specific row was recording someone purchasing a game or playing a game, so I was able to use this to filter out all the records of someone purchasing a game since this wasn't relevant to my goals. I then took out this column since I no longer needed it, alongside other columns like the user ID. I then used a groupby function to aggregate records of the same game together and get their mean, which, rather than records of individual users, was what I cared about.

For processing the data from IGDB, I used json.loads to directly load the data from the API to my program. I again had to take out the information that wasn't relevant to me. After looking at the documentation, I determined that I would only need information on the release date, genres and IGDB rating of the game (rather than things like artwork and involved companies). I then processed the date of release field, which was given as a Unix timestamp and was hard for me to read, by changing all the timestamps into years, since the year of release was what I really wanted to look at.

The joining of the data from the two sources happened quite naturally. Since I had to use the list of games from the Kaggle dataset to individual query the IGDB api anyways on a game by game basis, I set it up so that whenever a new game was queried, a dictionary containing both the original information on play times and the new information from IGDB would be added to a list, which was then converted into a dataframe containing both the information from Kaggle and IGDB.

Workflow

To get a comprehensive idea of the steps I went through for joining the two data resources, please refer to Final_Project.ipynb. A brief summary is provided here:

1. Importing and processing the first data source (from Kaggle)
2. Individually querying IGDB API using the list of games generated from Kaggle dataset and using the list of dictionaries to generate a combined dataframe with details from both datasets, processing IGDB data

Challenges encountered

A significant challenge I encountered was that I was limited to 10,000 API requests from IGDB (or I would have to pay $99 for a paid plan). Because of this, I had to be careful in planning out the number of API calls I would do in development. Since I had to query 3,600 games generated by the Kaggle dataset, I would only be able to query each game twice before my requests ran out.

The way I handled this was I started out by working with a reduced list of 50 games from the Kaggle dataset instead of the full dataset and made sure my code was working with the 50 games first. I also felt that it was important to be able to cache the query results for the full 3,600 games since it was highly likely that I would run out of query requests later and it would be helpful to be able to load from the cache instead as a failsafe method of getting the data I need.

Besides this, I was also very confused by the IGDB API at first and wasn't sure what syntax I was meant to use to get the information for a specific game given the name of the game. I eventually was able to figure this out, after much experimentation and trial and error.

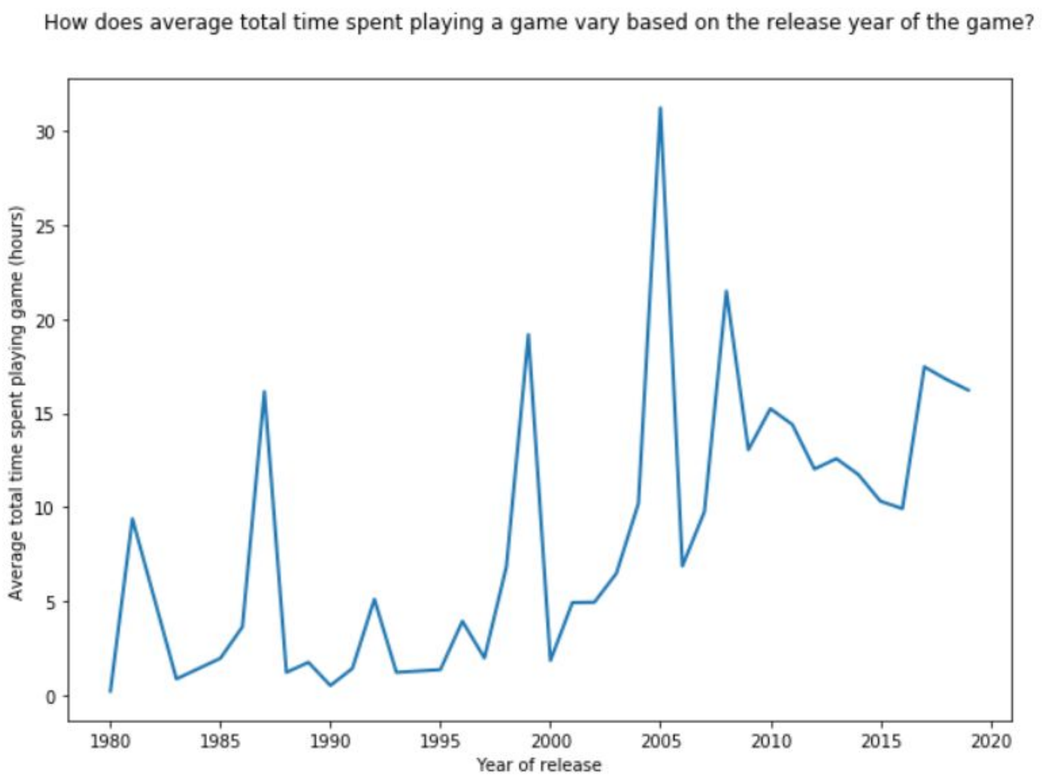
Steps for analysis

A brief summary of the steps I did for analysis/visualizations is provided here:

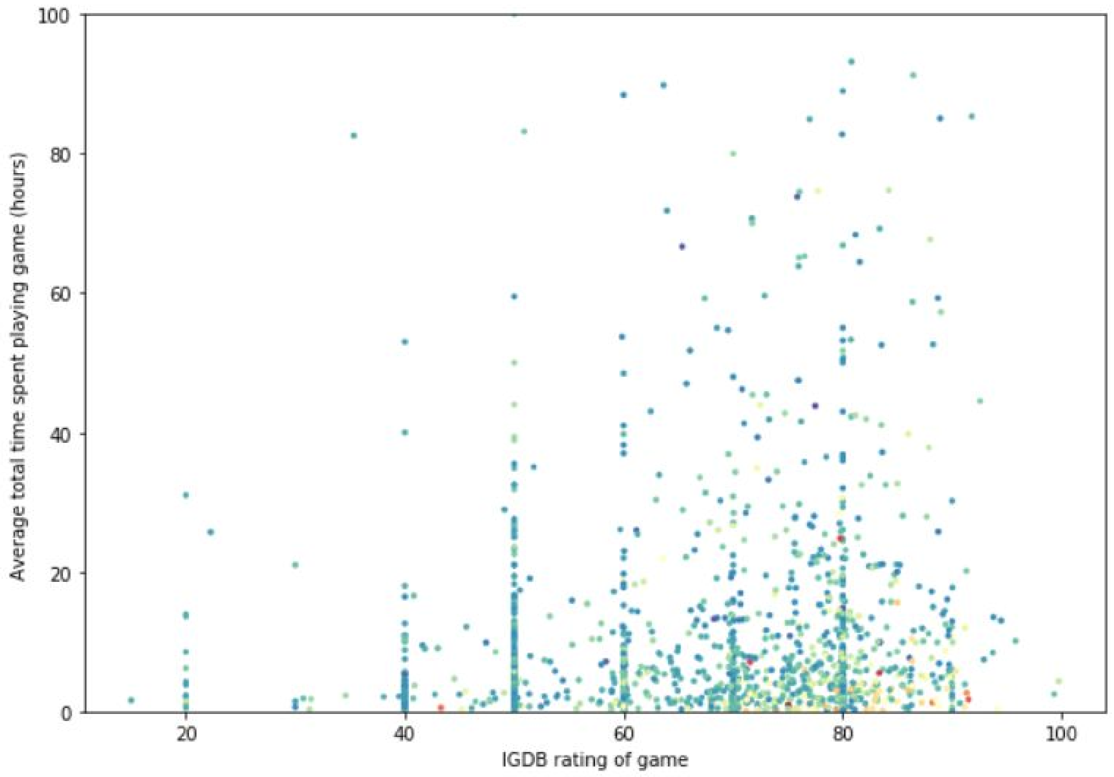
1. Combining the release date data (from IGDB) with the time played data (from Kaggle) to generate a visualization comparing the two factors
    a. Using groupby to group games with the same release year together and get their mean total play time
    b. Plotting this out on a line graph, with x axis being time and y axis being average total play time
    c. Investigating a weird anomaly (spike in line graph around year 2005)
2. Combining the game's IGDB rating with its time played data (from Kaggle) to generate a visualization comparing the two factors
    a. Taking time played and IGDB ratings from original dataframe to plot onto scatter plot, adjusting visual appearance of scatter plot to be clearer

3. Combining the game’s genres (from IGDB) with its time played data (from Kaggle) to generate a visualization to get an idea of which game genres have the highest average total time played
    a. Creating a new dataframe to separate out games with multiple genres into separate entries (previously genre would contain a list object containing different numbers - exploding entry out to multiple lines thus necessary)
    b. Using groupby to group games with the same genre together and get their mean total play time
    c. Since IGDB uses numerical codes to represent genres, it was necessary to convert these numbers into actual words that would tell a viewer what the genre was, then create new dataframe with the genres as words
    d. Creating a horizantal bar graph chart to show top 10 game genres with the highest average total time played

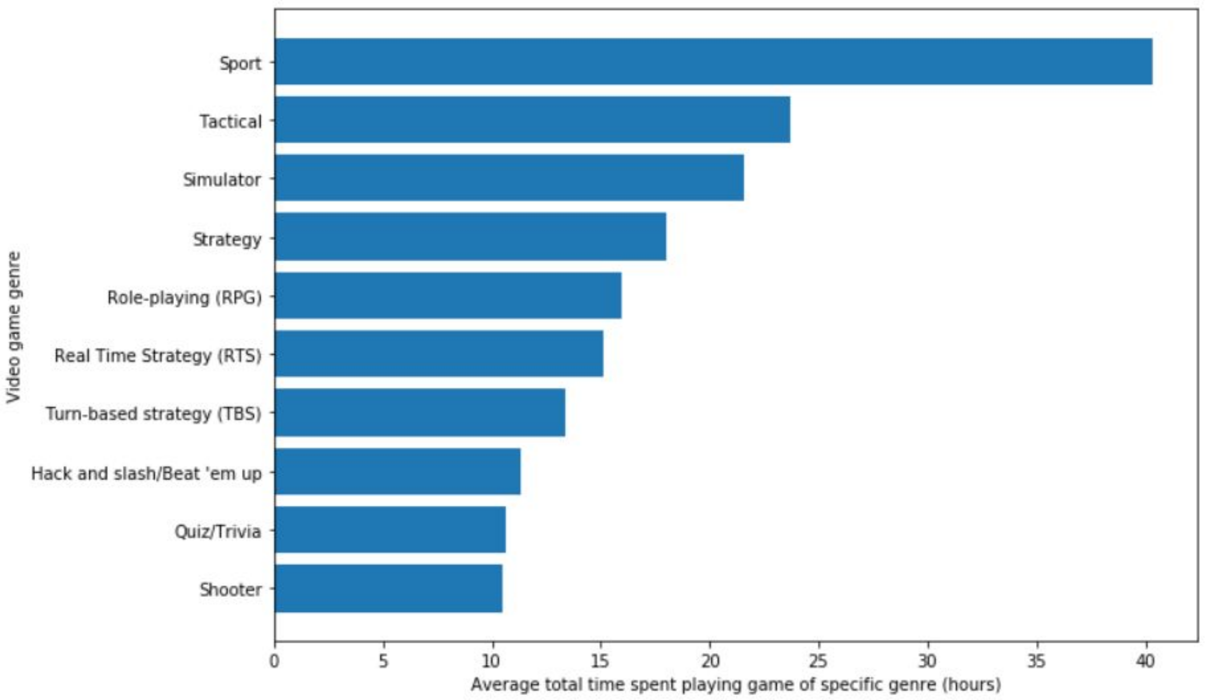
Final visualizations



How does the average total time spent on a game vary with the IGDB rating of the game?



Which genres of video games have the highest average time spent?



Interesting insights

The most interesting feature to me can be seen in the third visualization, which shows that Sport is the video game genre with by far the highest average total play time. I personally have a perception of sport video games as just something you play once in a while with friends and not something that one would play regularly, so it really surprised me that people put the most time into sport video games over genres like shooters and strategy games (which are genres that I would have expected to top out the list).

For the second visualization, it can be seen that there is a slight correlation between a game having a positive review and a game being played for a long time. This is less surprising since a positively reviewed game is more likely to be good and thus have more people wanting to play it for longer amounts of time, as opposed to finding it boring and not progressing further. However, it is still interesting to see this correlation in a visual way that confirms my assumptions. It is also interesting to see that a good chunk of games seem to have a total playtime of under 10 hours, regardless of whether it is highly reviewed.

Reflections - what didn't work?

It was a shame that I wasn't able to analyze factors that weren't present in the IGDB dataset (such as price, which was a factor I really wanted to analyze).
Besides this, the huge spikes in the first visualization (how the average total time spent on a game varies with time) did not really feel right to me. After looking into the dataset and seeing what was happening at the 2005 datapoint (the location of a huge spike), I found that it was skewed because of five games that had 100+ total hours. While it can be argued that this was valid, I wonder if the results were skewed this much because the sample size from Kaggle simply wasn't large enough and so the influence of five outliers managed to sway the overall results by a huge amount.

I also realized during the course of this project that taking a Kaggle dataset on user playtimes from Steam wouldn't be able to fully capture how much people really played games, since many may not actually use Steam to launch their games, and so the results could be greatly skewed. If I were to develop this project further, I would try to see if there was a way to get user playtimes without relying on Steam.