COMP 7150 — Data Science Project Report

Hicks, Eric
elhicks@memphis.edu

Kelly, Craig
cnkelly@memphis.edu

January 12, 2017

Abstract

The Steam video game store is a source for interesting data about video game use, including consumer behavior and industry sales. A single, usable data set is not available from the Steam Store, but there are available API's. These API's and other public resources were used to create a single, clean, flat data set for public exploration. Some initial hypotheses about what should be seen in the data were made and tested as part of this project. All code and data has been publicly available for use by others.

1 Introduction

In this paper, the path and results of the authors' dissection of the Steam platform will be detailed. The reason Steam was chosen for this project, was due to its large acceptance in the PC space. Steam is a company that sells PC, Mac, and Linux games for download over the Internet. With weekly sales, daily releases, a managed library, and no computer limit on games, it is the largest digital distributor of games and the most widely used. Additionally, Steam stores metrics on over a 100 million users on nearly 10,000 games, making it a perfect example of a feature rich dataset. However, as Steam has no saved repository of data publicly available, one was created for this project, hereby called the Steam dataset. [1]

2 Steam Dataset

This dataset and analysis project was inspired by the SteamDB project [4]. SteamDB does not provide data for download, and does not provide an API. SteamDB uses the SteamKit2 library for accessing the Steam API [12, 5]. This does not appear to be unusual; for instance the site Rhekua makes no data available either [3].

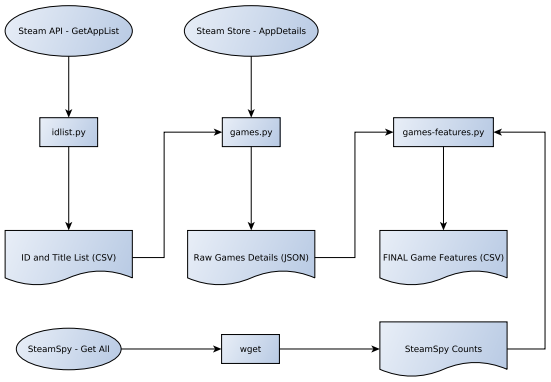
Unlike SteamDB and other services, one service with data available for download was found. The service "Steam Spy" makes their data available via API, and that API includes a full download [7]. The Steam Spy data is gathered via web scraping performed on a random sample of Steam user pages using a technique originally described by Ars Technica [6, 13]. The technique depends on the format of Steam user IDs, the URL for Steam user pages, and the data shown on a user's profile page. This dataset was used to include owner and player counts in our final data set of video game features (please see section 2.1).

2.1 Acquisition

SteamKit2 is designed for the entire API made available by Steam much of which requires a developer/-partner agreement with Steam [12, 2]. The authors chose to use SteamKit2 as a starting point, but to write their own, slim data acquisition library.

The entire process of acquiring, cleaning, and formatting is automated in a Makefile in the "data" subdirectory of the public GitHub repository for this project [9]. This process is shown visually in figure 1.

Figure 1: Data flow used to produce the final data set



The official Steam API has a few endpoints that are usable without an official API key. One of them provides a list of applications identifiers and titles. These "application" IDs actually refer to anything made available in the Steam store, including movies, video trailers, downloadable content, etc. This list is saved as a two column CSV named "idlist.csv".

Once an ID list is obtained, the bulk of the data retrieval process can begin. The script "games.py" was written to get the details of each ID one at a time from the Steam Store API. Each response is saved in it's original JSON form as a line in the file "games.json".

At one time bulk retrieval was allowed, but that is no longer the case so one request must be made per ID. In addition, the Steam Store throttles requests to around 200 queries per five minutes according to online sources. This limit is honored by pausing for five minutes and ten seconds every 150 requests. Even so, occasionally the Steam Store would stop accepting requests from the IP address used by the script. As a result, the script was updated to handle repeated execution. When running, all IDs already in the output file are skipped.

New games are constantly added to Steam. In addition, the detail data retrieved is constantly changing as users play games, vendors alter prices, etc. In an attempt to get the most and best data, the retry logic above was used to check and retrieve all current data from Steam on two occasions after the initial data gathering was complete. Because the code is publicly available, this same process could be used in the future to keep the data "fresh".

An additional file of details per Steam ID is downloaded from Steam Spy using the "wget" command line tool. Around 13 numeric fields are provide for Steam IDs in a single, huge JSON record.

The final stage of execution is the script "game-features.py", which creates the final data set in "game-features.csv" using the data files already created. This stage is described in section 2.2.

2.2 Cleaning and Formatting

The "raw" detail data in "games.json" includes the response for every ID found and saved in "idlist.csv" (see section 2.1). The ID list retrieved from Steam appears to be exhaustive. Some IDs never receive a valid detail record; it is assumed that this is because the record is deleted. In addition, many of the records are not for video games, so only records with a type of "game" are retained.

The remaining JSON records are transformed in "flat" CSV records. Four columns from the Steam Spy data are added by joining on the App ID. The resulting records are written to the flat file "games-features.csv".

The final file has 78 columns. There are five distinct "data types" that were used for the columns based on the data displayed:

1. Text — string value, sanitized for processing. Described further in section 2.3
2. TextualCount — the number of unique, non-empty strings in the original list
3. Integer — integral value, missing values are 0
4. Float — floating point value, missing values are 0.0
5. Boolean — field has one of two literal values: True or False

A complete list of columns, their types, and a brief description is available in the file "columns.md" in the public GitHub repository [8].

2.3 String Processing

Some of the data available for a video game (including the system requirements for various platforms) is in unstructured text fields. Where feasible, this data was included in the final CSV data set. However, the text was often hard to use for practical work. For instance, the system requirements often included HTML markup for display.

As a result, some non-trivial string processing was used to insure easy reading of the string data in the final data set. If access to the original, unchanged text is ever required, it is present in the original "games.json" file created as part of the data download process.

Some notes:

- String comparisons were done on versions of the string converted to lowercase with all leading and trailing whitespace removed.
- The JSON data is retrieved in UTF-8, so all textual data begins in UTF-8 before any processing takes place.

The full text processing process is:

1. Unicode sequences that are not in ASCII are transliterated via the library "unidecode" [15].

2. Any HTML entities (e.g. > or ") are unescaped using the Python standard library ("html.unescape").

3. The HTML tag stripping routine from above is re-run to strip out any HTML tags introduced via the unescaping.

4. Comma's, single-quotes, double-quotes, tabs, and line break characters are all removed from the string.

5. All leading and trailing whitespace is stripped

6. If the final string is empty, a default value is returned. Currently the default value is a single space, which insures that libraries like "pandas" treat the field as text when the CSV is loaded (instead of assuming a missing numeric value).

2.4 Metacritic

While many of the data dimensions are self-explanatory, Metacritic might require some explanation. Metacritic provides a rating service for video games, movies, television shows, and music [11]. The Metacritic score is a proprietary weighted average of various professional critics and publications, scaled from 1 to 100 [10]. It is one of the most frequently used metrics for video game quality [14].

2.5 Caveats

The process used could not capture all data dimensions used by Steam. Some are not recorded and others can only be accessed by an administrator account. The number of owners for each game could not be acquired, but has been estimated using an outside site [7]. Recommendations were collected as a total of reviews rather than as two separate values for positive and negative reviews because the API used does not provide review sentiment. Steam does not store historical price data, so sale history and price trends can not be analyzed from the data. These limitations did not stop the authors from gaining interesting insight from the data.

3 Predictions

As both authors were familiar with Steam before the start of the project, they made some predictions of what would be found in the data:

1. SteamOS games available on Steam would be a perfect match to Linux games available.

2. That free games get more reviews per game and have lower ratings than paid games.

3. The most recommended and the highest rated genre is action.

4. That Metacritic scores are an inverse bell curve when sorted by recommendation, i.e. lower and higher scoring games would have more recommendations that games with a middle score.

4 Testing

Both authors ran different models on the data independent of each other to cover the most ground. Every feature pair was plotted as a scatter plot to gain a basic understanding of feature relations alongside reading through the raw data. From here other options were tried.

5 Results

The authors found results corresponding to all predictions. Some other interesting results were found as a result of exploratory analysis.

5.1 SteamOS and Linux

As a sanity check, the authors insure that a simple prediction was verifiable by the dataset. By definition, all Steam games available on the Linux platform must also be on the SteamOS platform because SteamOS is the only way to run a Steam game on Linux. The data verified this "prediction".

5.2 Free vs Non-Free Games

Free games do get more recommendations than paid games (see figure 2). However, they are also rated lower than paid games (see figure 3). In fact, "free to play" as a genre received the most feedback via recommendations (see figure 4), but had the lowest rating scores (see figure 5). As a result, we conclude

that free games tend to have negative reviews; in fact, the plurality of reviews for free games might well be negative.

Figure 2: Count of recommendations from users on Free vs Non-Free games

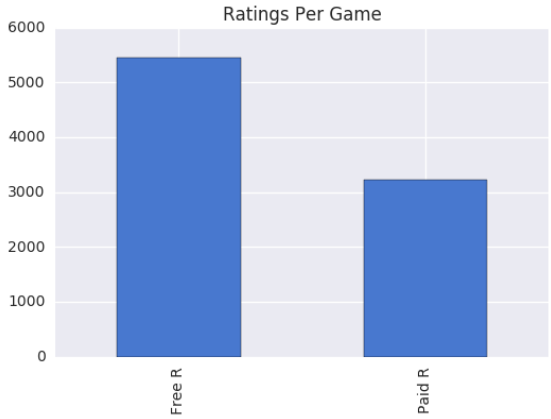
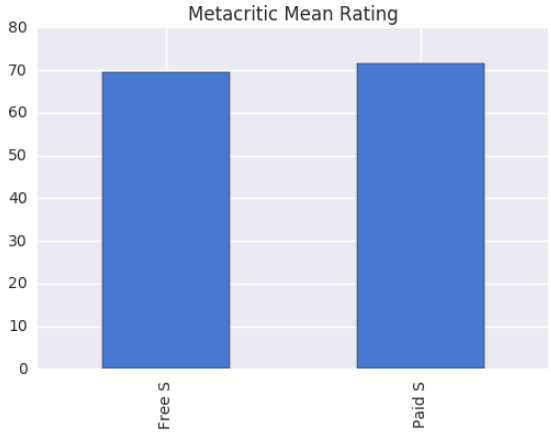


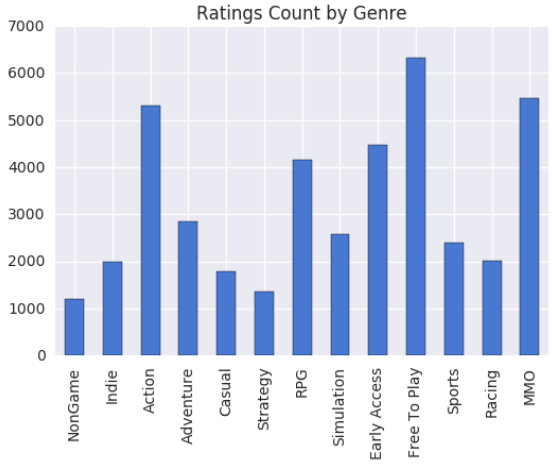
Figure 3: Metacritic mean score on Free vs Non-Free games



5.3 Genre

The most recommended genre was free to play and not action. The least recommended genre was non-game software (see figure 4). The highest scoring genre was sports instead of action. The lowest scoring genre was free to play (see figure 5).

Figure 4: Count of recommendations from users by genre



5.4 Recommendations, Ratings, and Price

Although not part of the original prediction, some relationships (or lack thereof) were noticed. Pricing compared to Metacritic scores is mostly uniform. The gaps for certain prices are almost entirely due to Steam's pricing structure. Pricing compared to user recommendations is also nearly uniform. There is a small increase in recommendations for cheaper games, but it is not significant. Please see figures 6, 7, and 8, which show these relationships for filtered data ranges. The full range of data can be seen in figures 9, 10, and 11.

5.5 Recommendations and Player Count

In addition to the predictions made, some exploratory data analysis was performed. Specifically, many of the numeric fields were examined for interesting distributions and correlations. Figure 12 shows four of these variables.

As can be seen from the figure, it appears that there is a strong correlation between Recommendation Count and the number of Players that a video game has. Although not an altogether surprising conclusion, it is not an obvious one. While more players indicates more potential opportunities for recommendations, the player numbers include people who were given a "free weekend" to play the game [7]. In addition, it seems intuitive that engagement across Steam

Figure 5: Metacritic mean score by genre

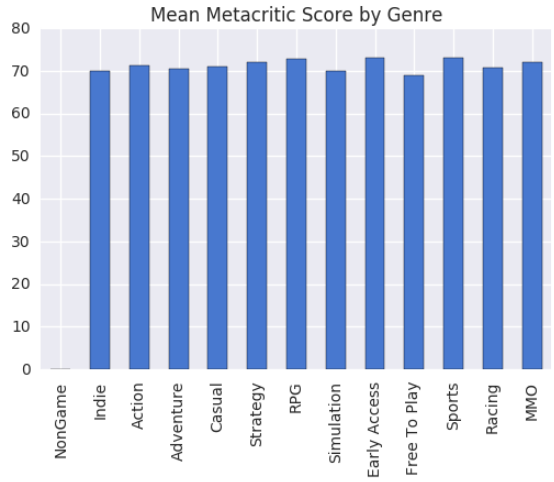
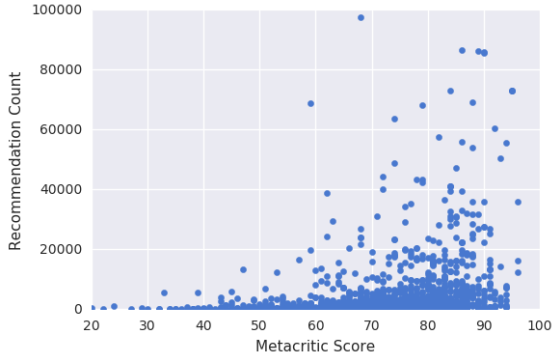


Figure 6: Count of User Recommendations to Metacritic Score



games would vary widely; perhaps too widely for such a strong correlation.

To examine this relationship, it was necessary to insure that only games with valid data for both variables were examined. In addition, it appeared that these variables had some fairly significant outliers, so an upper bound of the 90th percentile was used. The subset of games with non-zero values for both variables that were at or below the 90th percentile for both variables was used to examine the distributions as shown in figure 13.

Both variables appear to have similar shapes (a "long-tailed" distribution). They also appear to have fairly smooth distributions. It would appear that the potential correlation might be correct. However, as can be seen in the joint distribution graph in figure 14, there is not much of a relationship. A Pearson's R of 0.26 (regardless of the small p-value) does

Figure 7: Metacritic Score to Initial Price

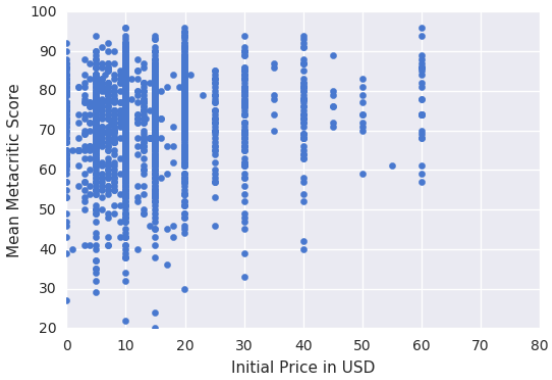
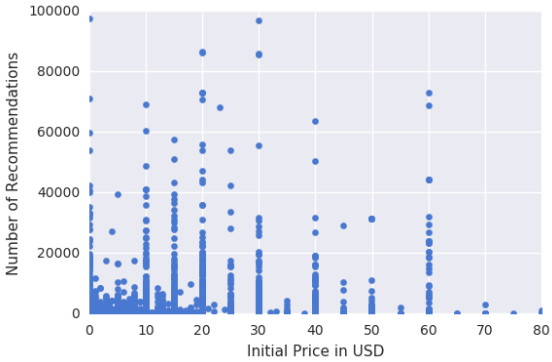


Figure 8: Initial Price to Count of User Recommendations



not indicate a very strong relationship. Nonetheless, the regression line and error bounds do seem to indicate that there might be a potential relationship for games with less than 10000 players.

5.6 Required Specifications

Of the features found in the Steam database, there are 6 relating to the minimum and suggested requirements to run a game sorted by OS. These fields typically include: OS, processor, memory, graphics, storage, and additional notes, all written in full text. As these fields are newer features in Steam, many games do not have data in them. Of the subset of games with non-empty fields: less than 1% have Linux requirements, roughly 10.5% have Mac requirements, and 100% have Windows requirements. However, only 23% have both minimum and suggested fields. Additionally, nearly 17% of these entries for Windows have missing or incorrect fields. After sorting these fields into similar groupings, the only interesting pat-

terns found were that the minimum required specifications to play a game increased nearly uniformly with the passage of time, i.e. newer games had higher requirements.

6 Future Work

The current data collection technique provides a single snapshot in time of Steam game data. Multiple snapshots collected over time could be used to to analyze changing trends, especially variables like price and number of players. In addition, a more fine-grained approach to recommendations could be used to determine if certain genres get more positive reviews than others. The same analysis could be performed on Metacritic scores and positive recommendations.

There is also a great deal of potential in the unstructured text data gathered. If nothing else, it represents thousands of descriptions of system requirements that could be used for various natural language processing tasks. In particular, it could serve as a specialized corpus.

7 Breakdown of Work Involved

7.1 Craig

- Editing of APIs
- Gathering of data
- Cleaning and formatting of data
- Analysis of data
- Writing of paper
- Manipulation of Latex and included figures

7.2 Eric

- Determining of project type and scope
- Finding of main API
- Hypothesis construction
- Analysis of data
- Writing of paper
- Editing of paper

8 Conclusion

After finding a suitable platform to mine data from, the authors acquired raw data from Steam. This data was cleaned, formatted, and processed to find several interesting results. While all of the authors' predictions were not proven in testing, the ones that were wrong proved to be the most surprising.

References

[1] Valve Corporation. *Steam*. URL: http://store.steampowered.com/.

[2] Valve Corporation. *Steam Community : Steam Web API Documentation*. URL: https://steamcommunity.com/dev.

[3] Dillon DeLoss. *Rhekua*. URL: http://steamsales.rhekua.com/.

[4] Pavel Djundik and Martin Benjamins. *SteamDB*. URL: https://steamdb.info/.

[5] Pavel Djundik and Martin Benjamins. *SteamDB*. URL: https://steamdb.info/faq.

[6] Sergey Galyonkin. *About — Learn About Steam Spy*. URL: http://steamspy.com/about.

[7] Sergey Galyonkin. *Steam Spy*. URL: http://steamspy.com/.

[8] Eric Hicks and Craig Kelly. *Column Descriptions for games-features.csv*. URL: https://github.com/CraigKelly/steam-data/blob/master/data/columns.md.

[9] Eric Hicks and Craig Kelly. *Steam-Data Repository*. URL: https://github.com/CraigKelly/steam-data.

[10] CBS Interactive. *How we create the Metascore magic*. URL: http://www.metacritic.com/about-metascores.

[11] CBS Interactive. *Metacritic — About Us*. URL: http://www.metacritic.com/about-metacritic.

[12] opensteamworks.org. *SteamKit2*. URL: https://github.com/SteamRE/SteamKit.

[13] Kyle Orland. *Introducing Steam Gauge: Ars reveals Steams most popular games*. URL: http://arstechnica.com/gaming/2014/04/introducing-steam-gauge-ars-reveals-steams-most-popular-games/.

[14] Mike Rose. *Metacritic is here to stay, but can we fix it?* URL: http://www.gamasutra.com/view/news/173879/Metacritic_is_here_to_stay_but_can_we_fix_it.php.

[15] Tomaz Solc. *Unidecode*. URL: https://pypi.python.org/pypi/Unidecode.

Figure 9: Count of User Recommendations to Metacritic Score

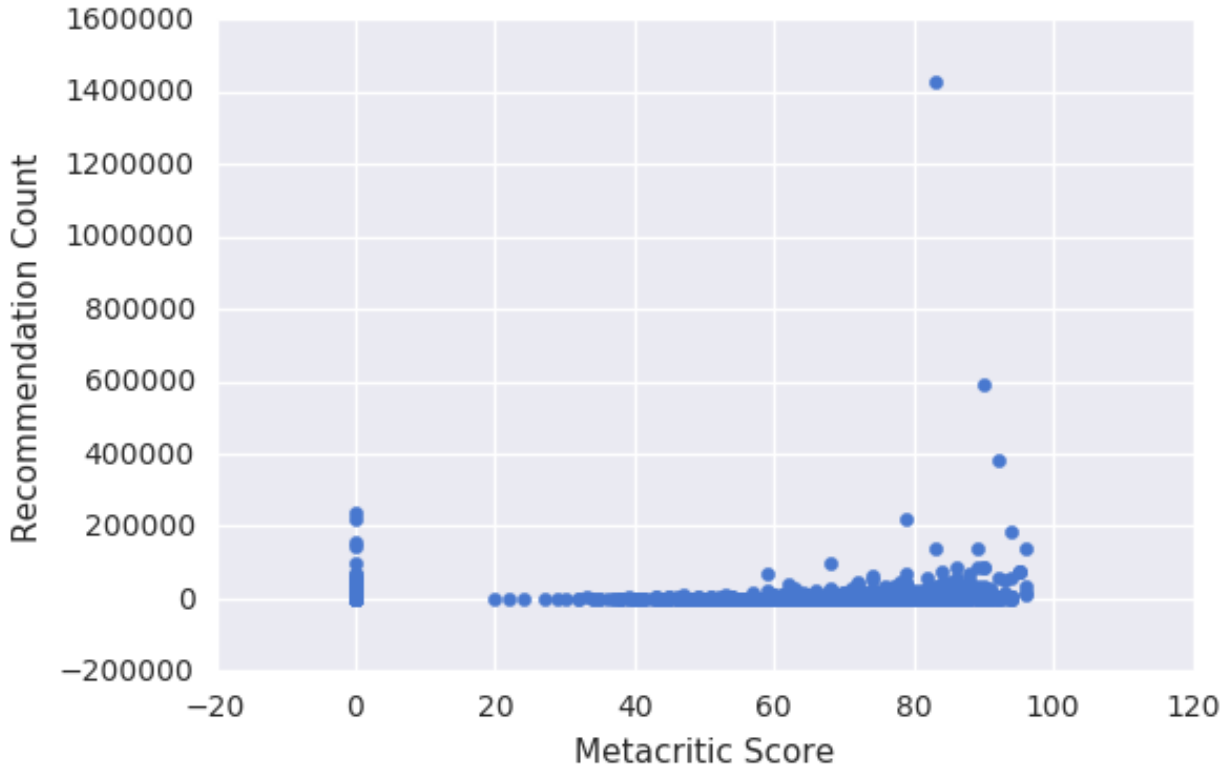


Figure 10: Metacritic Score to Initial Price

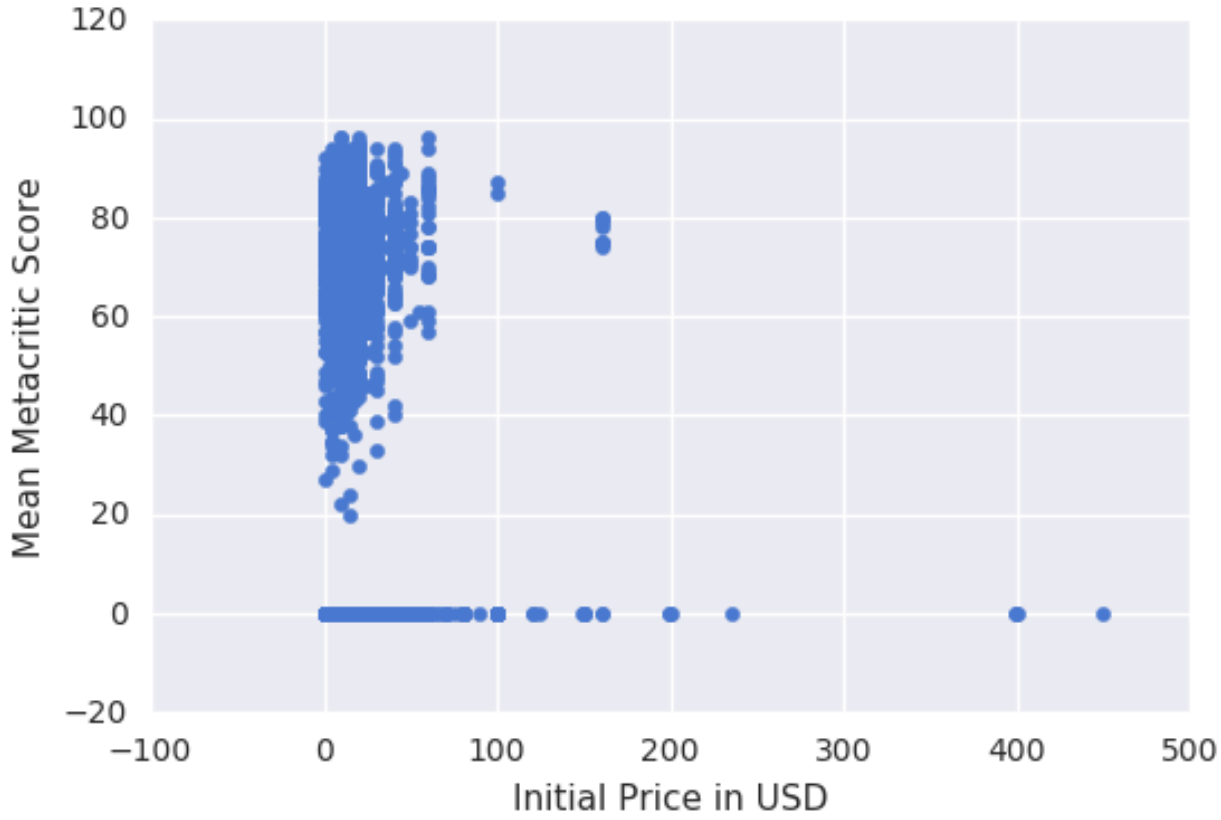


Figure 11: Initial Price to Count of User Recommendations

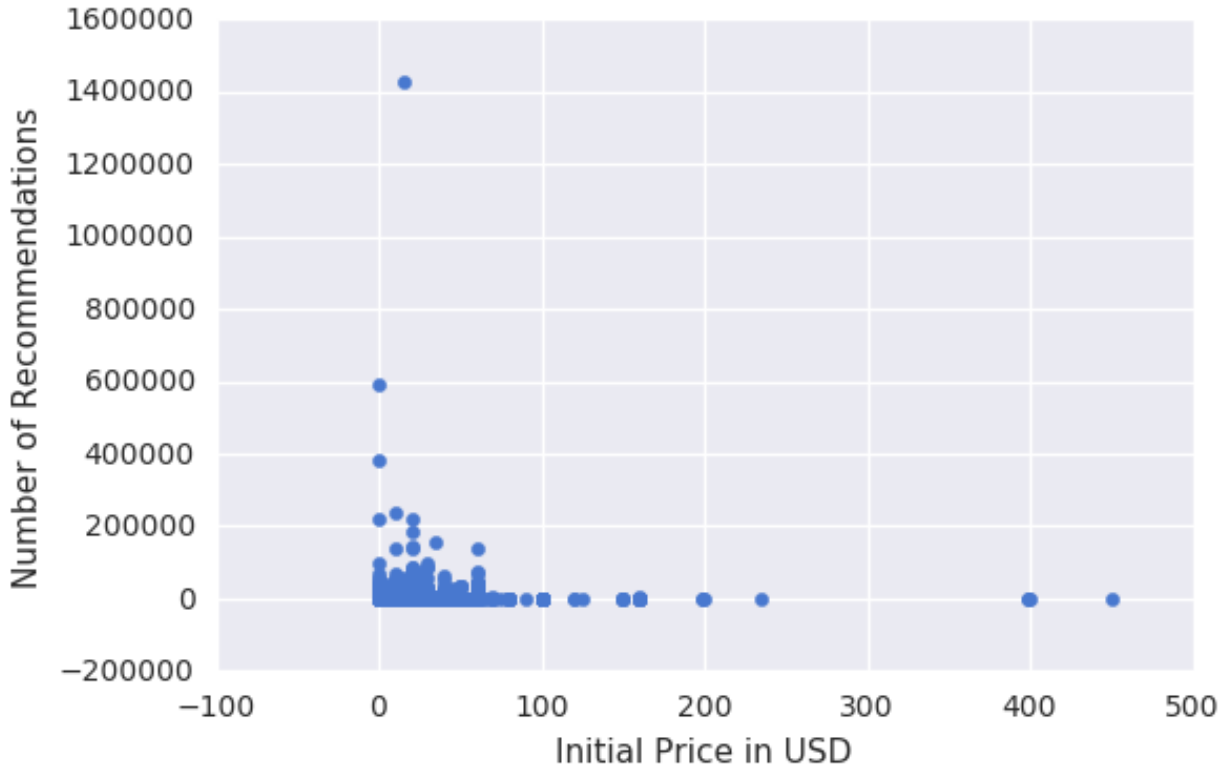


Figure 12: Exploring some of the more promising numeric columns

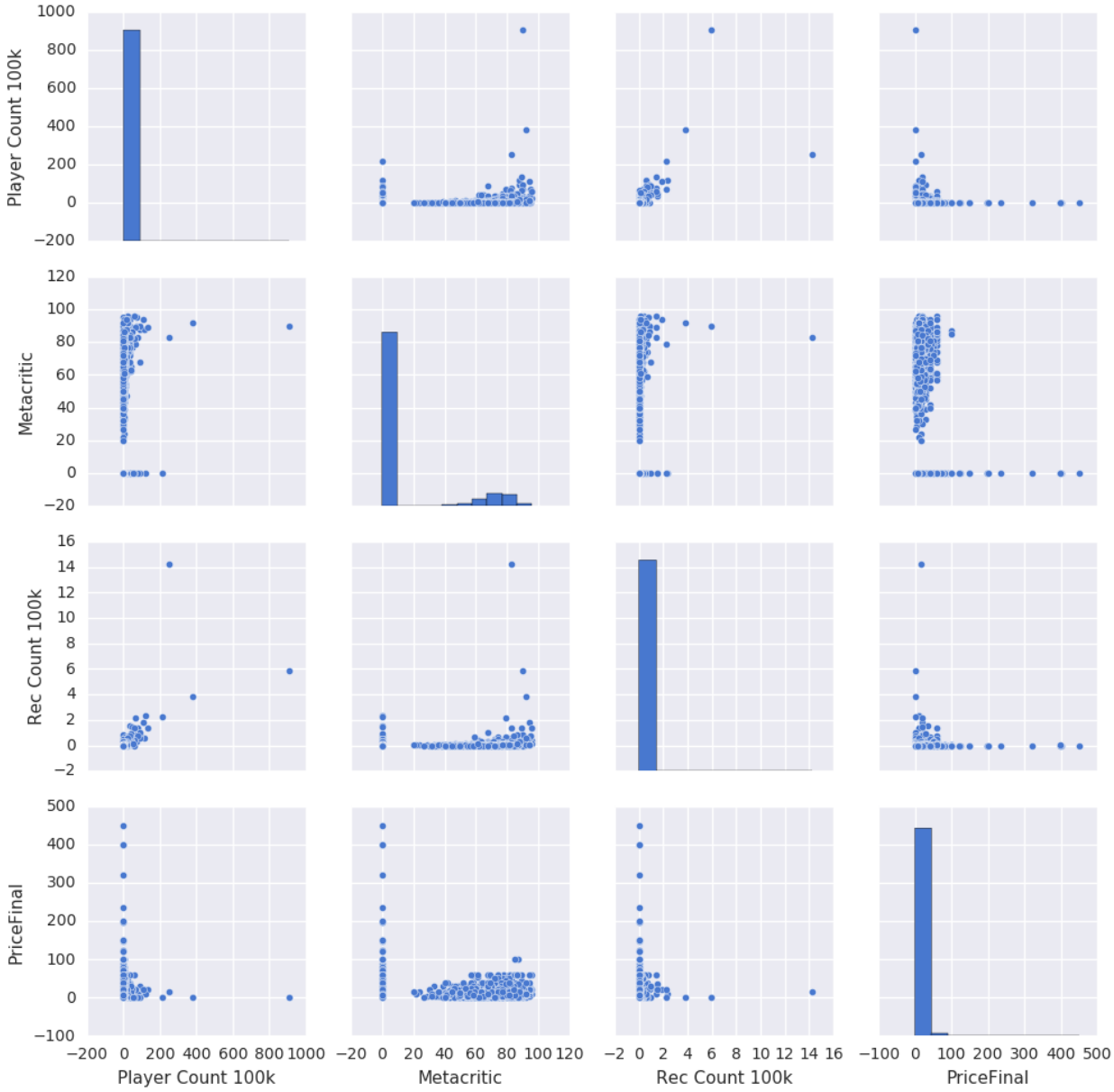


Figure 13: Player Count and Recommendation Count Distribution Estimates (only non-zero 90th percentile)

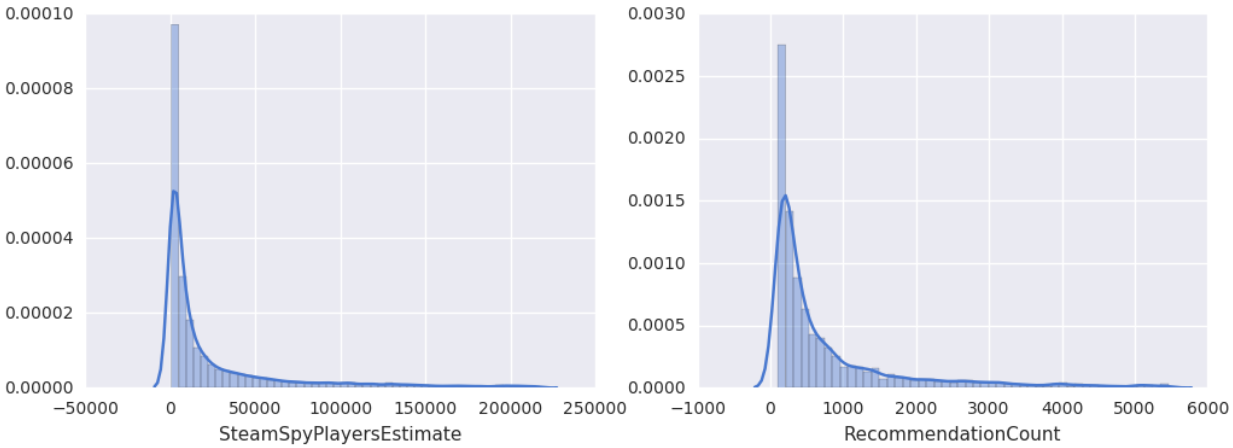


Figure 14: Player Count and Recommendation Count Joint Distribution and Regression (only non-zero 90th percentile)