Part I - Steam Games from 2013 to 2023 success exploration

by Hessa

Introduction

The dataset compile the information of the releases that occurred between 2013 and 2023 of the most successful platform for selling video games on PC, Steam. With an estimated number of sales, release dates, among other kinds of variables.

I am going to try to find the factors that make a video game successful from a commercial and critical point of view.

Preliminary Wrangling

The processed data is the work of Kaggle user 'Terenci Claramunt' who was responsible for cleaning and splitting the datasets collected by Steam API and Steam Spy.

> Reference link: https://www.kaggle.com/datasets/terencicp/steam-games-december-2023

First of all, the particularities will be explored, the important variables will be summarized and unified in a single Dataset.

```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

Data upload

- games.csv
- t-games-categories.csv
- t-games-tags.csv

```
# Read CSV files in a DataFrame
df_main_games = pd.read_csv('/content/datafiles/games.csv')
df_categories_games = pd.read_csv('/content/datafiles/t-games-categories.csv')
df_tag_games = pd.read_csv('/content/datafiles/t-games-tags.csv')
```

df_main_games

Features

- **name:** Game title.
- **release_date:** Date the game was published.
- **price:** Price in USD.
- **positive:** Number of positive reviews.
- **negative:** Number of negative reviews.
- **app_id:** Unique ID for games. (KEY)
- **min_owners:** Estimated sales (minimum), from SteamSpy.
- **max_owners:** Estimated sales (maximum), from SteamSpy.
- **hltb_single:** Game duration, hours, from HowLongToBeat.

```
df_main_games.describe()# Display the summary of the DataFrame
```

	price	positive	negative	app_id	min_owners	max_owners	hltb_single
count	60952.000000	6.095200e+04	60952.000000	6.095200e+04	6.095200e+04	6.095200e+04	12972.000000
mean	7.819159	1.045975e+03	193.455326	1.165637e+06	5.489319e+04	1.353160e+05	7.633595
std	9.756732	1.498527e+04	4408.960253	5.986746e+05	6.753193e+05	1.451847e+06	11.943980
min	0.000000	0.000000e+00	0.000000	5.700000e+02	0.000000e+00	2.000000e+04	1.000000
25%	1.990000	4.000000e+00	1.000000	6.760850e+05	0.000000e+00	2.000000e+04	1.000000
50%	4.990000	1.600000e+01	4.000000	1.095830e+06	0.000000e+00	2.000000e+04	3.000000
75%	9.990000	8.000000e+01	24.000000	1.579002e+06	2.000000e+04	5.000000e+04	8.000000
max	299.900000	1.477153e+06	895978.000000	2.690780e+06	1.000000e+08	2.000000e+08	100.000000

```
df_main_games.head(10)# Display the first 10 rows of the DataFrame
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single
0	Train Bandit	Oct 12, 2017	0.99	53	5	655370	0	20000	NaN
1	Henosis™	Jul 23, 2020	5.99	3	0	1355720	0	20000	NaN
2	Two Weeks in Painland	Feb 3, 2020	0.00	50	8	1139950	0	20000	NaN
3	Wartune Reborn	Feb 26, 2021	0.00	87	49	1469160	50000	100000	NaN
4	TD Worlds	Jan 9, 2022	10.99	21	7	1659180	0	20000	NaN
5	MazM: Jekyll and Hyde	Apr 2, 2020	14.99	76	6	1178150	0	20000	7.0
6	Deadlings: Rotten Edition	Nov 11, 2014	3.99	225	45	320150	50000	100000	4.0
7	WARSAW	Oct 2, 2019	23.99	589	212	1026420	20000	50000	5.0
8	Cthulhu Realms	Jul 1, 2016	0.00	147	58	485000	50000	100000	3.0
9	Clockwork Dungeon	Aug 27, 2021	1.99	5	0	1620060	0	20000	NaN

Pasos siguientes: Generar código con `df_main_games` Ver gráficos recomendados New interactive sheet

df_categories_games

Features

- **app_id:** Unique ID for games.
- **categories:** Game features.

```
df_categories_games.describe()
```

	app_id
count	2.105200e+05
mean	1.055516e+06
std	5.879844e+05
min	5.700000e+02
25%	5.616100e+05
50%	9.577150e+05
75%	1.453205e+06
max	2.690780e+06

```
df_categories_games.head(10)
```

	app_id	categories
0	655370	Single-player
1	655370	Steam Achievements
2	655370	Full controller support
3	655370	Steam Leaderboards
4	655370	Remote Play on Phone
5	655370	Remote Play on Tablet
6	655370	Remote Play on TV
7	1355720	Single-player
8	1355720	Full controller support
9	1139950	Single-player

df_tag_games

Features

- **app_id:** Unique ID for games.
- **tags:** User defined tags. (KEY)
- **tag_frequencies:** Number of user votes for a tag in a game.

```
df_tag_games.describe()
```

	app_id	tag_frequencies
count	7.774010e+05	777401.000000
mean	1.289685e+06	118.540383
std	6.125159e+05	350.977533
min	5.700000e+02	1.000000
25%	7.858500e+05	24.000000
50%	1.303950e+06	64.000000
75%	1.727520e+06	153.000000
max	2.690780e+06	62902.000000

```
df_tag_games.head(10)
```

	app_id	tags	tag_frequencies
0	655370	Indie	109
1	655370	Action	103
2	655370	Pixel Graphics	100
3	655370	2D	97
4	655370	Retro	93
5	655370	Arcade	86
6	655370	Score Attack	84
7	655370	Minimalist	82
8	655370	Comedy	76
9	655370	Singleplayer	69

Unify

We unify the data into one

```
# Merge df_main_games-df_categories_games
# Merge df_main_games-df_categories_games
df_merged_c = pd.merge(df_main_games, df_categories_games, on='app_id', how='left')
grouped_data_c = df_merged_c.groupby('app_id')['categories'].apply(list).reset_index()
# Merge df_main_games-df_tag_games
df_merged_t = pd.merge(df_main_games, df_tag_games, on='app_id', how='left')
grouped_data_t = df_merged_t.groupby('app_id')['tags'].apply(list).reset_index()
# Final Merge
df_main_games = pd.merge(df_main_games, grouped_data_c, on='app_id')
df = pd.merge(df_main_games, grouped_data_t, on='app_id')
df
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single	categories	tags
0	Train Bandit	Oct 12, 2017	0.99	53	5	655370	0	20000	NaN	[Single-player, Steam Achievements, Full contr...	[Indie Action, Pixe Graphics 2D, Retro Arc..
1	Henosis™	Jul 23, 2020	5.99	3	0	1355720	0	20000	NaN	[Single-player, Full controller support]	[2D Platformer Atmospheric Surreal Mystery,..
2	Two Weeks in Painland	Feb 3, 2020	0.00	50	8	1139950	0	20000	NaN	[Single-player, Steam Achievements]	[Indie Adventure Nudity Violent Sexua Con..
3	Wartune	Feb 26, 2021	0.00	87	49	1469160	50000	100000	NaN	[Single-player, Multi-player,	[Turn-Basec Combat

Pasos siguientes: Generar código con df Ver gráficos recomendados New interactive sheet

```
# DataFrame Save
df.to_csv('games_cleaned.csv')
```

What is the structure of your dataset?

The dataset contains 60,952 rows and 11 columns. The columns are as follows:

name (object): Name of the game

release_date (object): Release date of the game

price (float): Price of the game

positive (int): Number of positive reviews

negative (int): Number of negative reviews

app_id (int): Unique identifier for the game

min_owners (int): Minimum number of owners

max_owners (int): Maximum number of owners

hltb_single (float): How long to beat the game in single-player mode

categories (object): Categories of the game

tags (object): Tags associated with the game

What is/are the main feature(s) of interest in your dataset?

The main features of interest in the dataset are the price, positive and negative reviews, and the number of owners (min_owners and max_owners). These features provide insights into the popularity, reception, and economic aspects of the games.

What features in the dataset do you think will help support your investigation into your feature(s) of interest?

Release Date: This will help analyze trends over time.

Categories and Tags: These can help segment the data to see how different types of games perform.

Price: Analyzing the relationship between price and reviews/ownership can provide insights into pricing strategies.

Review Counts (Positive and Negative): These will help assess the overall reception and satisfaction of the games.

Ownership Numbers (Min and Max Owners): These are crucial for understanding the market reach and popularity of the games.

Univariate Exploration

Function to display the distribution of a variable in the dataset.

The most relevant is to visualize the price, release date and duration columns. Their independent properties are interesting to evaluate.

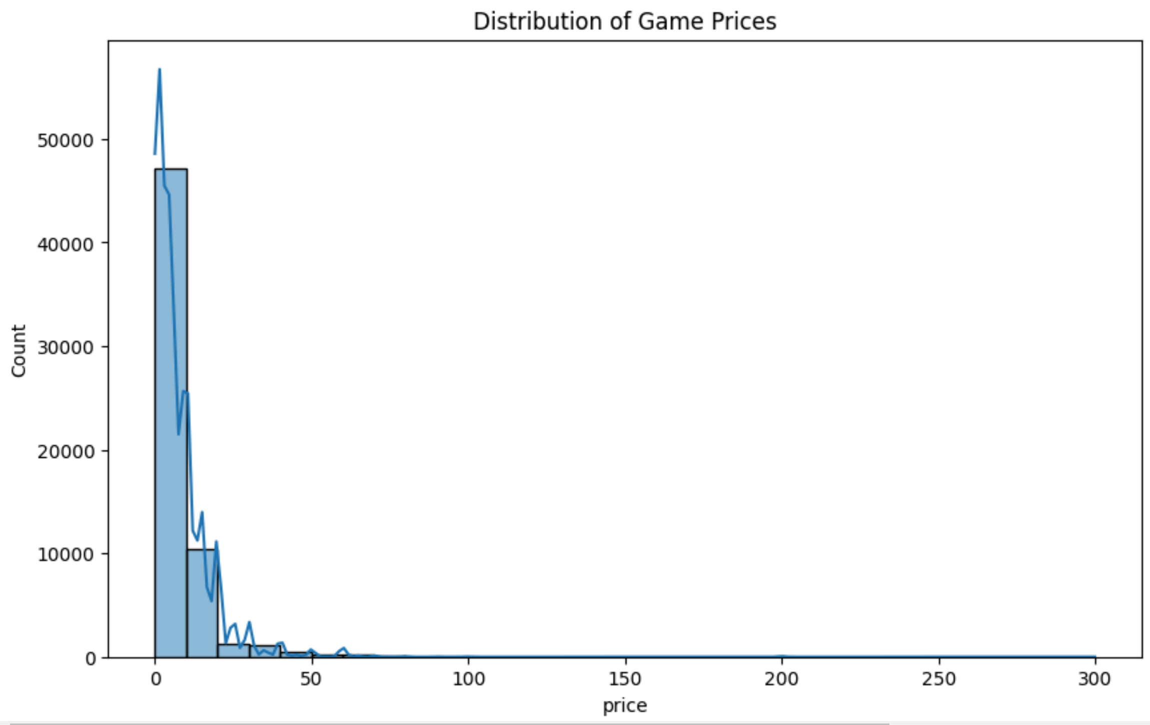
```
def chart_univariate(column_name:str,n:int,tag=''):
    # Set up the plot
    plt.figure(figsize=(10, 6))
    sb.histplot(df[column_name], bins=n, kde=True)

    # Add labels and title
    plt.xlabel(column_name)
    plt.ylabel('Count')
    plt.title(f'Distribution of Game {tag}')
```

```
df.describe()
```

	price	positive	negative	app_id	min_owners	max_owners	hltb_single
count	60952.000000	6.095200e+04	60952.000000	6.095200e+04	6.095200e+04	6.095200e+04	12972.000000
mean	7.819159	1.045975e+03	193.455326	1.165637e+06	5.489319e+04	1.353160e+05	7.633595
std	9.756732	1.498527e+04	4408.960253	5.986746e+05	6.753193e+05	1.451847e+06	11.943980
min	0.000000	0.000000e+00	0.000000	5.700000e+02	0.000000e+00	2.000000e+04	1.000000
25%	1.990000	4.000000e+00	1.000000	6.760850e+05	0.000000e+00	2.000000e+04	1.000000
50%	4.990000	1.600000e+01	4.000000	1.095830e+06	0.000000e+00	2.000000e+04	3.000000
75%	9.990000	8.000000e+01	24.000000	1.579002e+06	2.000000e+04	5.000000e+04	8.000000
max	299.900000	1.477153e+06	895978.000000	2.690780e+06	1.000000e+08	2.000000e+08	100.000000

Question: What is the distribution of game prices?

```
chart_univariate('price',30,'Prices')
```

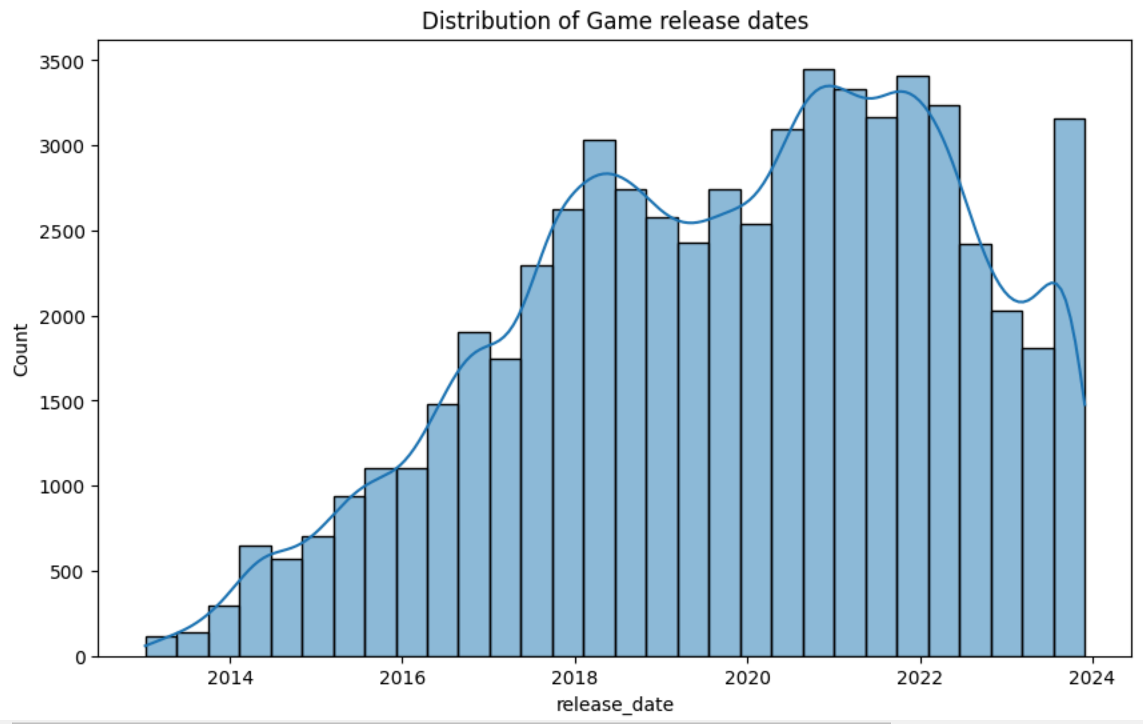


Observations:

- The mean price is 7.82 with a standard deviation of 9.76.This indicates a wide variety of pricing strategies, with some games being priced significantly higher than others.
- The price ranges from 0 to 299.90. The distribution shows a concentration of lower-priced games
- The 25th percentile (Q1) is 1.99, the median (Q2) is 4.99, and the 75th percentile (Q3) is 9.99.
- Prices above 24.98 or below -13.00 can be considered outliers based on the IQR method (Q1 - 1.5 * IQR and Q3 + 1.5 * IQR).

Question: How are the release dates of the games distributed?

```
df['release_date'] = pd.to_datetime(df['release_date'], errors='coerce').dropna()
chart_univariate('release_date',30,'release dates')
```



Observations:

- There is a noticeable upward trend in the number of game releases from 2012 to 2018. This period marks a significant growth in the number of games being released each year.
- After 2018, there is a slight decline in the trend and then fluctuations in the number of releases. However, the overall count remains relatively high.
- There is a resurgence in game releases post-2020 (COVID), with a high number of releases continuing into 2021 and 2022.
- The distribution shows that the industry has become more prolific over time, with more games being developed and released each year.

Question: What is the distribution of the average length of the games (to complete)?

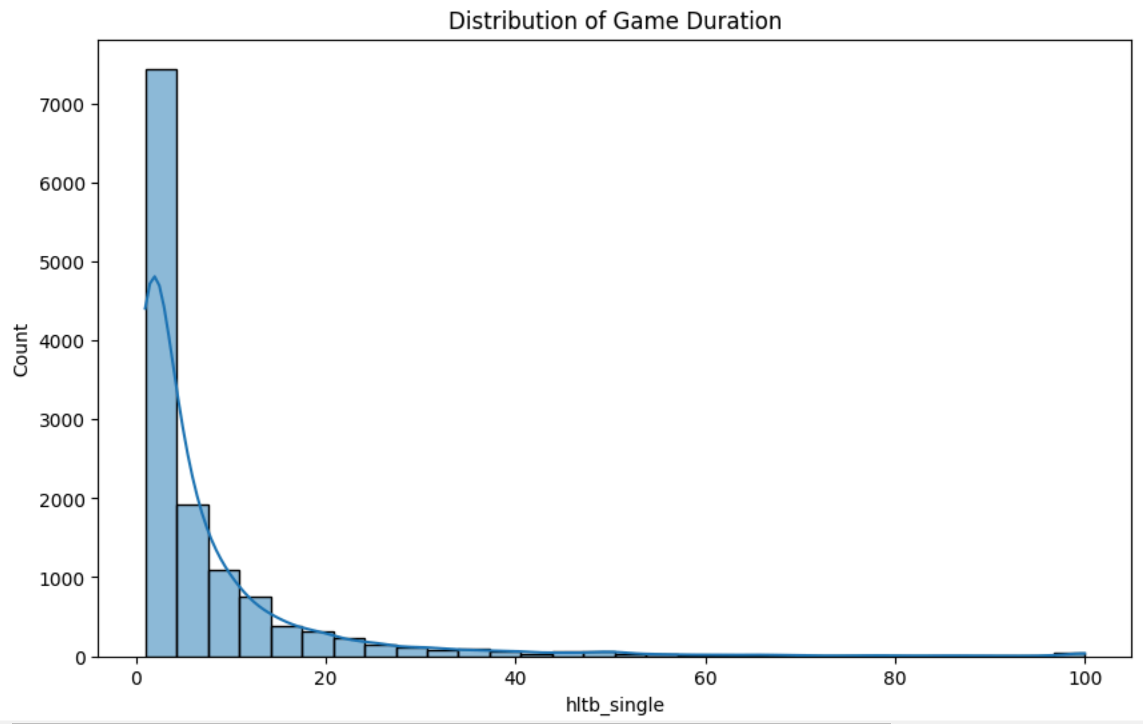
There are video games that do not have a declared ending, let's find out how they are contemplated in the dataset.

```
df.loc[df['hltb_single'].isna()]
print(len(df.loc[df['hltb_single'].isna()]), 'endless games')
```

```
47980 endless games
```

there are 47980 infinite games, which means that only about 20% of the games on the list have endings.

```
chart_univariate('hltb_single',30,'Duration')
```



Observations:

- The playtime for single-player games ranges from 1 to 100 hours. The median playtime is 3 hours, with the 75th percentile at 8 hours. This suggests that most single-player games can be completed relatively quickly, with a few outliers requiring significantly more time.
- The average playtime is 7.63 hours, with a standard deviation of 11.94 hours. This indicates that while most games have moderate playtimes, there are some games that are significantly longer.

Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

It is not surprising the gigantic gap between the economic or directly free games versus the games above 30 dollars, one of the revolutions that brought the easy access to distribution of games on PC on Steam was to open the doors to indie development which in turn by quantity and production values would result in a greater number of cheaper games.

At the same time it is interesting to know that this huge amount of free games is not because they are non-commercial works, but because they monetize their product by other means, usually with in-game microtransactions.

It is interesting to see that in 2022 the publication of games had a slight decline due to the immediate problems to COVID, then in a very short time recovered the trend, to suffer again the post-COVID crisis that reaches our days.

To work with the min_owners and max_owners variables, it will be interesting to make a bivariate exploration, this also applies to the positive and negative reviews.

Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

There were no major transformations outside of a convenient adjustment using datetime in order to display the release date plot.

The most controversial criterion is the duration of the games, since only 12972 out of 60952 games have a true duration. And I didn't decide to clean the NaN because those kind of games are important and their characteristics can be extracted from other places, specifically in the tags and categories column.

Exploring this data there is no doubt that the most effective section will be the multivariate exploration.

Bivariate Exploration

min_owners vs. **max_owners**:

Evaluating only one of the two variables is very incomplete, as the ranges are too wide for games with less attention. There are games that haven't sold zero copies but are certainly nowhere near 20,000.

Step 1: Plotting the Data

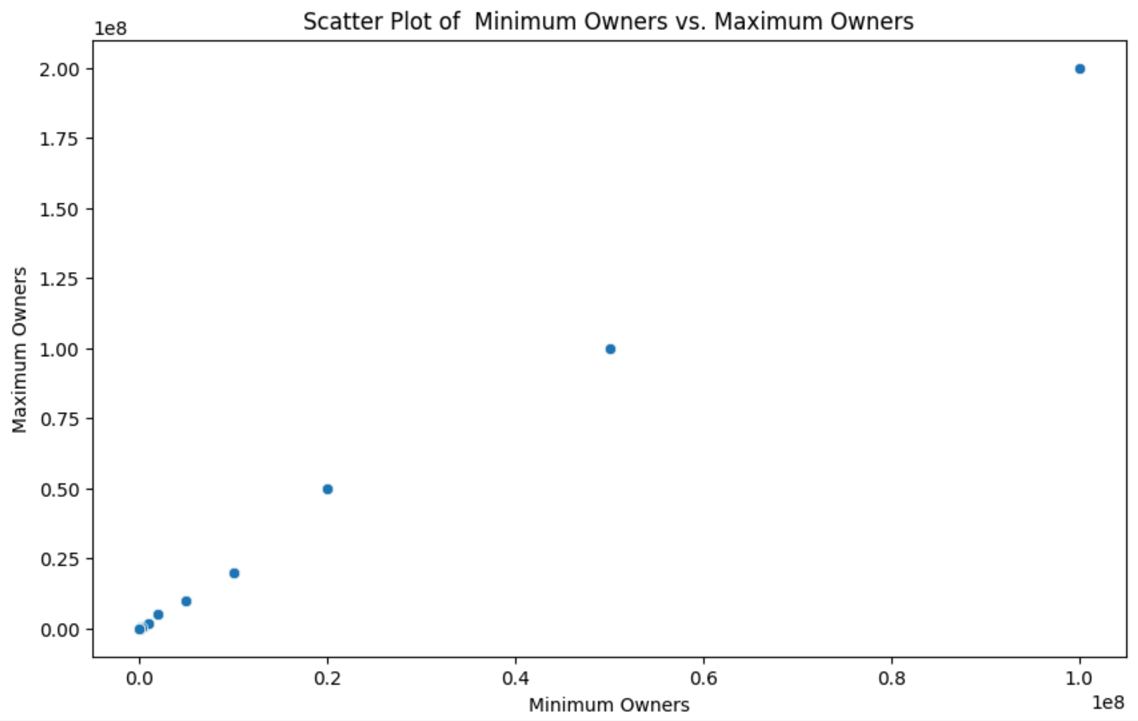
We'll use a scatter plot with both axes in logarithmic scale to handle the wide range of values effectively.

Step 2: Adjusting the Data

Given the wide range of values, we should use a variable that is adapted to be deterministic with the number of sales or acquisitions to better visualize the distribution and the relationship between min_owners and max_owners.

Original

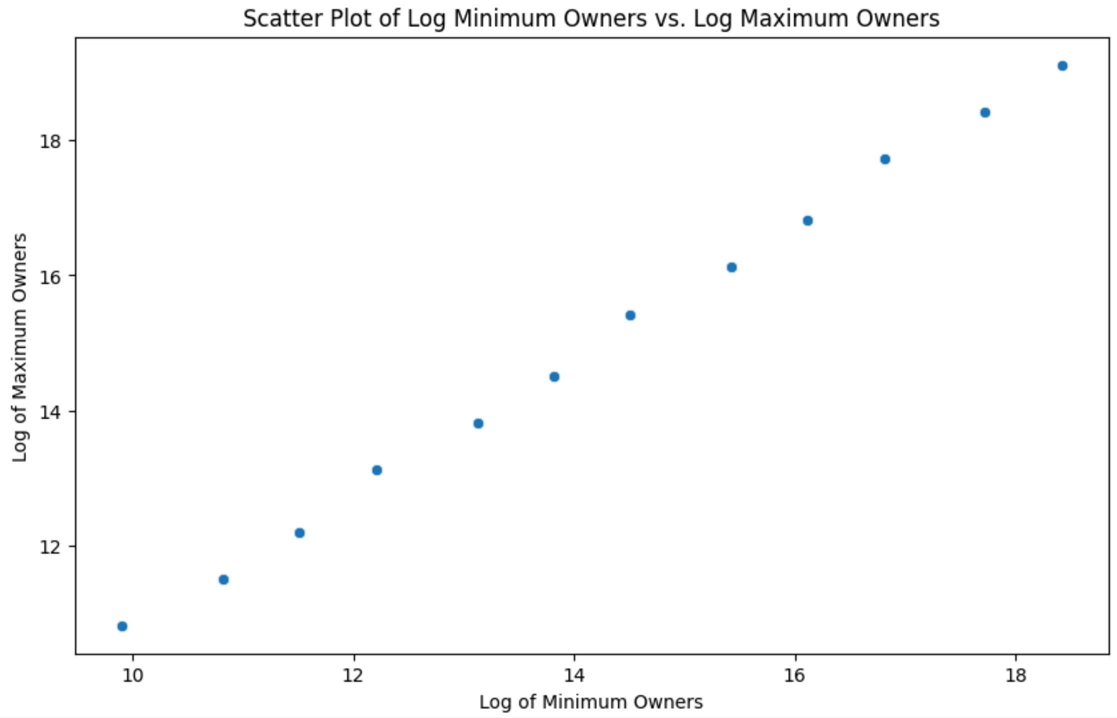
```
# Set up the plot
plt.figure(figsize=(10, 6))
sb.scatterplot(data=df, x="min_owners", y="max_owners")

# Add labels and title
plt.xlabel('Minimum Owners')
plt.ylabel('Maximum Owners')
plt.title('Scatter Plot of  Minimum Owners vs. Maximum Owners')

plt.show()
```



Logarithm

```
# Apply logarithm to 'min_owners' and 'max_owners' for plotting
df['log_min_owners'] = np.log(df['min_owners'])
df['log_max_owners'] = np.log(df['max_owners'])

# Set up the plot
plt.figure(figsize=(10, 6))
sb.scatterplot(data=df, x="log_min_owners", y="log_max_owners")

# Add labels and title
plt.xlabel('Log of Minimum Owners')
plt.ylabel('Log of Maximum Owners')
plt.title('Scatter Plot of Log Minimum Owners vs. Log Maximum Owners')

plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:396: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



Correlation: There is a positive correlation between min_owners and max_owners. Games with higher minimum ownership generally also have higher maximum ownership.

Cluster of Low-Owner Games: A significant cluster of points lies in the lower range, suggesting many games have relatively low ownership. This is likely due to the minimum ownership values of 20,000 affecting many games.

Outliers: There are outliers with extremely high ownership, indicating some games are exceptionally popular.

```
df.sort_values(by=['max_owners']).head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/nanops.py:1010: RuntimeWarning: invalid value encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single	categories	tags	lo
0	Train Bandit	2017-10-12	0.99	53	5	655370	0	20000	NaN	[Single-player, Steam Achievements, Full contr...	[Indie, Action, Pixel Graphics, 2D, Retro, Arc...	
38199	Tyr: Chains of Valhalla	2018-05-08	10.99	19	11	609720	0	20000	1.0	[Single-player, Steam Achievements, Full contr...	[Action, Adventure, Indie, Platformer, 2D, Ani...	
										[Single-player,		

```
df.sort_values(by=['max_owners']).tail()
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single	categories	t
50144	Path of Exile	2013-10-23	0.00	167454	16653	238960	20000000	50000000	25.0	[Single-player, Multi-player, MMO, PvP, Online...	[Fr A RPG, I and S RI
43855	Halo Infinite	2021-11-15	0.00	106133	38163	1240440	20000000	50000000	11.0	[Multi-player, PvP, Online PvP, LAN PvP, Cross...	[Fr Play, I Multipl Ac Sh
										[illegible]	[Surv

The value 1 is equal to 100000000 of acquisitions.

Even applying logarithmic scale it is difficult to perceive the least purchased sets, which in turn result in the highest quantity.

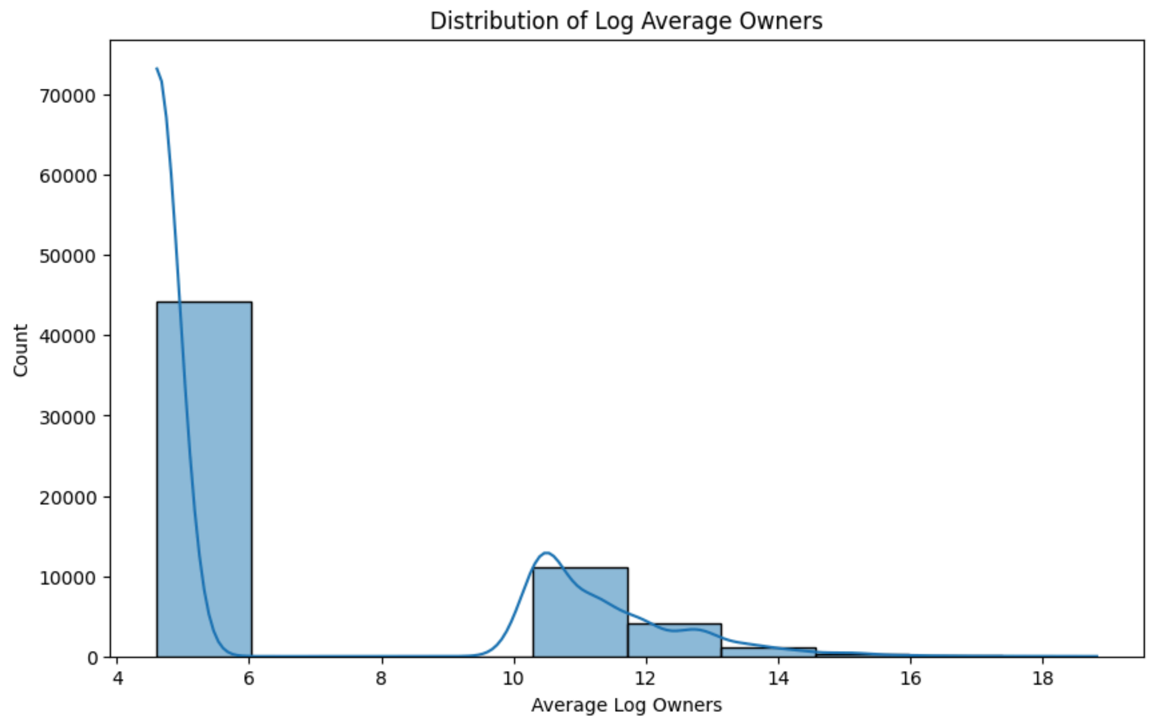
Using this graph we can have a more or less accurate intuition to be more deterministic and unify in a single variable.

```
# Create the new column based on the average
df['avg_owners'] = df[['min_owners', 'max_owners']].mean(axis=1)

# Modify the new column based on the conditions
df['avg_owners'] = df['avg_owners'].replace(10000, 100) #We define 100 to try to be fairer to those who abound but do not sell v

df
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/nanops.py:1010: RuntimeWarning: invalid value encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
```

	name	release_date	price	positive	negative	app_id	min_owners	max_owners	hltb_single	categories	tags
0	Train Bandit	2017-10-12	0.99	53	5	655370	0	20000	NaN	[Single-player, Steam Achievements, Full contr...	[Indie Action, Pixe Graphics 2D, Retro Arc..
1	Henosis™	2020-07-23	5.99	3	0	1355720	0	20000	NaN	[Single-player, Full controller support]	[2D Platformer Atmospheric Surreal Mystery,..
2	Two Weeks in Painland	2020-02-03	0.00	50	8	1139950	0	20000	NaN	[Single-player, Steam Achievements]	[Indie Adventure Nudity Violent Sexua Con..
3	Wartune Reborn	2021-02-26	0.00	87	49	1469160	50000	100000	NaN	[Single-player, Multi-player, MMO, PvP, Online...	[Turn-Based Combat Massively Multiplayer Mul..
4	TD Worlds	2022-01-09	10.99	21	7	1659180	0	20000	NaN	[Single-player, Steam Achievements, Steam Cloud]	[Towe Defense Rogue-lite RTS, Replay Value,..
...	...	...	...	...	...	...	...	...	...	...	..

```
# Plotting the avg_owners column
plt.figure(figsize=(10, 6))
sb.histplot(np.log(df['avg_owners']), bins=10, kde=True)

# Add labels and title
plt.xlabel('Average Log Owners')
plt.ylabel('Count')
plt.title('Distribution of Log Average Owners')
plt.show()
```

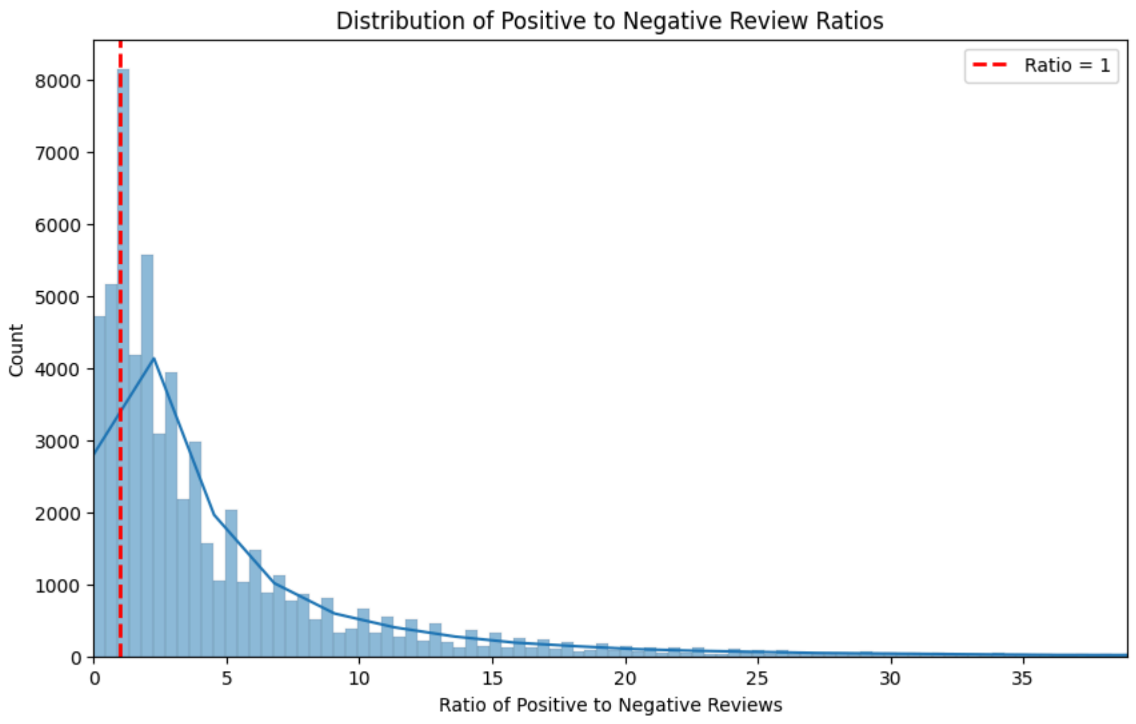


Ratio of **positive** to **negative**

Step 1: Calculating the Ratio We'll create a new column in the DataFrame to store the ratio of positive to negative reviews. To avoid division by zero, we can add a small constant to the denominator.

Step 2: Plotting the Ratio We'll use a histogram to visualize the distribution of this ratio.

```
# Calculate the ratio of positive to negative reviews
df['review_ratio'] = df['positive'] / (df['negative'] + 1)

# Set up the plot
plt.figure(figsize=(10, 6))
sb.histplot(df['review_ratio'], bins=1000, kde=True)

# Add labels and title
plt.xlabel('Ratio of Positive to Negative Reviews')
plt.ylabel('Count')
plt.title('Distribution of Positive to Negative Review Ratios')

plt.xlim(0, df['review_ratio'].quantile(0.99))  # Limiting x-axis to 99th percentile to handle outliers

# Add a vertical line at x=1
plt.axvline(x=1, color='r', linestyle='--', linewidth=2, label='Ratio = 1')
plt.legend()


plt.show()
```



Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

This in summary is the great relationship I find strongest.

Compare sales with critical acclaim and game length: Look particularly at the sales performance of the most critically acclaimed games and their length. This can be one of the fundamental keys to understanding how the market functions and whether there really is a close relationship between quality, longevity and number of sales.

The following is my extended interpretation of the bivariate exploration

Relationship Between min_owners and max_owners:

Logarithmic Relationship: When plotting the logarithm of min_owners against max_owners, it becomes evident that there's a strong positive correlation between the two. This makes sense since games with a higher minimum number of owners also tend to have a higher maximum number of owners.

Wide Range: The values of min_owners and max_owners span several orders of magnitude, highlighting the disparity between less popular and highly popular games.

Clustering: There is a noticeable clustering of games within certain ranges, indicating that many games tend to have a similar range of ownership. For instance, games with min_owners in the lower range (e.g., 5,000 to 20,000) tend to cluster together.

Relationship Between Ratio of Positive to Negative Reviews:

Skewed Distribution: The ratio of positive to negative reviews is heavily skewed towards the lower end, with a long tail extending towards higher ratios. This suggests that while many games have a moderate number of positive reviews compared to negative ones, a smaller subset of games receives overwhelmingly positive feedback.

Critical Threshold: The vertical line at a ratio of 1 on the x-axis is crucial. Games with a ratio above 1 have more positive reviews than negative ones, while those below 1 have more negative reviews. Most games fall above this line, indicating generally positive reception.

Outliers: The distribution has a few significant outliers with extremely high positive-to-negative review ratios. These outliers likely represent highly acclaimed games with a substantial number of positive reviews relative to negative ones.

Observations on Combined Analysis:

Popularity and Reviews: There seems to be a relationship between the number of owners (both minimum and maximum) and the review ratio. Games with a higher number of owners tend to have higher positive-to-negative review ratios, suggesting that popular games generally receive better reviews.

Review Ratio's Impact: The review ratio can serve as an indicator of a game's success and reception in the market. Games with higher ratios are more likely to attract more players, leading to higher ownership numbers.

In summary, the investigation reveals that games with higher ownership numbers tend to receive better reviews, and the positive-to-negative review ratio is a significant indicator of a game's reception and success.

Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Here are a number of ideas that I found interesting.

Seasonal trends: It would be interesting to see if there are specific months or quarters within each year where there are more game releases, you could split this between big price and modest prices to understand both sectors of the industry.

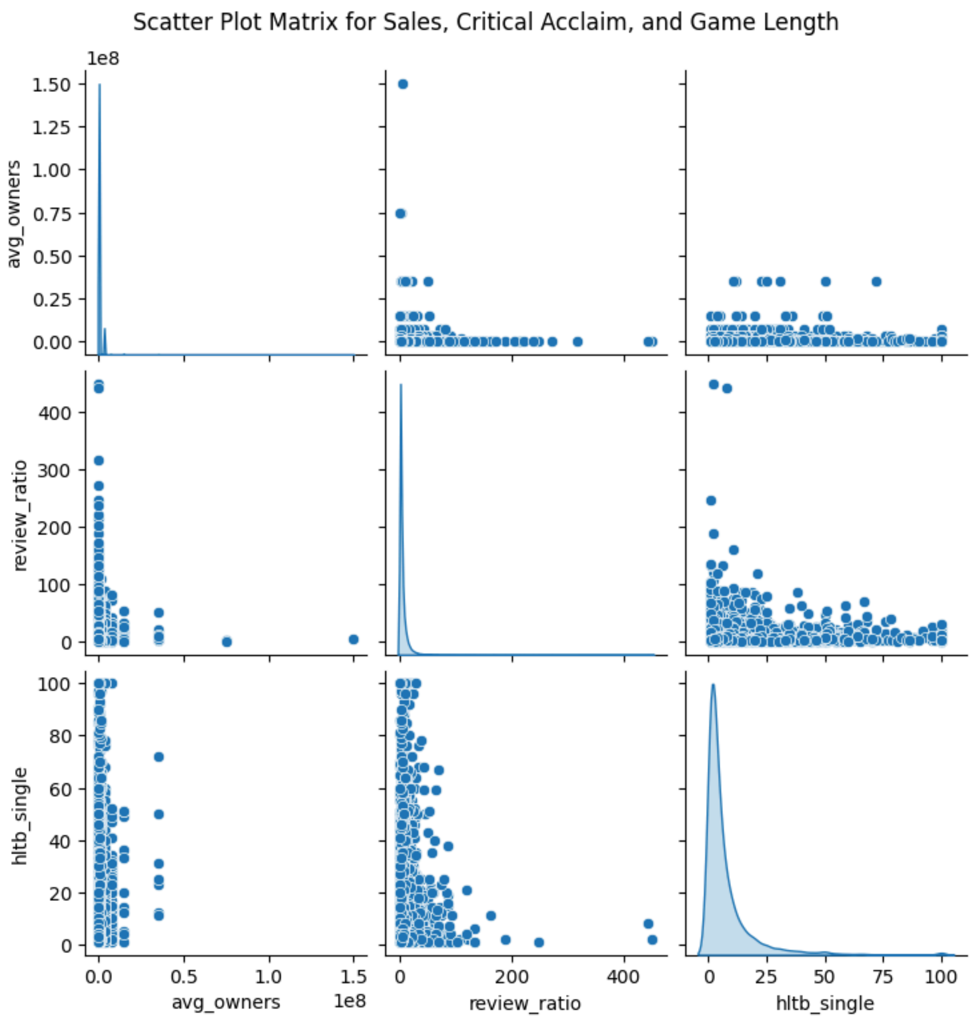
Genre analysis: investigate whether particular game genres have contributed more to the increase in game releases over time.

Impact of major events: Further analysis of how major world events (economic crises, pandemics, etc.) have influenced game releases.

Multivariate Exploration

1. Explore the relationship between sales (using avg_owners as a proxy), critical acclaim (using review_ratio as a proxy), and game length (hltb_single)

```
# Select the relevant columns for multivariate analysis
df_multivariate = df[['avg_owners', 'review_ratio', 'hltb_single']]

# Pairplot to see the relationship between avg_owners, review_ratio, and hltb_single
sb.pairplot(df_multivariate, diag_kind='kde')
plt.suptitle('Scatter Plot Matrix for Sales, Critical Acclaim, and Game Length', y=1.02)
plt.show()
```



Observations:

Sales (avg_owners) vs. Critical Acclaim (review_ratio):

There is not a positive correlation between the number of owners and the review ratio. Critically acclaimed games tend not to be more popular and therefore reserve a more moderate place in the critics. Contrary to the overflow that can be found with lesser-owned games, which seem to have a more heterogeneous critical distribution.

Outliers exist where games have a high number of owners but a low review ratio, indicating that despite being popular, they might not be critically acclaimed.

Sales (avg_owners) vs. Game Length (hltb_single):

The relationship between game length and sales is complex. Some players prefer longer games for their greater value, while others may prefer shorter games for a quicker experience.

We can see that extremely long or short games have different levels of popularity, indicating different player preferences.

So it seems that the result is fairly homogeneous in terms of interest in buying products in their various lengths.

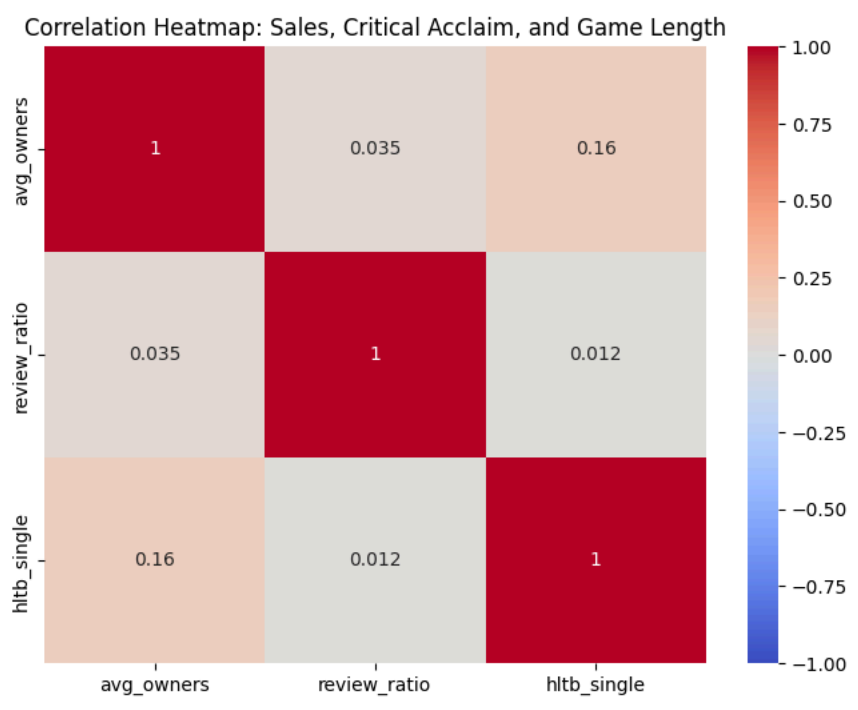
Critical Acclaim (review_ratio) vs. Game Length (hltb_single):

The relationship between game length and reviews is complex.

While it can be seen that games with the highest number of positive reviews are in the 3 to 15 minute range, games with the highest number of positive reviews are in the 3 to 15 minute range.

This is to be expected knowing that this section is made up of single-player games and a shorter, juicier experience may be more satisfying for the user.

```
# Calculate the correlation matrix
corr_matrix = df_multivariate.corr()

# Plotting the heatmap
plt.figure(figsize=(8, 6))
sb.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap: Sales, Critical Acclaim, and Game Length')
plt.show()
```



Observations:

Sales and Critical Acclaim:

Out of logic, there is no strong relationship between sales and prestige. It can be understood that in the most popular games, communities are built more deeply rooted and therefore become constant critics of the work.

Sales and Game Length:

The strongest of the relationships. We could be quite sure that the games of less than 20 hours are the ones with the best conversion sale and duration.

Critical Acclaim and Game Length:

The least strong relationship. This implies that along with the idea presented in 'Sales and Game Length' that there is a special predisposition of players for games with a duration of less than 20 hours without being very relevant the quality of the game itself.

2. Explore the seasonal trends in game releases, particularly focusing on whether there are specific months or quarters with more game releases, and to compare these trends between high-priced and modestly priced game

```
df['price'].describe()
```

	price
count	60952.000000
mean	7.819159
std	9.756732
min	0.000000
25%	1.990000
50%	4.990000
75%	9.990000
max	299.900000

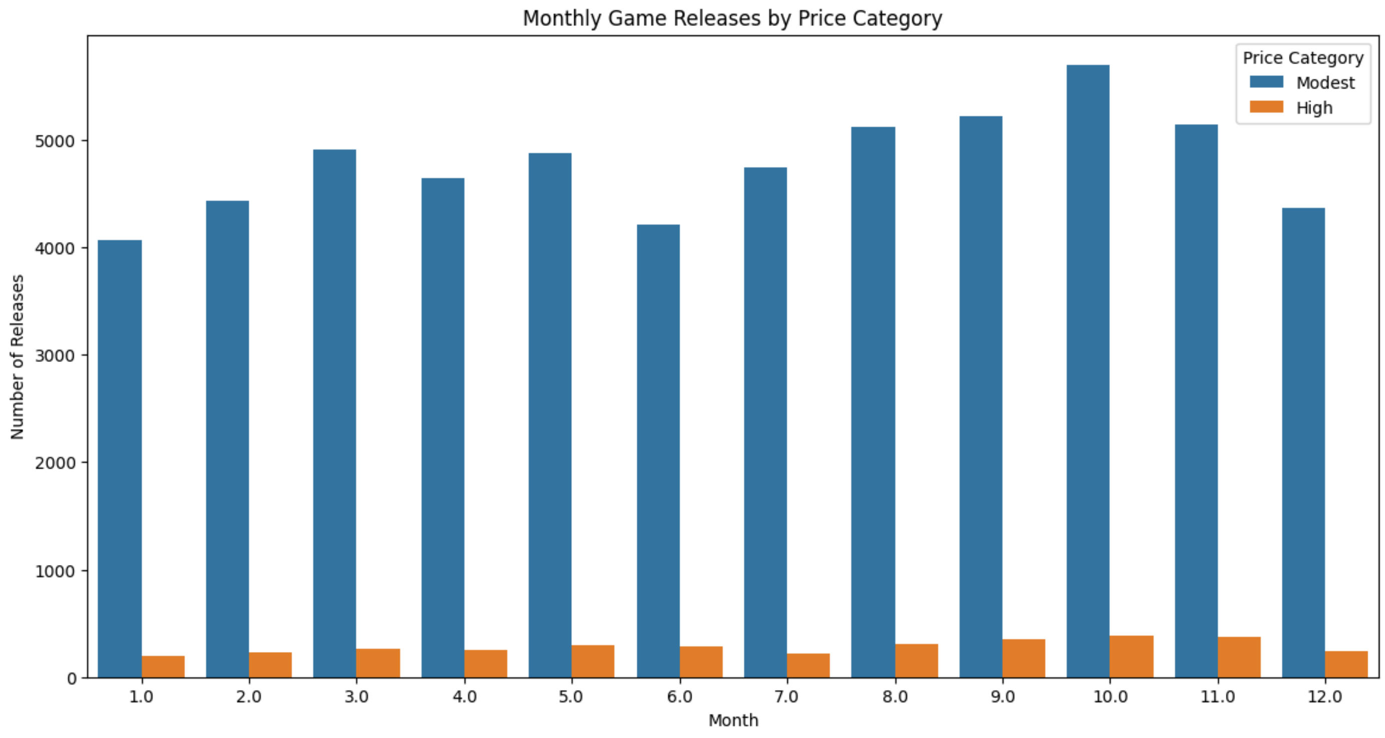
dtype: float64

In terms of quantity it is easy to observe a total hegemony of games under $10. In order to enter a middle ground we will use as a threshold the 20 usd to divide the independent industry and the industry of the big productions.

```
# Extract month and quarter
df['release_month'] = df['release_date'].dt.month
df['release_quarter'] = df['release_date'].dt.quarter

# Classify games as high-priced or modestly priced
price_threshold = 20 # Using the price above 75% of all prices
df['price_category'] = df['price'].apply(lambda x: 'High' if x > price_threshold else 'Modest')

print('Treshold: ',price_threshold)
```

```
Treshold:  20
```

```
plt.figure(figsize=(14, 7))

# Plotting monthly release trends
sb.countplot(data=df, x='release_month', hue='price_category')
plt.xlabel('Month')
plt.ylabel('Number of Releases')
plt.title('Monthly Game Releases by Price Category')
plt.legend(title='Price Category')
plt.show()
```

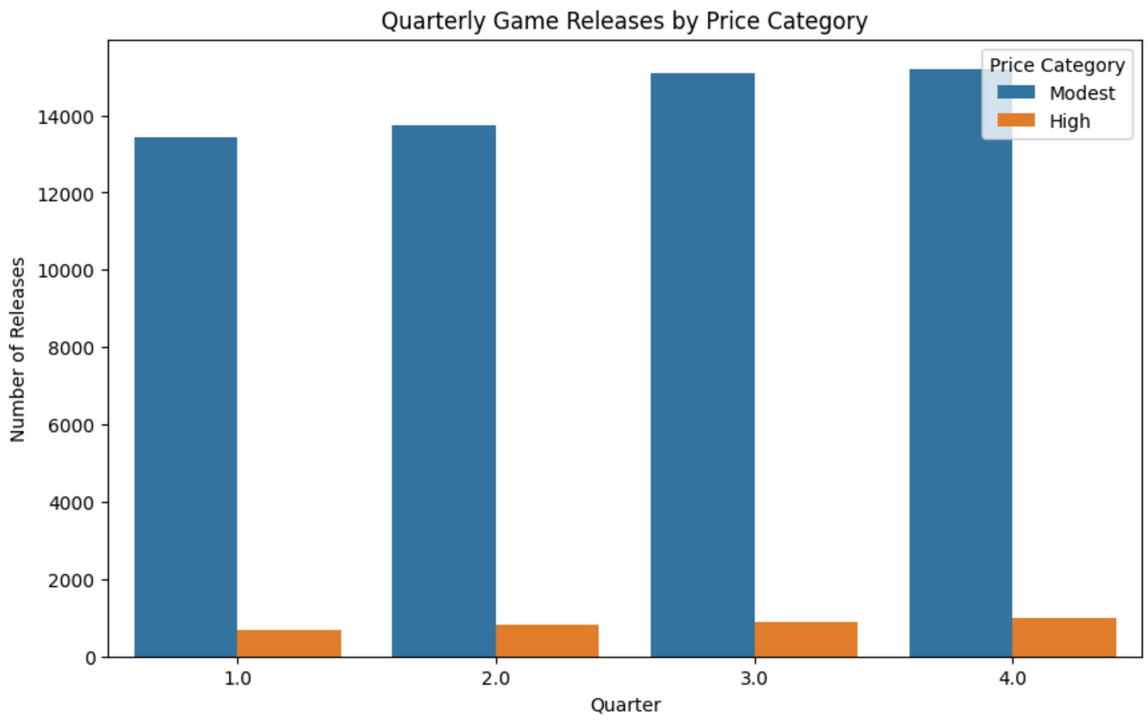


The difference in the scope and quantitative dominance of the independent industry is clear.

Even knowing that these videogames deal with particular market niches, their behavior is very similar in both cases. The middle of the year being scarcer and reaching its peak at the end of the year.

The reality is that throughout the year the launching of videogames is quite homogeneous.

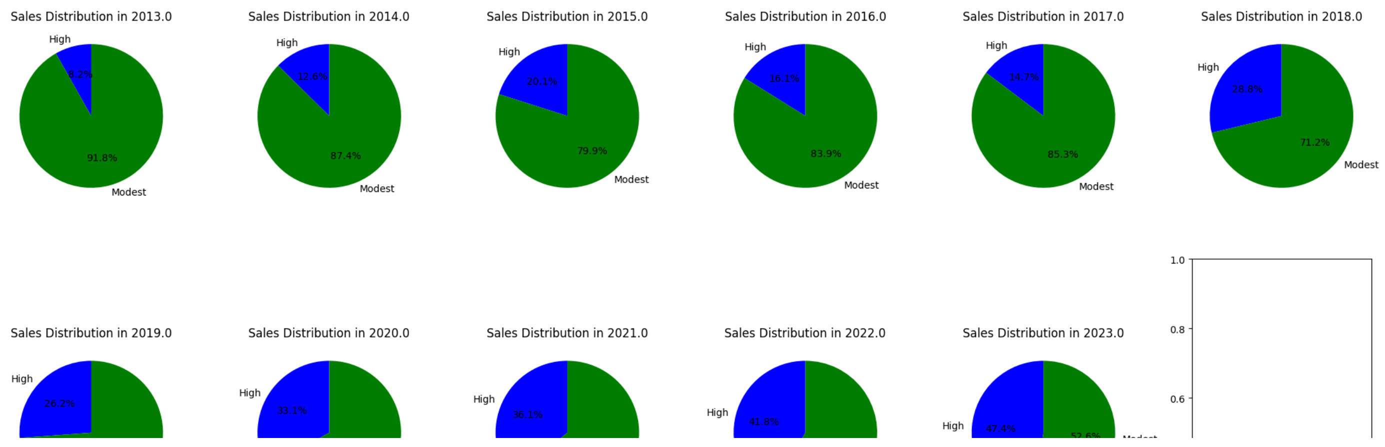
```
plt.figure(figsize=(10, 6))

# Plotting quarterly release trends
sb.countplot(data=df, x='release_quarter', hue='price_category')
plt.xlabel('Quarter')
plt.ylabel('Number of Releases')
plt.title('Quarterly Game Releases by Price Category')
plt.legend(title='Price Category')
plt.show()
```



We confirm that the second half of the year is the most competitive.

3. Analysis of Sales Distribution Between High-Priced and Modestly Priced Games

```
# Extract the year from the release date
df['release_year'] = df['release_date'].dt.year

# Aggregate total average owners per year for each price category
annual_sales = df.groupby(['release_year', 'price_category'])['avg_owners'].sum().reset_index()


# List of unique years
years = annual_sales['release_year'].unique()

# Set up the plot
fig, axes = plt.subplots(nrows=2, ncols=len(years)//2 + len(years)%2, figsize=(20, 10))
axes = axes.flatten()

# Plot pie charts
for i, year in enumerate(years):
    data = annual_sales[annual_sales['release_year'] == year]
    axes[i].pie(data['avg_owners'], labels=data['price_category'], autopct='%1.1f%%', startangle=90, colors=['blue', 'green'])
    axes[i].set_title(f'Sales Distribution in {year}')

plt.tight_layout()
plt.show()
```


Sales Distribution in 2013.0
High
8.2%
91.8%
Modest
Sales Distribution in 2014.0
High
12.6%
87.4%
Modest
Sales Distribution in 2015.0
High
20.1%
79.9%
Modest
Sales Distribution in 2016.0
High
16.1%
83.9%
Modest
Sales Distribution in 2017.0
High
14.7%
85.3%
Modest
Sales Distribution in 2018.0
High
28.8%
71.2%
Modest
Sales Distribution in 2019.0
High
26.2%
Sales Distribution in 2020.0
High
33.1%
Sales Distribution in 2021.0
High
36.1%
Sales Distribution in 2022.0
High
41.6%
Sales Distribution in 2023.0
High
47.4%
52.6%
1.0
0.8
0.6