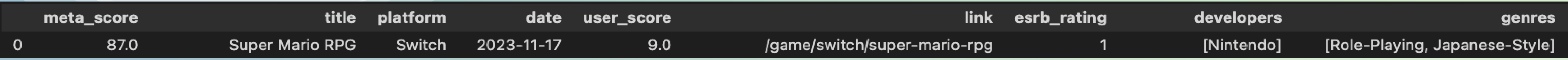
# Data Science Final

By: Ethan

# Nintendo Games

- The dataset chosen contained most the Nintendo games ever made. Dating back as far as 1996. An example of a row in this dataset would look like this. This data was found on kaggle here

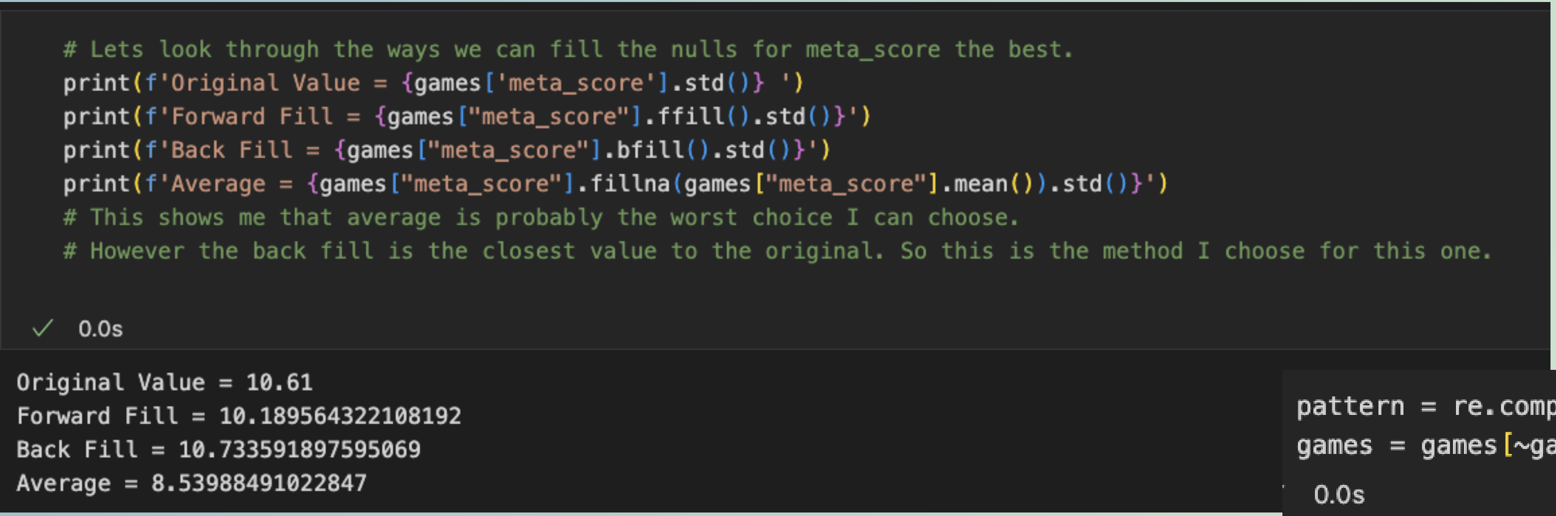


| | meta_score | title | platform | date | user_score | link | esrb_rating | developers | genres |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 87.0 | Super Mario RPG | Switch | 2023-11-17 | 9.0 | /game/switch/super-mario-rpg | 1 | [Nintendo] | [Role-Playing, Japanese-Style] |

- Meta Score
  - The rating a popular game critic site, Metacritic, gives the game. 0-100 range
- Platform
  - The original game system that this game was released on.
- User Score
  - The average score a user gives the game on Metacritic. 0-10 range
- ESRB Rating
  - These are the ratings that ESRB gives the game. These are categorical and have an order like this; RP -> E -> E10+ -> T -> M
- Developers
  - The companies that help make the game. This comes in a list as occasionally developers' team up to make a single game.
- Link
  - I don't use it for anything data related, but it is how you get to the game page on the official Metacritic website

# Challenges – Missing Data

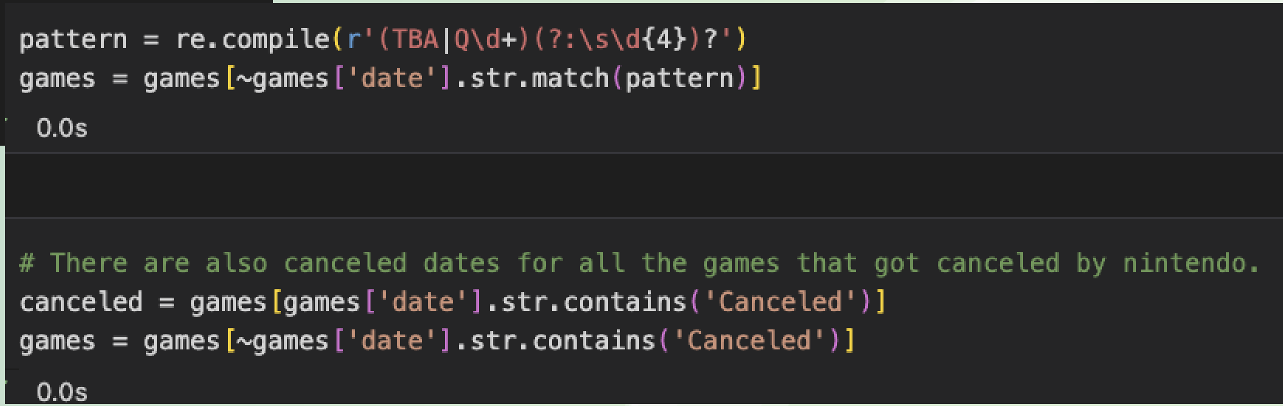
- Originally when loading the dataset, I knew there would be some missing data. The missing data was in a couple different columns. When filling the null values, I wanted the standard deviation to change the least from the original value. So, I tested all the ways we learned to fill data and chose the one that changed the sd least. This was used multiple times of the varying columns.
- Looking through the other columns and their unique values I found that the ESRB_Rating had a few ways to say that the data was null. Be it "Cancelled" or "Q3 2025". Which is just data that I didn't care about. And to me, that is just RP (rating pending). So I replaced all of those "missing values" with "RP".



```
# Lets look through the ways we can fill the nulls for meta_score the best.
print(f'Original Value = {games['meta_score'].std()} ')
print(f'Forward Fill = {games["meta_score"].ffill().std()}')
print(f'Back Fill = {games["meta_score"].bfill().std()}')
print(f'Average = {games["meta_score"].fillna(games["meta_score"].mean()).std()}')
# This shows me that average is probably the worst choice I can choose.
# However the back fill is the closest value to the original. So this is the method I choose for this one.
```

✓ 0.0s

```
Original Value = 10.61
Forward Fill = 10.189564322108192
Back Fill = 10.733591897595069
Average = 8.53988491022847
```



```
pattern = re.compile(r'(TBA|Q\d+)(?:\s\d{4})?')
games = games[~games['date'].str.match(pattern)]
```

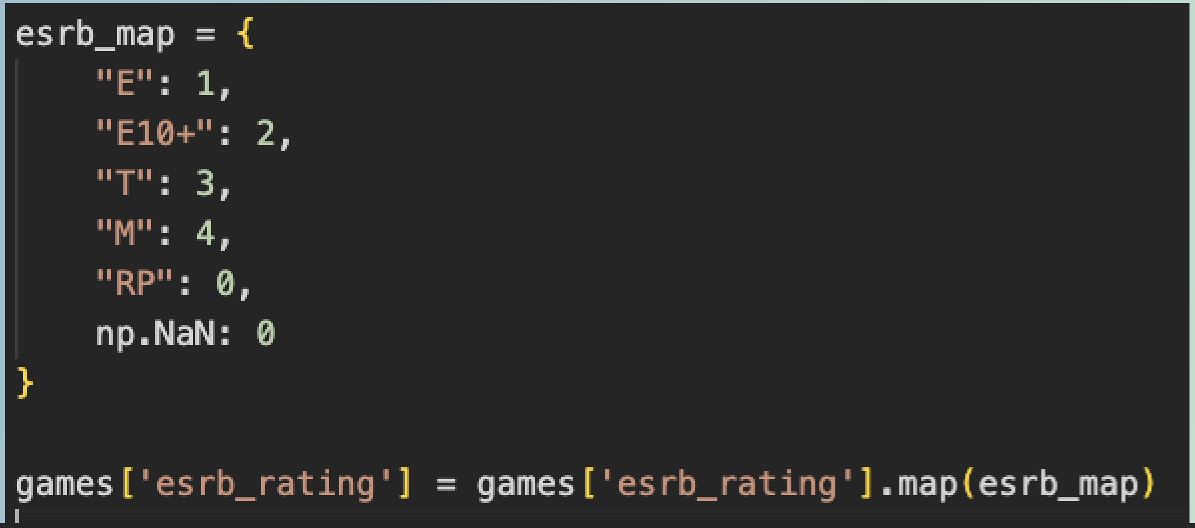
0.0s

```
# There are also canceled dates for all the games that got canceled by nintendo.
canceled = games[games['date'].str.contains('Canceled')]
games = games[~games['date'].str.contains('Canceled')]
```

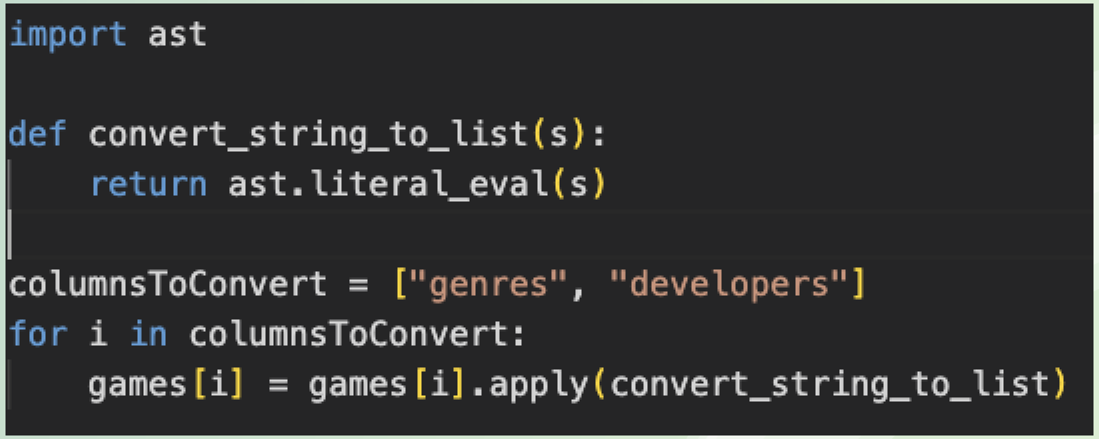
0.0s

## Challenge: Wrangling Data

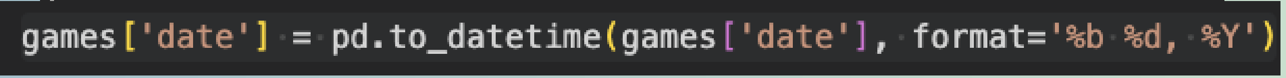
- When looking through the data I noticed there were multiple columns with lists, and I thought this would be a great time to practice wrangling data. Found an interesting package that turned string literals of lists into lists and used that to solve my problem of having the lists be strings.
- Another thing that came up was ESRB_Rating. I wanted to turn this categorical variable into numbers so that working with it would become surprisingly easy. I just decided to map a list of values to the column and reassign it.
- The last thing that was wrangled was the date. I thought it would be tough cause the data started with values like 'Nov 13, 2022'. That string of the month scared me.



```python
esrb_map = {
    "E": 1,
    "E10+": 2,
    "T": 3,
    "M": 4,
    "RP": 0,
    np.NaN: 0
}

games['esrb_rating'] = games['esrb_rating'].map(esrb_map)
```



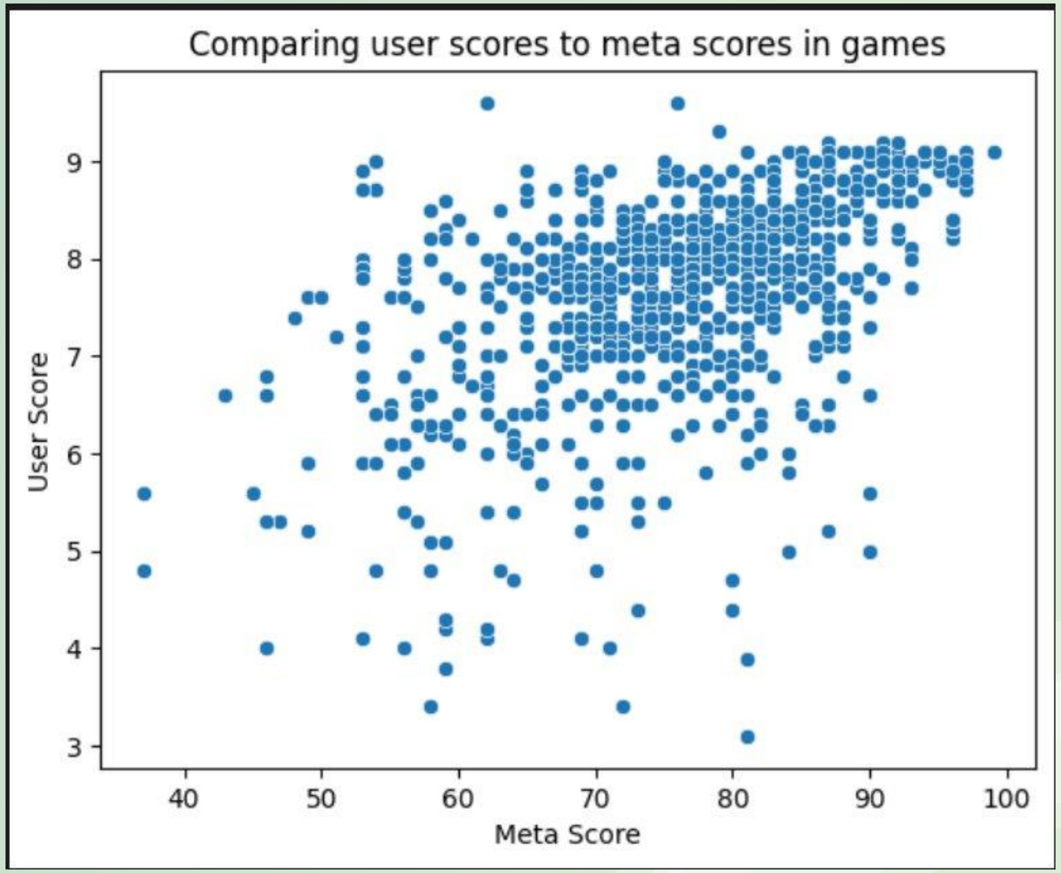
```python
import ast

def convert_string_to_list(s):
    return ast.literal_eval(s)

columnsToConvert = ["genres", "developers"]
for i in columnsToConvert:
    games[i] = games[i].apply(convert_string_to_list)
```



```python
games['date'] = pd.to_datetime(games['date'], format='%b %d, %Y')
```

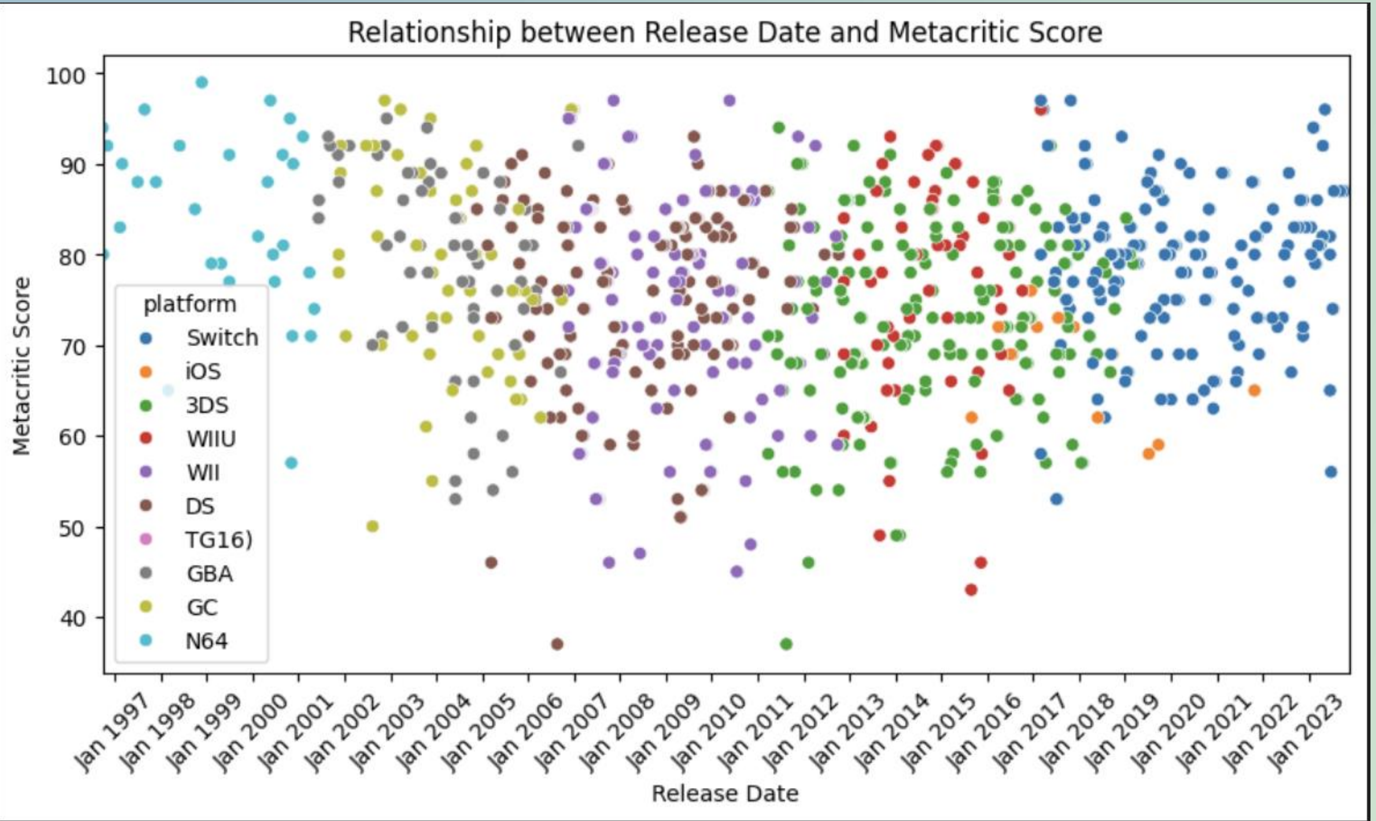
# Figures: Part 1

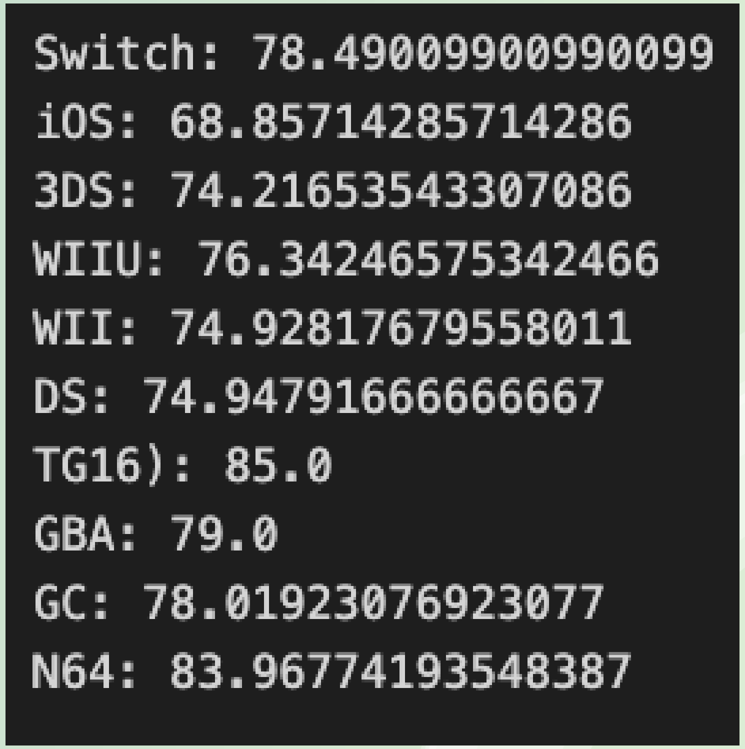
- In the first graph I decided to graph what I felt had the largest chance of being correlated. User Scores compared to Metacritic Scores. The graph to me seems to have a semi-decent correlation. Nothing crazy. But what we can see is that for the most part as Metacritic scores go up, so will user scores. But this is not always the case. There are still plenty of disagreements between the users and Metacritic. Resulting in what we with the correlation semi-existing, but it isn't a guarantee



# Figures: Part 2

- A question I had coming into this dataset was if the scores Metacritic were giving Nintendo over time were increasing. So, I compared the date the game released to the meta score. Along with setting a hue to the platform the game released on to make describing dates easier. While looking at the graph, I feel I see that the ds and the 3ds had some of the worst rated games. It may be possible that the average may be higher these days. Let's looks at the averages of each platform. It seems the Nintendo 64 had the highest average for ratings. So, it may be possible that Nintendo games have been going down.





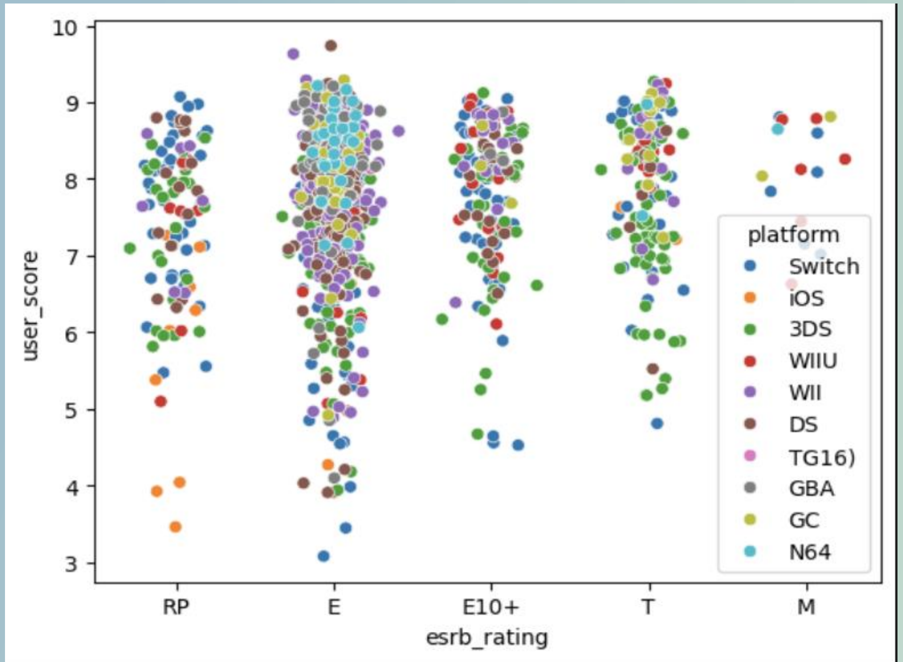
```
Switch: 78.49009900990099
iOS: 68.85714285714286
3DS: 74.21653543307086
WIIU: 76.34246575342466
WII: 74.92817679558011
DS: 74.94791666666667
TG16): 85.0
GBA: 79.0
GC: 78.01923076923077
N64: 83.96774193548387
```

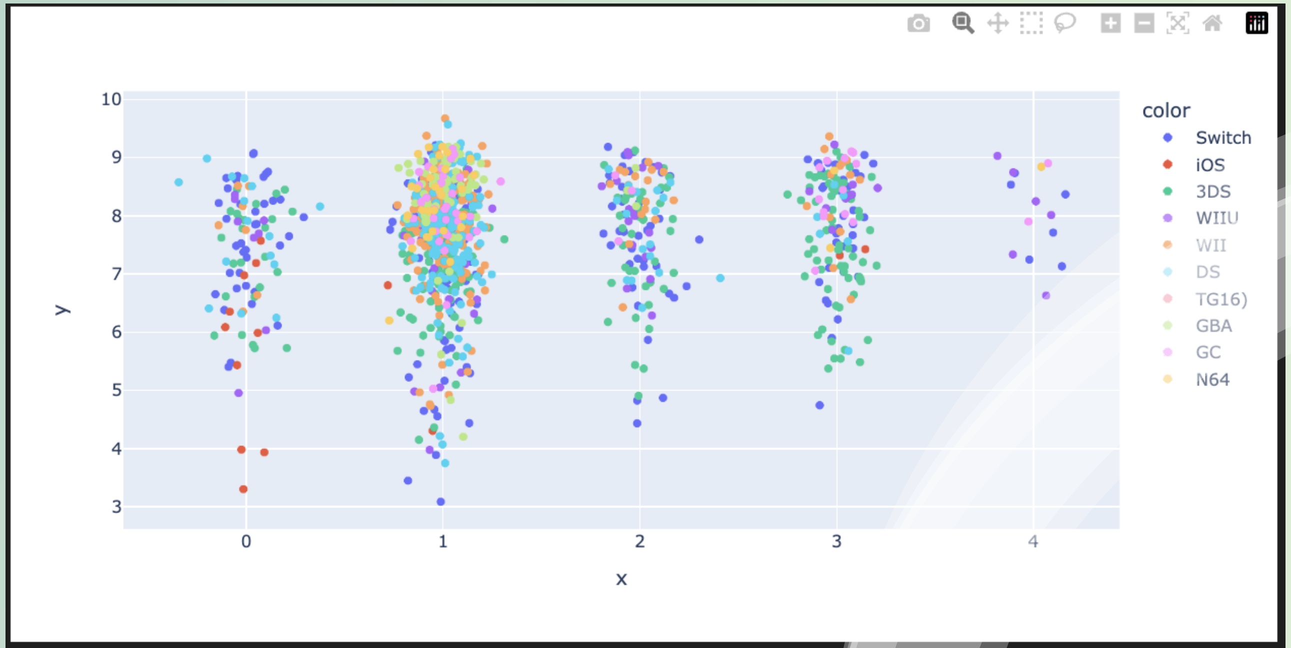
## What Was That?

- There was one platform in the last page that had a higher average than all the others. But I just ignored it and said the Nintendo 64 was the highest. This was mainly because there is only a SINGLE game released on this TG16. A game called Ordyne. I had never heard of this console. So, I had to google what it was. It was apparently a console called Turbo-Grafx16. I believe this showed up because Nintendo bought the rights to this Ordyne game. This was quite an interesting search to figure out what the heck was going on. Just thought you would love to hear about this question that came up as I was looking through the other questions.

# Figures: Part 3 (Interactive Graphs)

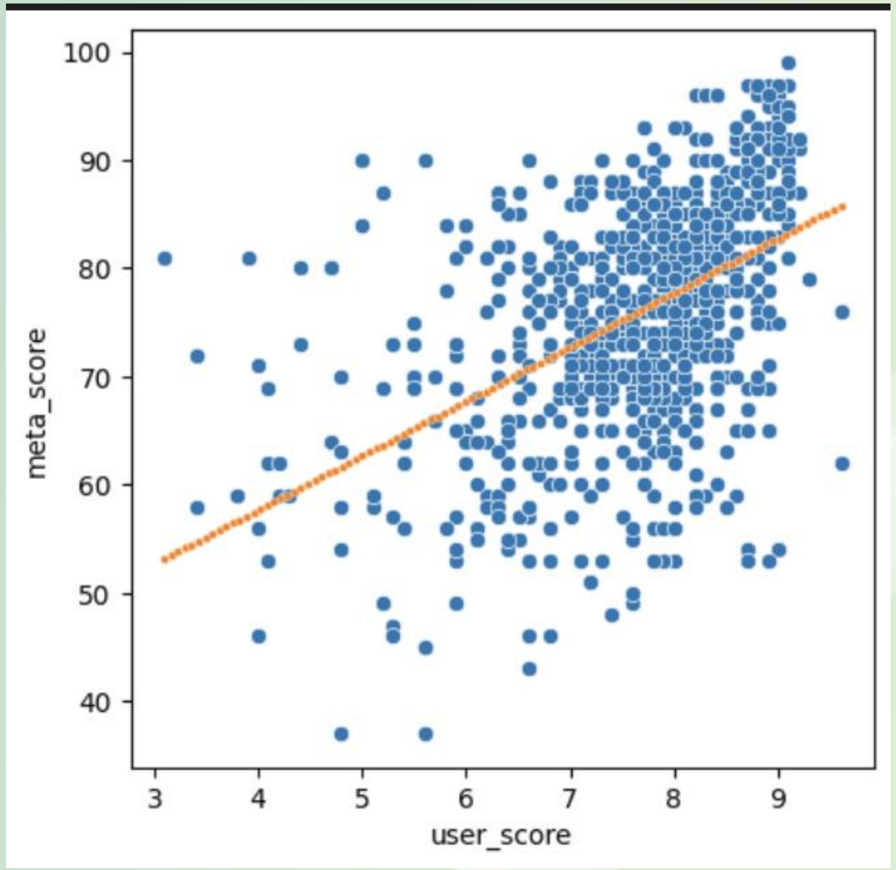
For the last graph I wanted to see if there was any difference between the ESRB rating and the User Scores. I wanted to see if users liked certain types of games more than others. Graphing this was a little difficult, so I added a jitter to the y axis. The groups are still visible, so I felt like this was okay. It is also fun to see how many E rated games they have over M rated games.





## Analysis

- I decided that doing a linear regression on the meta score and the user score would be the most fun to display. This correlation was .43 so there wasn't too strong of a correlation, but it does exist. Lets look at how a regression line looks. The line seems average. It could be a little bit better if the correlation was better. But this is the best we can do. The testing metrics resulted in this.
- MAE
    - 7.32
- SSE
    - 94495.5
- MSE
    - 88.8
- RMSE
    - 9.4
- RMSLE
    - 7.7



## Conclusion

- Finding correlations for this dataset doesn't feel like the true purpose of this dataset. I found the most fun was looking though all the games that were unique. The ones with the highest user score because it got artificially boosted. The games that were planned to be released but eventually got canceled. These parts of the dataset were the most fun. There are still small correlations to be found. I am sure it could be fun to try to figure out the developers based on the Metacritic score or the user score. But because the developers is a list, I am unsure how to test something like that. That is where I would like to continue off if I ever came back to this dataset. There is still plenty beauty to discover in this dataset.