Project 4: Visualization with Matplotlib

John Wesley Mathis

Dr. Anthony Choi

June 14, 2024

ECE/SSE 591, Summer 2024

Table of Contents:

[Deliverable Table 5](#__RefHeading___Toc2569_331549091)

[1. Introduction 6](#__RefHeading___Toc2573_331549091)

[2. Adapting SIR Model for Visualization 7](#__RefHeading___Toc2577_331549091)

[3. Adapting Movie Analysis Model for Visualization 8](#__RefHeading___Toc2969_2801450533)

[4. Other Models 9](#__RefHeading___Toc8059_276922697)

[5. Conclusion 10](#__RefHeading___Toc2609_331549091)

[References 11](#__RefHeading___Toc3692_331549091)

Table of Figures

[Figure 1: Using Pandas objects and importing CSV files 7](#Figure!0|sequence)

[Figure 2: Collecting information about the data 7](#Figure!1|sequence)

[Figure 3: Collecting information about the dataset 8](#Figure!2|sequence)

[Figure 4: Changing the index of the dataset 9](#Figure!3|sequence)

[Figure 5: Adding a Rank column 10](#Figure!4|sequence)

[Figure 6: Moving 'Rank' column to the beginning of the dataset 11](#Figure!5|sequence)

[Figure 7: Sorting the data by IMDB rating 12](#Figure!6|sequence)

[Figure 8: Finding what data is missing 13](#Figure!7|sequence)

[Figure 9: Removing unused columns from the dataset 14](#Figure!8|sequence)

[Figure 10: Dealing with missing data 15](#Figure!9|sequence)

[Figure 11: Checking for duplicated data 16](#Figure!10|sequence)

[Figure 12: Converting 'Gross' column from String to Integer 16](#Figure!11|sequence)

[Figure 13: Cleaning up the 'Runtime' column 17](#Figure!12|sequence)

[Figure 14: Converting release year to datatime format 18](#Figure!13|sequence)

[Figure 15: Cleaning up the genre column 18](#Figure!14|sequence)

[Figure 16: Determining top movie genres 19](#Figure!15|sequence)

[Figure 17: Determining top 5 most voted movies 19](#Figure!16|sequence)

[Figure 18: Determining bottom 5 least voted movies 20](#Figure!17|sequence)

[Figure 19: Determining which movie has the longest runtime 20](#Figure!18|sequence)

[Figure 20: Determining which movie has the shortest runtime 21](#Figure!33|sequence)

[Figure 21: Using `query()` to filter data 21](#Figure!19|sequence)

[Figure 22: Using `eval()` to calculate new data 22](#Figure!20|sequence)

[Figure 23: Determining which year had the most top movies 22](#Figure!21|sequence)

[Figure 24: Determining highest grossing movie 23](#Figure!22|sequence)

[Figure 25: Determining which genre has the IMDB rating and which has the highest gross 24](#Figure!23|sequence)

[Figure 26: Discovering which directors consistently produce top movies using pivot tables 25](#Figure!24|sequence)

[Figure 27: Demonstrating hierarchical indexing 26](#Figure!25|sequence)

[Figure 28: Importing two new datasets and showing information about dataset 1 27](#Figure!26|sequence)

[Figure 29: Dataset 2 information 27](#Figure!27|sequence)

[Figure 30: Merging the two datasets on 'id' 28](#Figure!28|sequence)

[Figure 31: Removing unused columns 29](#Figure!29|sequence)

[Figure 32: Setting the index to 'genres' and 'release\_date' 30](#Figure!30|sequence)

[Figure 33: Code for vectorized string operations 30](#Figure!31|sequence)

[Figure 34: Output to Figure 32 code 31](#Figure!32|sequence)

[Figure 35: Demonstrating `concat()` 32](#Figure!34|sequence)

[Figure 36: Using `concat()` to append new data 33](#Figure!35|sequence)

# Deliverable Table

The purpose of this table is to provide a complete view of the concepts covered in chapter 4 of *"Python Data Science Handbook"* (VanderPlas, 2016) and provide a general page location for where the topic was demonstrated.

|  |  |
| --- | --- |
| Deliverables | Location |
| Simple Line Plots |  |
| Simple Scatter Plots |  |
| Visualizing Errors |  |
| Density and Contour Plots |  |
| Histograms, Binnings, and Density |  |
| Customizing Plot Legends |  |
| Customizing Colorbars |  |
| Multiple Subplots |  |
| Text and Annotation |  |
| Customizing Ticks |  |
| Customizing Matplotlib: Configurations and Stylesheets |  |
| Three-Dimensional Plotting in Matplotlib |  |
| Geographic Data with Basemap |  |
| Visualization with Seaborn |  |

Additionally, here is a link to my GitHub were the datasets and the Jupyter Notebook for the project can be downloaded: https://github.com/jwmathis/SSE591\_Project4. In order to run the file, Python and other dependencies must be installed.

# 1. Introduction

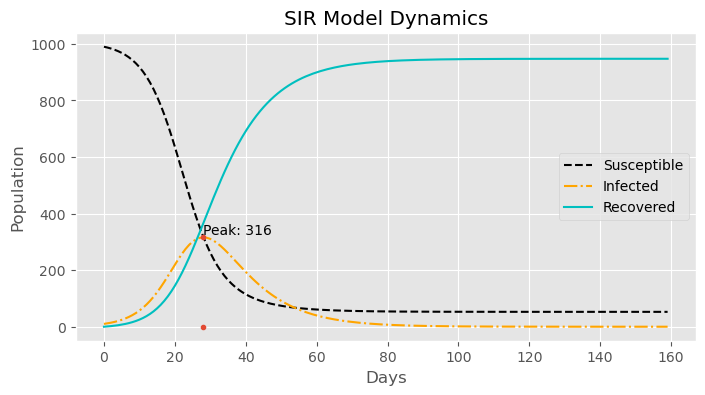
Matplotlib provides a comprhensive and flexible interface for creating static, animated, and interactive visualizations in Python. While libraries like Pandas and NumPy are essential for data manipulation and numerical computations, Matplotlib excels at presenting this data in a visual format that can uncover insights and trends. Because of its wide range of plotting functions and customization options, it makes it an invaluable tool for data scientists who aim to present their data clearly and effectively. Additionally, its integration with Pandas and NumPy allows for seamless data visualization directly from the libraries respective data structures.

This report aims to demonstrate my proficiency in Python data visualization techniques as covered in Chapter 4 of the “Python Data Science Handbook” by Jake VanderPlas (2016). This report attempts to illustrate the core concepts and functionalities of the Pandas library by implementing the concepts into practical examples. The code presented in this report was developed using Visual Studio Code with Jupyter Notebook extensions. I will provide detailed explanations, highlighting key features and operations that make Matplotlib an essential tool for data analysis.

# 2. Adapting SIR Model for Visualization

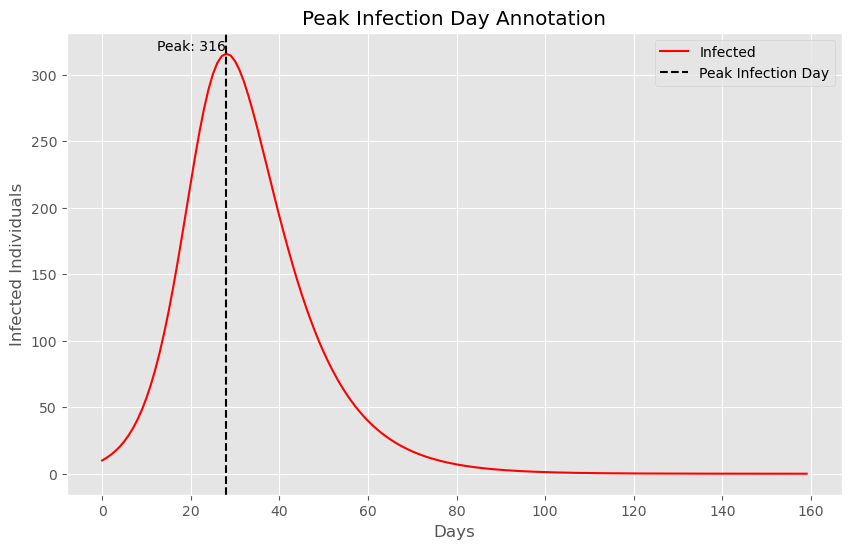
Using my previous SIR model from Project 2 that covered using the NumPy library, I decided to revisit the project code to construct graphs for various scenarios using Matplotlib. I first began by transferring the necessary code from my modeling infection spread to recreate the simulated data. I also transerred the Monte Carlo simulation code to include in the visualization. First, I began by importing the necessary libraries that I would need for constructing the graphs for the entire project. Then I began constructing simple line plots. The first line plot demonstrates how to plot multiple sets of data on the same graph and how to annotate the graph and change the line styles and colors using appropriate parameters. Figure ## below shows the code and the output graph.

Figure and Code



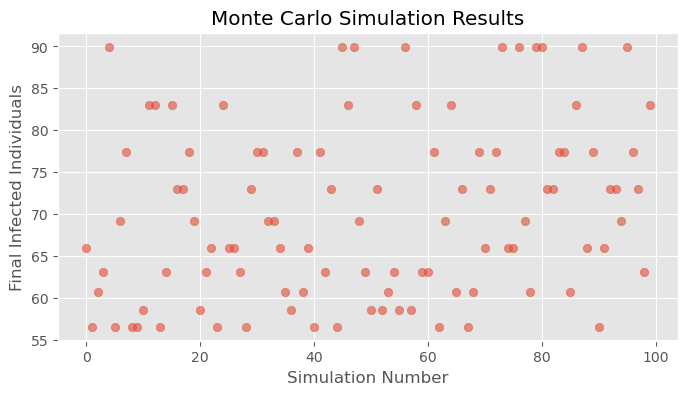
To further demonstrate how to annotate a graph, I isolated the line plot for infected. Then I added a dashed vertical line along with the peak value to represent on the graph the peak infection day. Figure ## shows the code and output plot.

Figure and Code



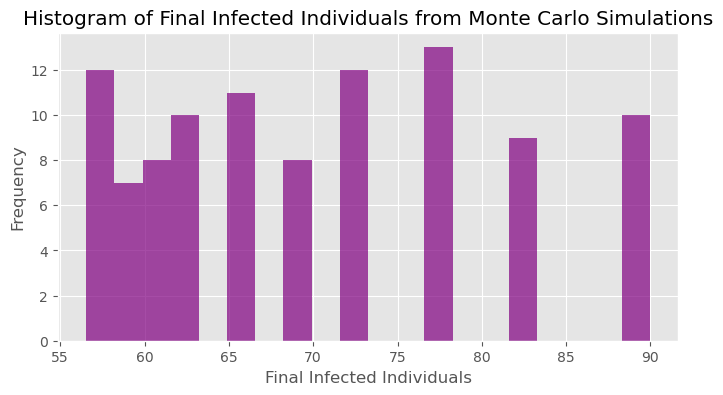
I decided to run the Monte Carlo simulation. I made a scatter plot for this data to show the number of final infected individuals for each simulation number. Figure ## shows the code along with the resulting plot.

Figure and Code



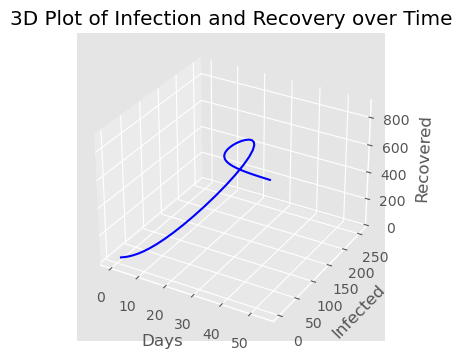
Though the scatter plot was a good representation of the data, I felt that a histogram would be a better visual to display the frequency of the final infected individuals after each simulation.

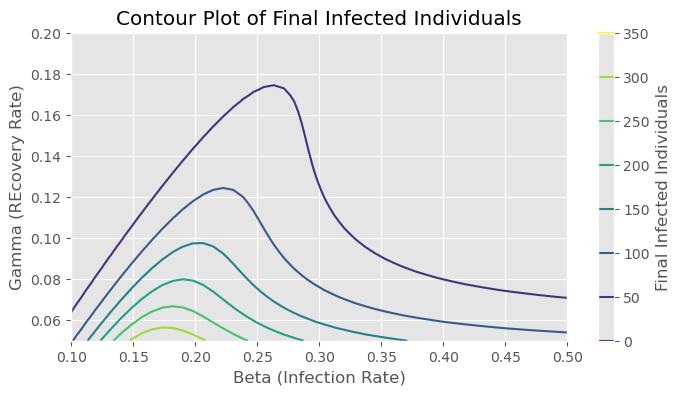
Figure and Code



To demonstrate working with a 3D contour plot, I use the original SIR model and plot the relationship between the infected and recovered over time. Figure ## shows the code and the resulting plot. From this plot we are able to visualize the dynamics of disease spread. With this plot we can see how the number of infected and recovered individuals changes over time simultaneously.

Figure and Code



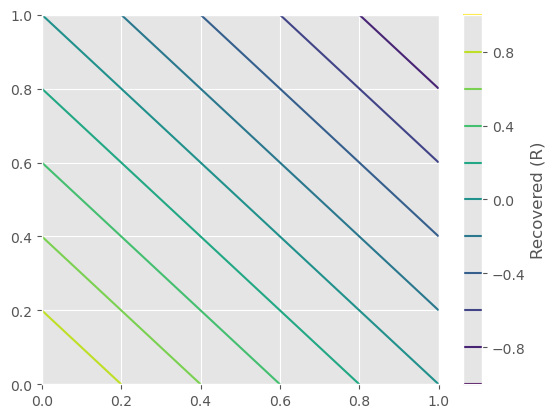


The SIR model is an excellent tool to simulate and learn about disease spread. Though it is limited, it is still widely used in the real world to model infections and in classroom environments to teach differential equations. For a fun experiment, I wanted to create a more interactive graph that would allow the user to have more control to explore the different model parameters (susceptible, infected, recovered, beta, gamma) and observe the result. For this, I leveraged *`ipywidgets`* and imported the interact, Float slider, IntSlider, and BoundedIntText functions. I constructed a new function that would be used to call the SIR model and update its values as parameters change. Figure ## shows the code and the output.

Figure and Code

Another important concept in studying parameter relationships in differential equations is the trajectory plot also known as the phase plot plane. A phase plane plot visualizes the trajectory of the susceptible and infected populations in the SIR model. This code shown below in Figure ## calculates this relationship over time. The purpose of this graph is to help understand the dynamic interactions between susceptible and infected individuals. There are better ways to produce this plot, however I wanted to attempt to recreate this plot using Matplotlib. Figure ## shows the ouptut.

Figure and Code



# No description has been provided for this image

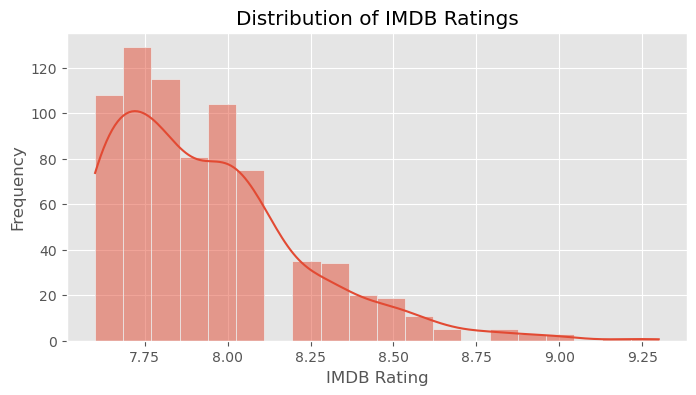
# 3. Adapting Movie Analysis Model for Visualization

To continue showing how Matplotlib can be used to visualize data and provide a more comprehensive understanding of data, I used my previous project that demonstrated using Pandas by analyzing a movie data and converting it to a dataframe for analysis. I first began by reading in the CSV and of top 1000 movies. Using my understanding of Pandas, I cleaned the data up by reordering columns, reducing the types of genres to make them a little more general, dropping any unused columns, removing rows containing null data. Further, I converted the data that should be a numerical datatype, sorted the data by ratings, and finally adding a rank column. The code and the result are shown below in Figure ###.

Figure and code

To begin to better understand the data, I plotted the IMDB\_Rating in a histogram to better understand the distribution of ratings. Here we see that the majority of top movies tend to have a rating of 7.6 to 8.20. It is rare to have a movie with a rating above a 9. Figure ## shows the code and the results.

Figure and code



Next I used a similar histogram plot to show the distribution of gross earnings. Figure ### shows the code and output. Along with showing the distribution of runtimes. The majority of the top movies tend to stick around a runtime of 100 to 125 minutes.

Figure and code

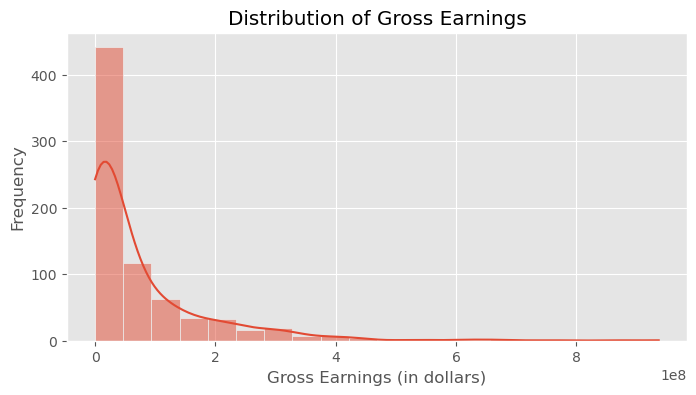
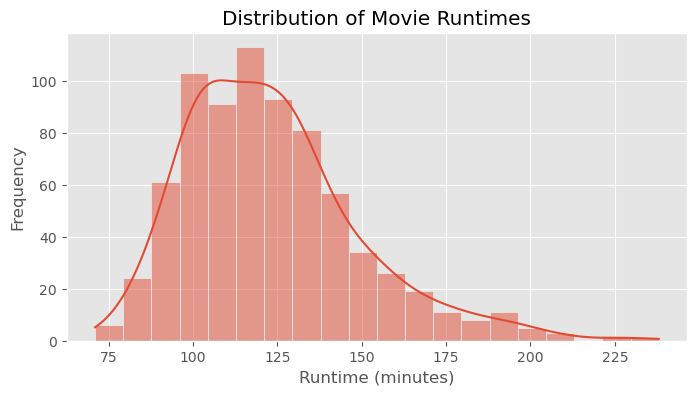


Figure and code



The next bar plot displays the number of movies that are considered the top movies for each year. The code processes the dataset to extract release years, handling missing and incorrect data, and the plots the counts using *`seaborn`*. This graph provides insight into trends and patterns in movie production over time.

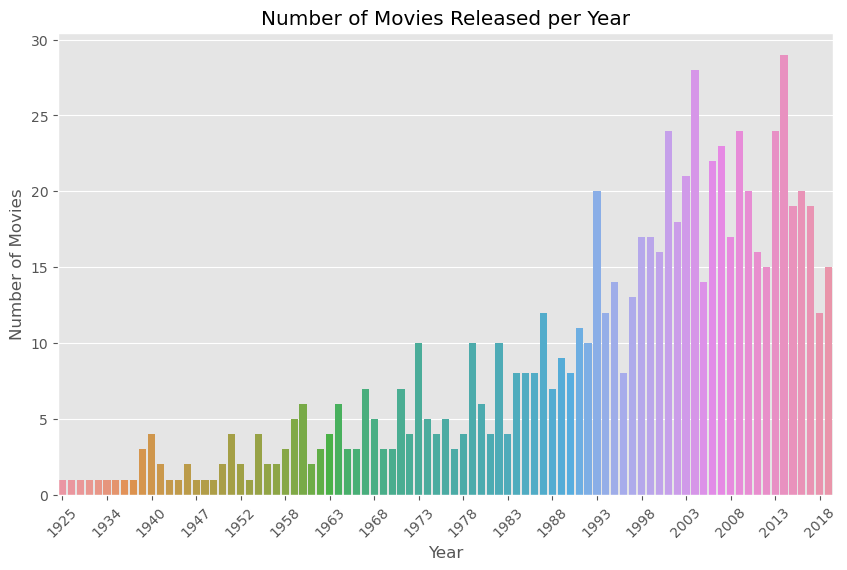
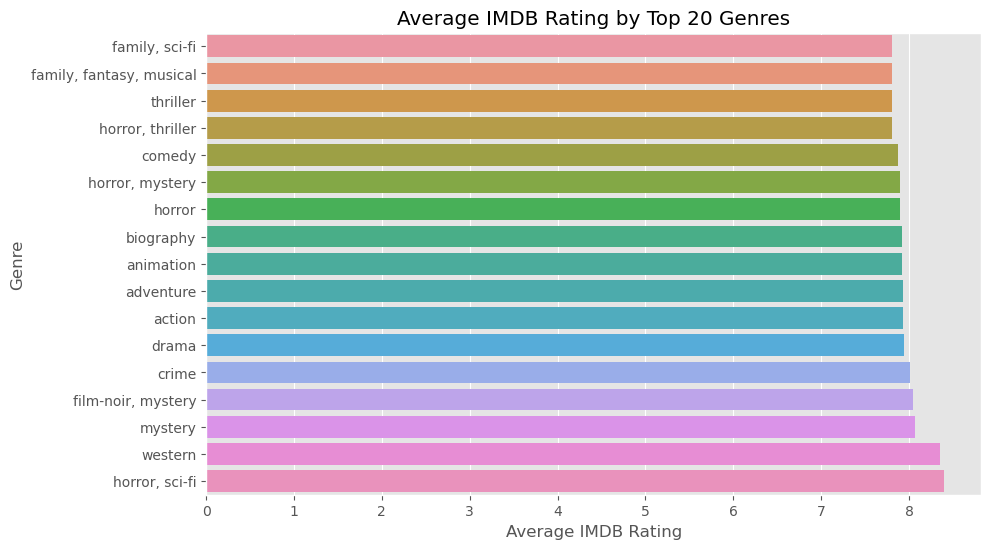


Figure and code

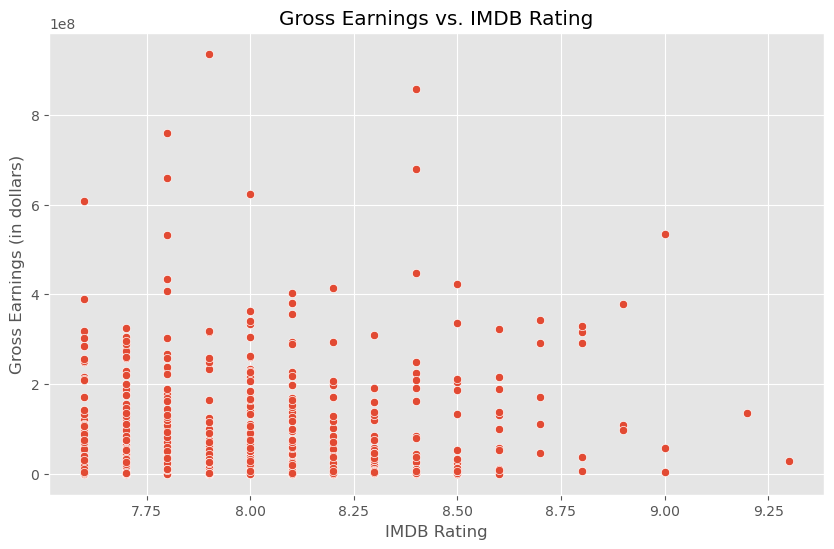
The next graph is another bar plot that shows the average IMDB ratings for the different genres. The code simplifies genre names using regex, groups the data by genre, and calculates the average rating for each. This visualization highlights differences in audience reception across various genres.

Figure and code



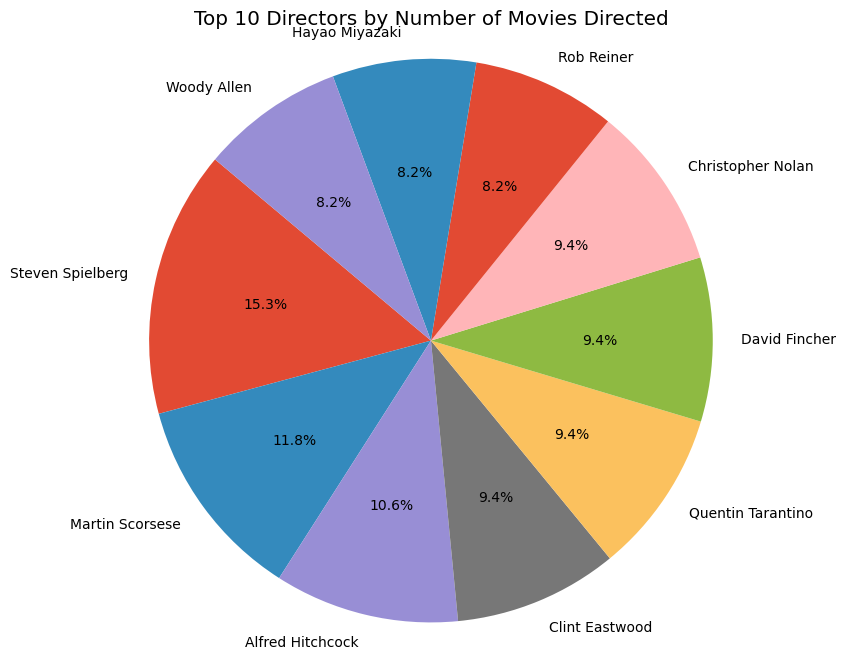
The next graph is a scatter plot to examine the relationship between gross earnings and IMDB ratings. The code plots these two variables, showing how movie earnings correlate with their ratings to identify potential patterns in the data. Figure ## shows the output and code.

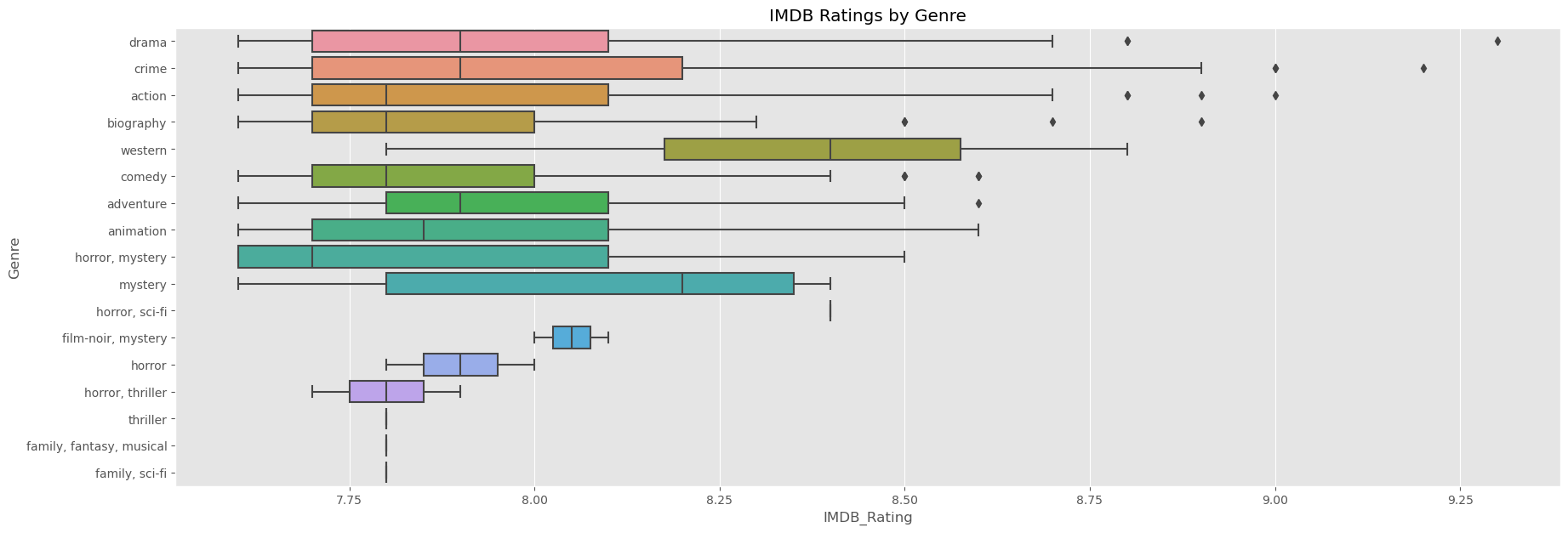
Figure and code

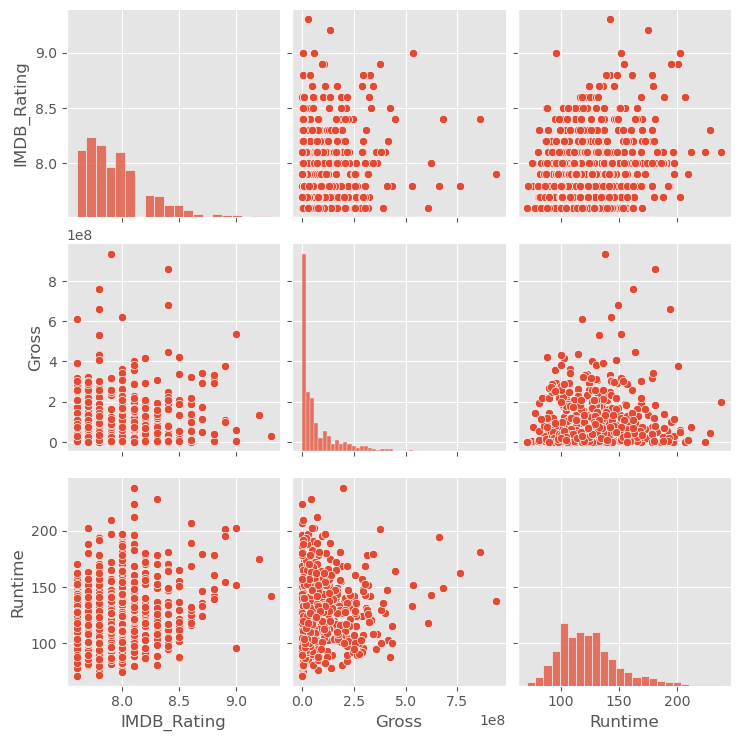


Next I used a pie chart to represent the top 10 directors by the number of movies directed. The code counts the occurrences of each director in the dataset and visualizes the top 10. This graph gives a view of which directors have the most significant presence in the top 1000 movies.

Figure and code

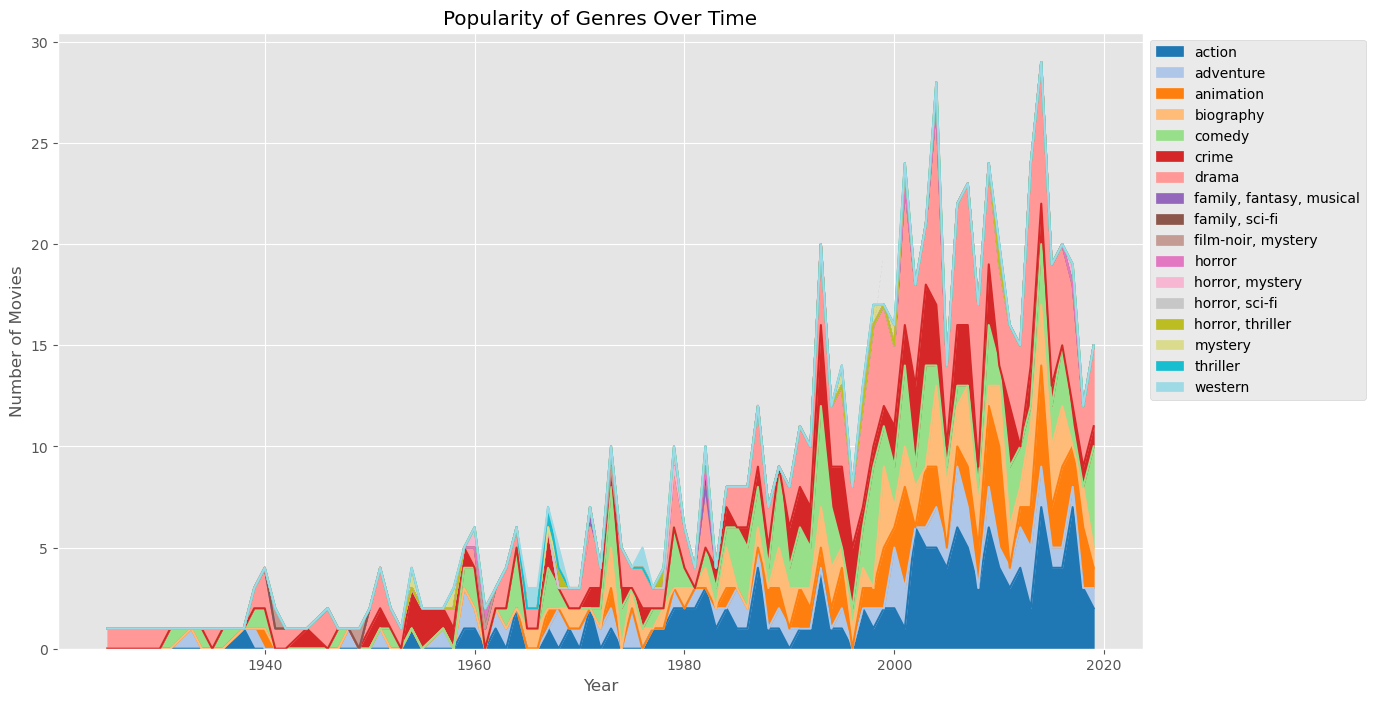




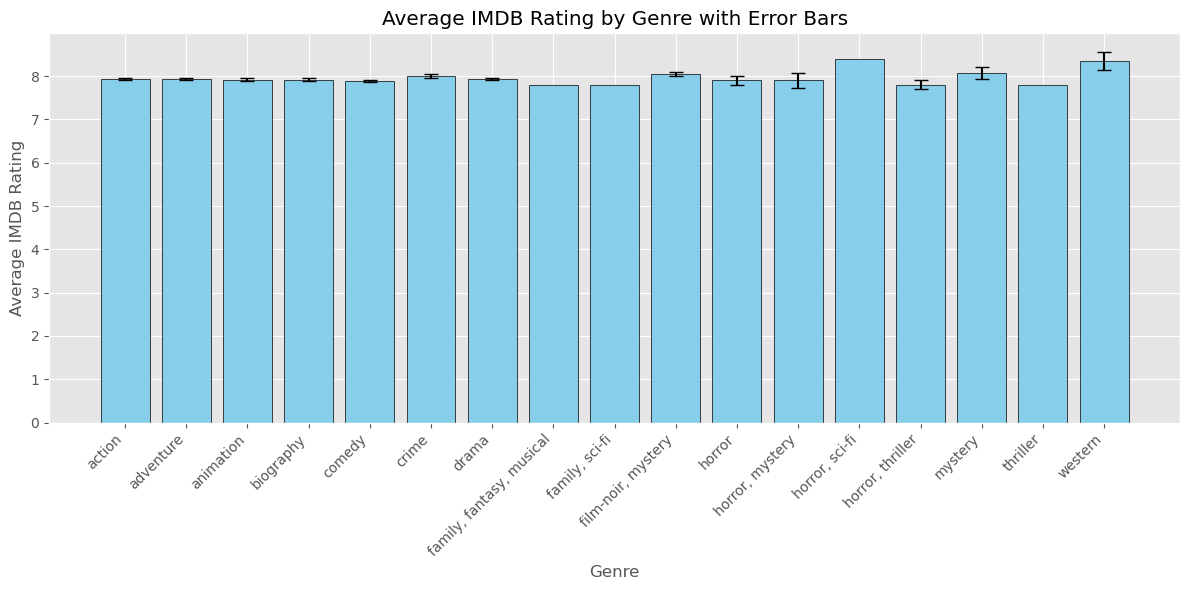


The next graph is an area plot that shows the popularity of different genres over time. The code groups the data by release year and genre, then plots the number of movies in each genre per year to reveal any trends in genre popularity over the years.

Figure and code



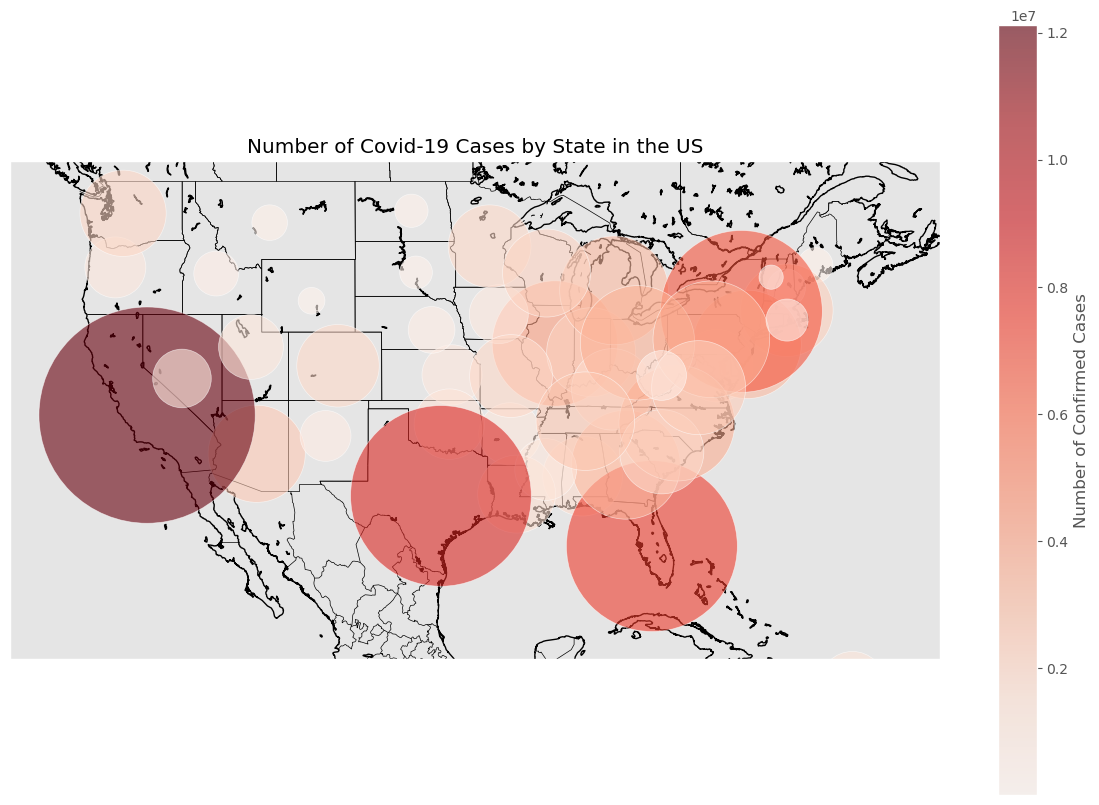
The last graph is a bar plot with error bars to visualize the average IMDB rating for each genre. The error bars represent the standard error of the mean, which help to provide a visual indication of the variability around the mean rating for each genre. Figure ## shows the code and output.



# 4. Other Models

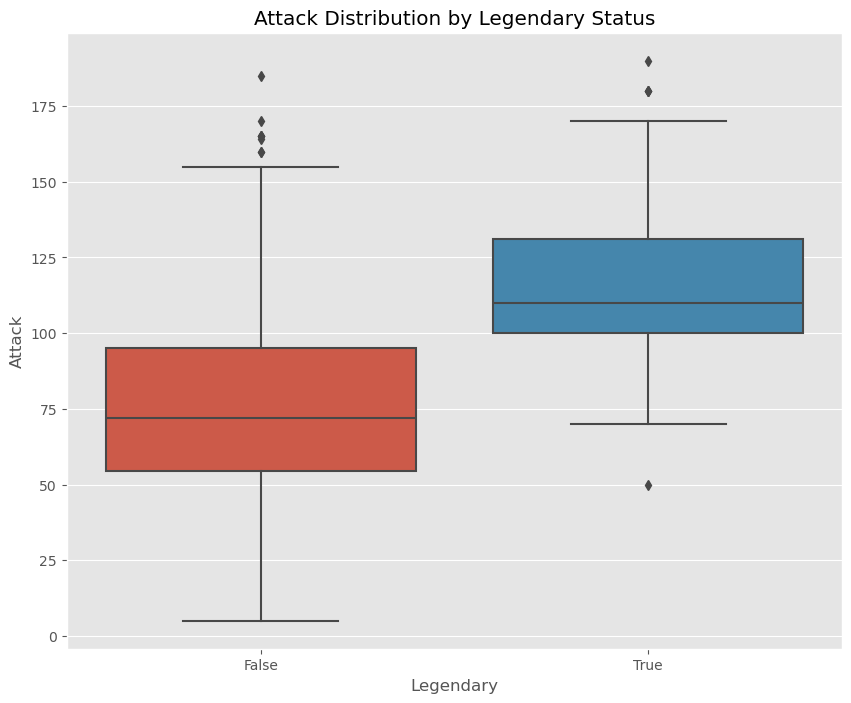
To demonstrate the ability of using the Basemap toolkit, I decided to visualize COVID-19 cases across the US. I downloaded a dataset from John Hopkins University GitHuub. I cleaned the data up by dropping rows that did not contain any information about the latitude and longitude, and filling specific columns that contain missing data with zero. I used the Basemap toolkit to plot the geographic data, which helps provide a spatial understanding of COVID-19’s impact. Figure ## shows the code and output.

Figure and code



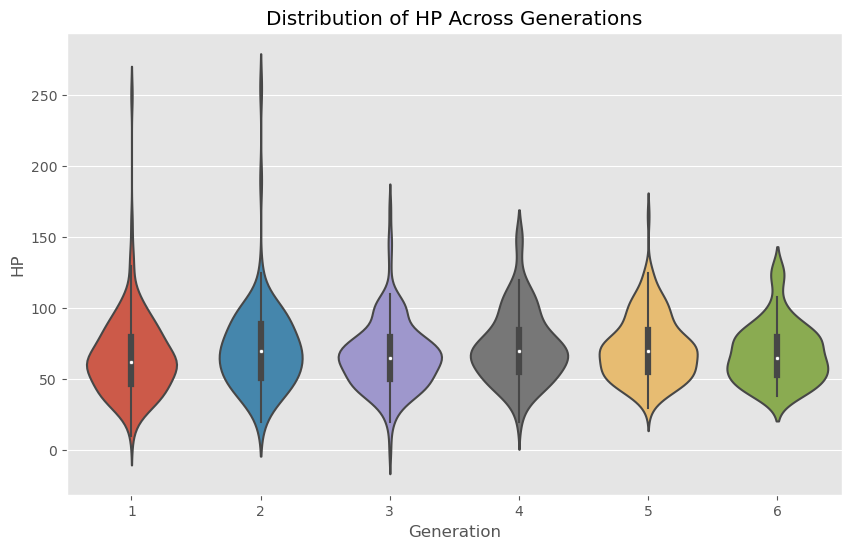
I also imported a Pokemon dataset that contains basic information such as the Pokemon name,, Type, HP, Attack, and other stats. The first graph is a box plot to show the distribution of attack values for Pokemon, grouped by if they are a legendary or not. It uses *`seaborn`* to create the box plot, allowing you to visually compare the central tendency and spread of attack values. Figure # shows the code and output.

Figure and code



The next graph is a violin plot that displays the distribution of HP values for Pokemon, separated by generation. It uses *`seaborn`* to create the plot which helps provide a view of the data distribution for each generation. Figure ## shows the code and output.

Figure and code



# 5. Conclusion

This report documents my journey in learning Matplotlib, a powerful data manipulation library in Python. Key concepts I explored include data structures essential for handling and analyzing structured data. I learned to perform various data operations, including data indexing, merging datasets, grouping data and more.

By using real data, I was able to explore how to go about cleaning the data up properly before beginning to analyze it. Many errors I encountered were related to missing values and data types, which are significantly different from syntax errors I encountered in previous projects. However, through practice and persistence, I was able to clean the data up and obtain datasets that could be analyzed properly using Pandas.

# References

1. Harshit Shankhdhar. (2021). IMDB Dataset of Top 1000 Movies and TV Shows. Retrieved from https://www.kaggle.com/datasets/harshitshankhdhar/IMDB-dataset-of-top-1000-movies-and-tv-shows
2. The Movie Database (TMDB). (2018). TMDB 5000 Movie Dataset. Retrieved from https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
3. VanderPlas, J. (*2016*).  *Python Data Science Handbook*. O’Reilly Media. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/index.html