Project 4: Visualization with Matplotlib

John Wesley Mathis

Dr. Anthony Choi

June 15, 2024

ECE/SSE 591, Summer 2024

Table of Contents:

[Deliverable Table 6](#__RefHeading___Toc2569_331549091)

[1. Introduction 7](#__RefHeading___Toc2573_331549091)

[2. Adapting SIR Model for Visualization 8](#__RefHeading___Toc2577_331549091)

[3. Adapting Movie Analysis Model for Visualization 21](#__RefHeading___Toc2969_2801450533)

[4. Other Models 34](#__RefHeading___Toc8059_276922697)

[5. Conclusion 43](#__RefHeading___Toc2609_331549091)

[References 44](#__RefHeading___Toc3692_331549091)

Table of Figures

[Figure 1: Importing Libraries and Customizing Matplotlib 8](#Figure!0|sequence)

[Figure 2: SIR Model Code 9](#Figure!1|sequence)

[Figure 3: SIR Model Dynamics Line Plot Code 10](#Figure!2|sequence)

[Figure 4: SIR Model Dynamics Line Plot 10](#Figure!3|sequence)

[Figure 5: Peak Infection Day Annotation Code 11](#Figure!4|sequence)

[Figure 6: Peak Infection Day Annotation Plot 11](#Figure!5|sequence)

[Figure 7: Monte Carlo Simulation Code 12](#Figure!6|sequence)

[Figure 8: Monte Carlo Simulation Results Code 12](#Figure!7|sequence)

[Figure 9: Monte Carlo Simulation Results Scatter Plot 13](#Figure!8|sequence)

[Figure 10: Final Infected Individuals Histogram Code 13](#Figure!9|sequence)

[Figure 11: Final Infected Individuals Histogram Plot 14](#Figure!10|sequence)

[Figure 12: 3D Plot of SIR Model Code 14](#Figure!11|sequence)

[Figure 13: 3D Plot of SIR Model 15](#Figure!12|sequence)

[Figure 14: Contour Plot of Final Infected Individuals Code 16](#Figure!13|sequence)

[Figure 15: Contour Plot of Final Infected Individuals 17](#Figure!14|sequence)

[Figure 16: Interactive SIR Model Dynamics Code 18](#Figure!15|sequence)

[Figure 17: Interactive SIR Model Dynamics Plot and Layout 19](#Figure!16|sequence)

[Figure 18: Trajectory Plot Code 20](#Figure!17|sequence)

[Figure 19: Trajectory Plot (a) 20](#Figure!18|sequence)

[Figure 20: Trajectory Plot (b) 21](#Figure!19|sequence)

[Figure 21: Movie Analysis Code (a) 22](#Figure!20|sequence)

[Figure 22: Movie Analysis Code (b) 23](#Figure!21|sequence)

[Figure 23: Movie Analysis Code (c) 23](#Figure!22|sequence)

[Figure 24: Distribution of IMDB Ratings Code 24](#Figure!23|sequence)

[Figure 25: Distribution of IMDB Ratings Histogram Plot 24](#Figure!24|sequence)

[Figure 26: Distribution of Gross Earnings Code 25](#Figure!25|sequence)

[Figure 27: Distribution of Gross Earnings 25](#Figure!26|sequence)

[Figure 28: Distribution of Movie Runtimes Code 26](#Figure!27|sequence)

[Figure 29: Distribution of Movie Runtimes 26](#Figure!28|sequence)

[Figure 30: Number of Movies Released per Year Code 27](#Figure!29|sequence)

[Figure 31: Number of Movies Released per Year Plot 27](#Figure!30|sequence)

[Figure 32: Average IMDB Rating by Genre Code 28](#Figure!31|sequence)

[Figure 33: Average IMDB Rating by Genre Plot 28](#Figure!32|sequence)

[Figure 34: Scatter Plot of Gross Earnings vs IMDB Rating Code 29](#Figure!33|sequence)

[Figure 35: Scatter Plot of Gross Earnings vs IMDB Rating 29](#Figure!34|sequence)

[Figure 36: Visualizing Top Directors Code 30](#Figure!35|sequence)

[Figure 37: Visualizing Top Directors Pie Chart 30](#Figure!36|sequence)

[Figure 38: Box Plot of IMDB Ratings by Genre Code 31](#Figure!37|sequence)

[Figure 39: Box Plot of IMDB Ratings by Genre 31](#Figure!38|sequence)

[Figure 40: Pairplot of Ratings, Gross Earnings, and Runtime Code 32](#Figure!39|sequence)

[Figure 41: Pairplot of Ratings, Gross Earnings, and Runtime 32](#Figure!40|sequence)

[Figure 42: Genre Popularity over Time with Area Plot Code 33](#Figure!41|sequence)

[Figure 43: Genre Popularity over Time with Area Plot 33](#Figure!42|sequence)

[Figure 44: Average IMDB Rating by Genre with Error Bars Code 34](#Figure!43|sequence)

[Figure 45: Average IMDB Rating by Genre with Error Bars 34](#Figure!44|sequence)

[Figure 46: Importing and Cleaning COVID-19 Dataset Code 35](#Figure!45|sequence)

[Figure 47: Importing and Cleaning COVID-19 Dataset Output 35](#Figure!46|sequence)

[Figure 48: Visualizing COVID-19 Cases in the US with Basemap Code 36](#Figure!47|sequence)

[Figure 49: Visualizing COVID-19 Cases in the US with Basemap 37](#Figure!48|sequence)

[Figure 50: Subplots of Average Number of Confirmed COVID-19 Cases and Deaths by State Code 38](#Figure!49|sequence)

[Figure 51: Subplots of Average Number of Confirmed COVID-19 Cases and Deaths by State 39](#Figure!50|sequence)

[Figure 52: Importing Pokemon dataset code 40](#Figure!51|sequence)

[Figure 53: Output of Pokemon dataset 41](#Figure!52|sequence)

[Figure 54: Violin Plot with Pokemon Data Code 41](#Figure!53|sequence)

[Figure 55: Violin Plot with Pokemon Data 42](#Figure!54|sequence)

# Deliverable Table

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

|  |  |
| --- | --- |
| Deliverables | Location |
| Simple Line Plots | p. 9-10 |
| Simple Scatter Plots | p. 11-13 |
| Visualizing Errors | p.34 |
| Density and Contour Plots | p. 16 |
| Histograms, Binnings, and Density | p. 23 |
| Customizing Plot Legends | p. 17-18 |
| Customizing Colorbars | p. 36 |
| Multiple Subplots | p. 37-38 |
| Text and Annotation | p. 11 |
| Customizing Ticks | p. 27 |
| Customizing Matplotlib: Configurations and Stylesheets | p. 8 |
| Three-Dimensional Plotting in Matplotlib | p. 14-15 |
| Geographic Data with Basemap | p. 36 |
| Visualization with Seaborn | p. 40-41 |

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.

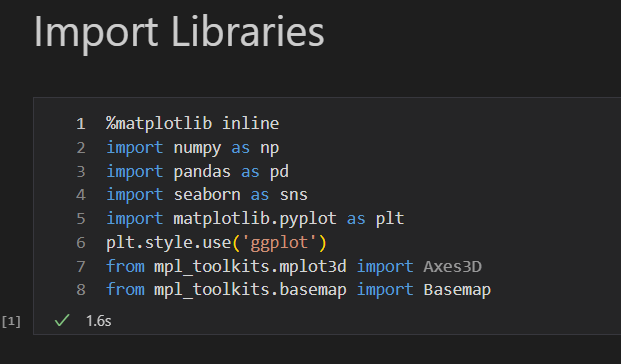
# 1. Introduction

Matplotlib provides a comprehensive 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.

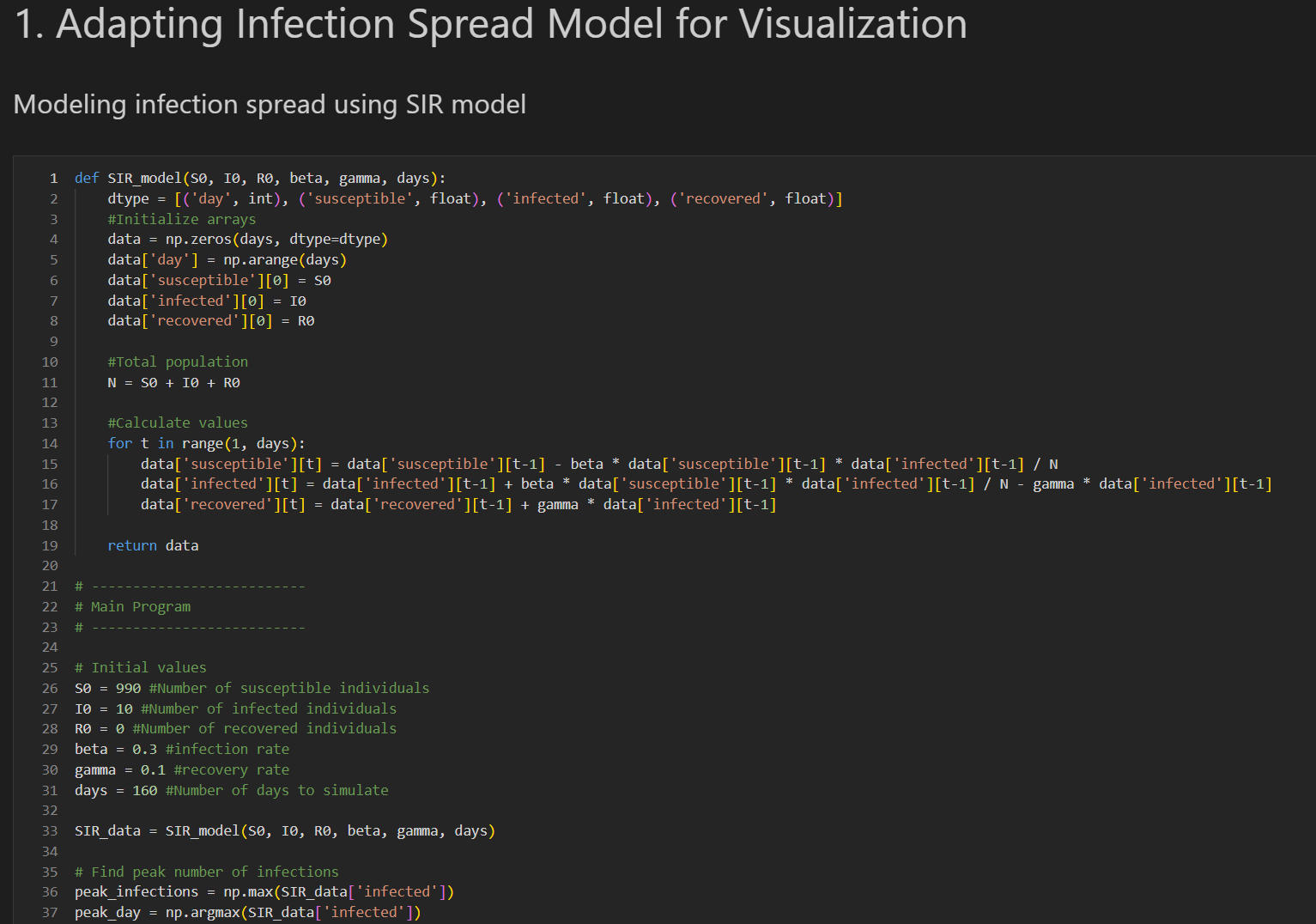
This report aims to demonstrate my proficiency in Python data visualization techniques as covered in Chapter 4 of the “Python Data Science Handbook” by Jake VanderPlas (2016). This report attempts to illustrate the core concepts and functionalities of the Pandas library by implementing the concepts into 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

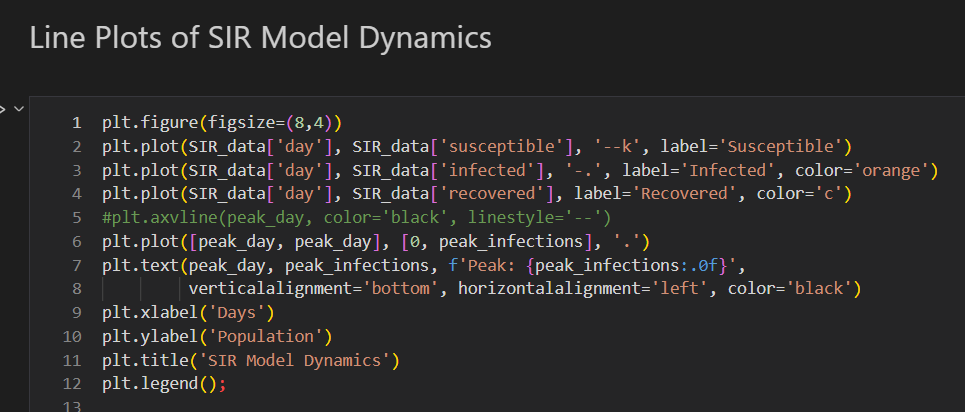
Before beginning any of the projects, I first imported the necessary libraries that I would be using, such as NumPy, Matplotlib, Seaborn and more. Figure 1 shows the libraries I used for the entire project. Additionally, I used *`plt.style.use()`* to customize Matplotlib to my likening. For the graphs, I chose to use the *ggplot* style for my charts and graphs. Along with the necessary libraries, I also imported a couple of toolkits that I would need for later graphs.

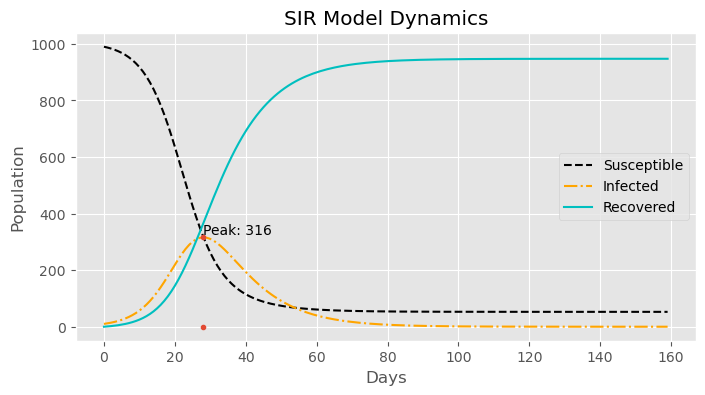
Figure 1: Importing Libraries and Customizing Matplotlib

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 transferred the Monte Carlo simulation code to include in the visualization. Figure 2 below shows the final code that I made for my SIR model analysis.

Figure 2: SIR Model Code

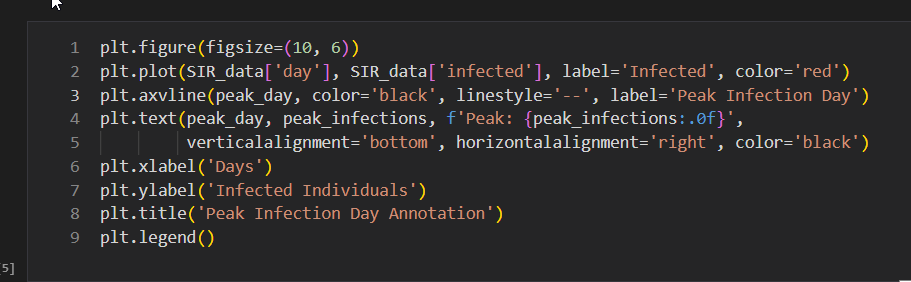
Once the data was generated, I began constructing simple line plots. The first graph demonstrates various concepts from chapter 4 including how to plot multiple sets of data on the same graph and how to annotate the graph and change the line styles and colors using appropriate parameters. Figures 3 and 4 below shows the code and the output graph.

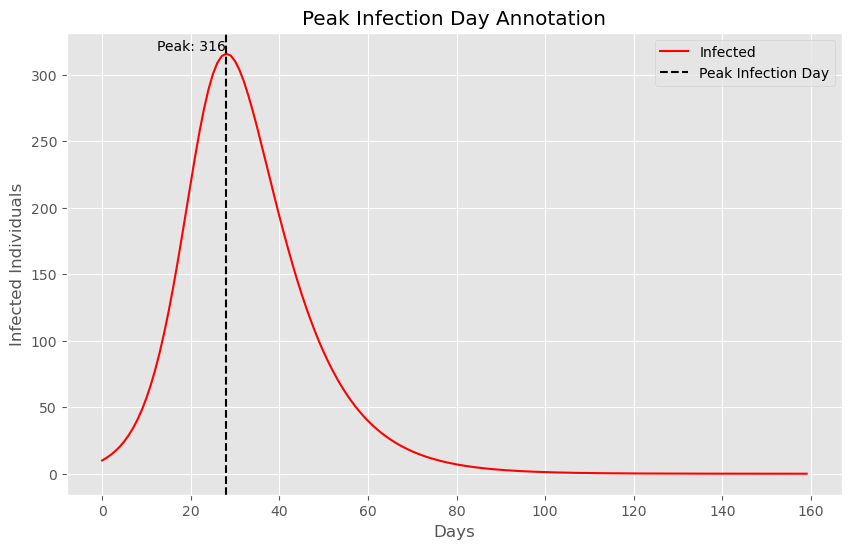
Figure 3: SIR Model Dynamics Line Plot Code

Figure 4: SIR Model Dynamics Line Plot

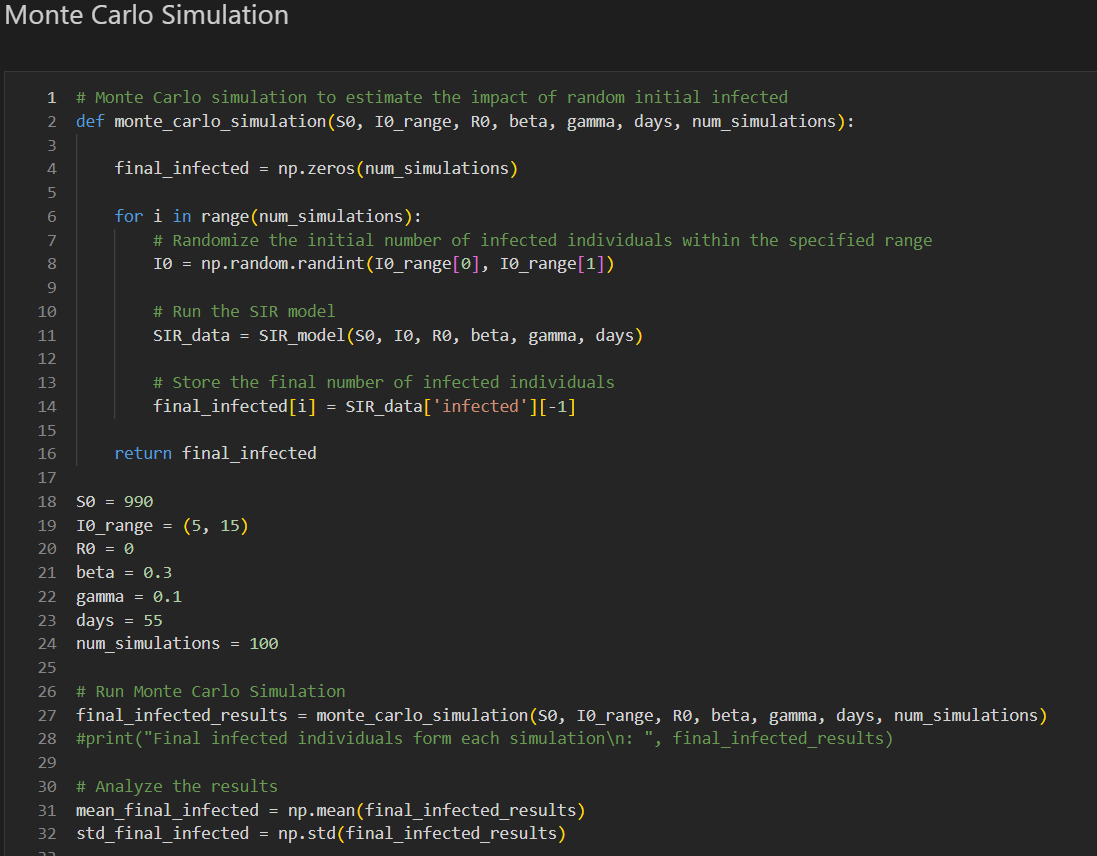
To further demonstrate how to annotate a graph, I isolated the line plot for infected. Then I added a dashed vertical line along with the peak value to represent on the graph the peak infection day. Figures 5 and 6 show the code and output plot. This was actually the first plot I constructed for this data. But typically, for infection models, the graph in figure 4 is what is used.

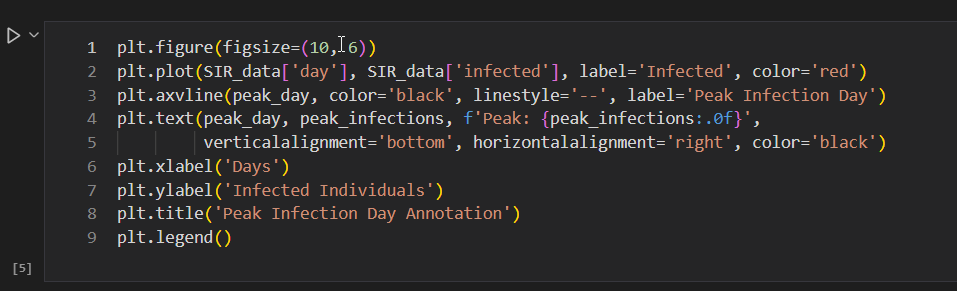
Figure and Code

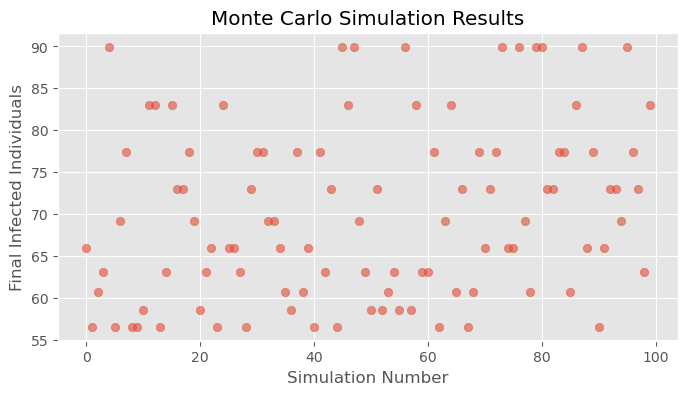
Figure 5: Peak Infection Day Annotation Code

Figure 6: Peak Infection Day Annotation Plot

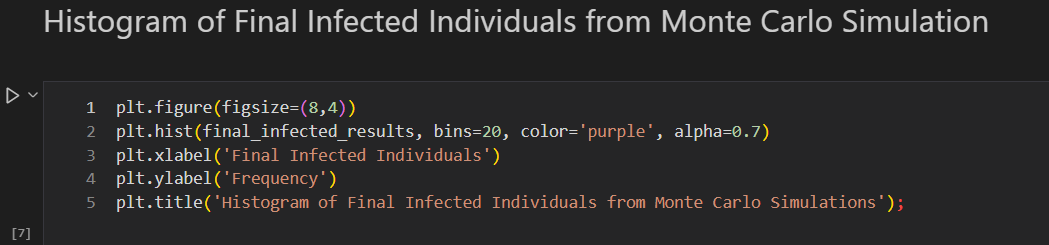
Next I ran the Monte Carlo simulation (Fig. 7). I made a scatter plot for this data to show the number of final infected individuals for each simulation number. Figures 8 and 9 show the code along with the resulting scatter plot.

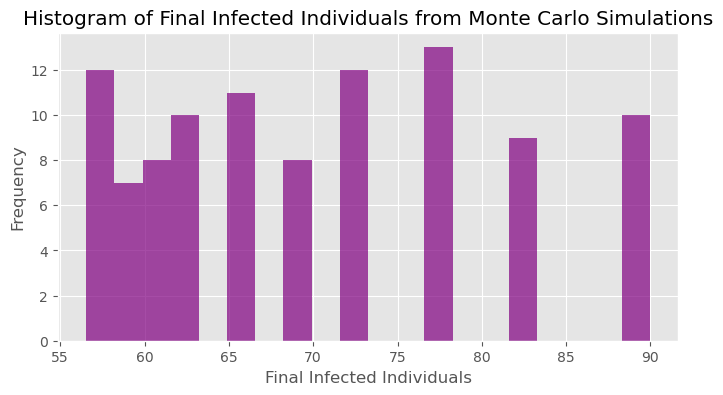
Figure 7: Monte Carlo Simulation Code

Figure 8: Monte Carlo Simulation Results Code

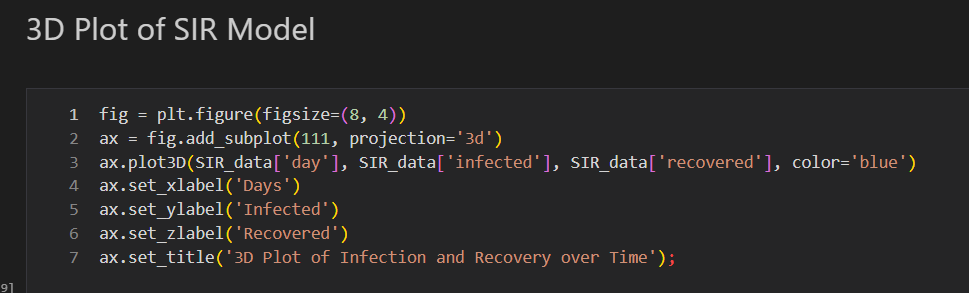
Figure 9: Monte Carlo Simulation Results Scatter Plot

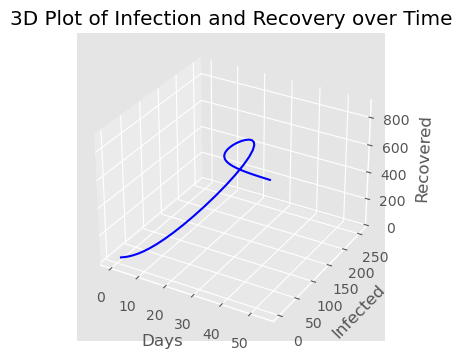
Though the scatter plot was a good representation of the data, I felt that a histogram would be a better visual to display the frequency of the final infected individuals after each simulation. So I used *`plt.hist()`* to construct a histogram with 20 bins. Figures 10 and 11 show the code and the output.

Figure 10: Final Infected Individuals Histogram Code

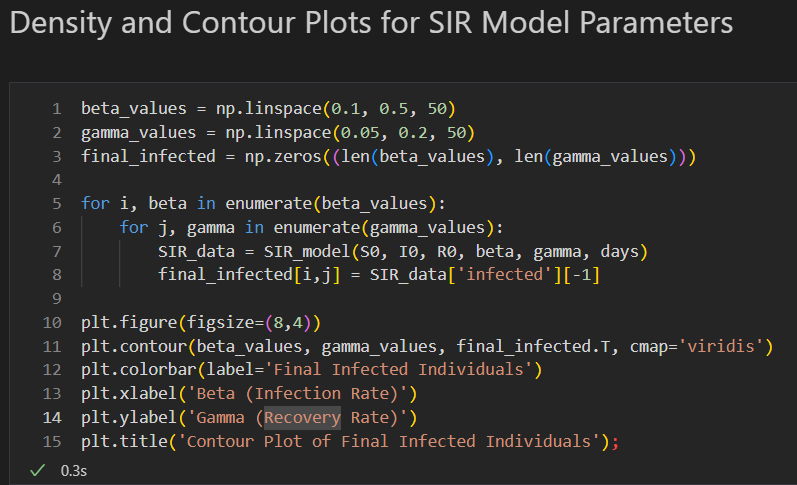
Figure 11: Final Infected Individuals Histogram Plot

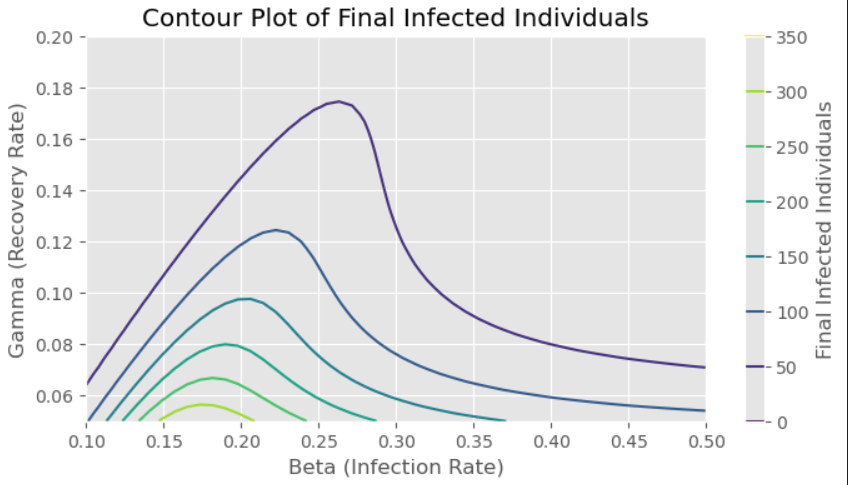
To demonstrate working with a 3D contour plot, I used the original SIR model to plot the relationship between the infected and recovered over time. Figures 12 and 13 show the code and the resulting plot. From this graph we are able to visualize the dynamics of disease spread. We can see how the number of infected and recovered individuals changes over time simultaneously.

Figure 12: 3D Plot of SIR Model Code

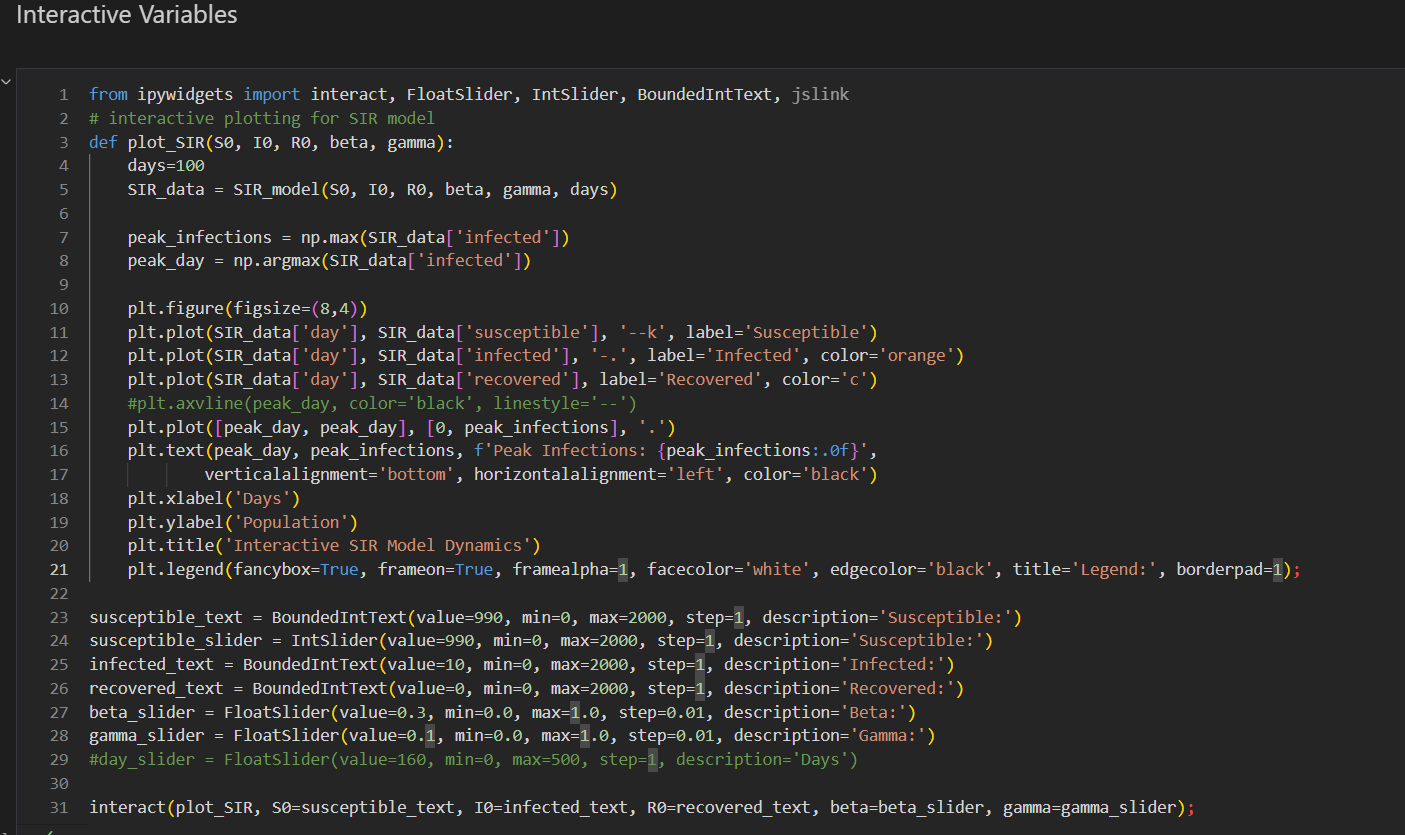
Figure 13: 3D Plot of SIR Model

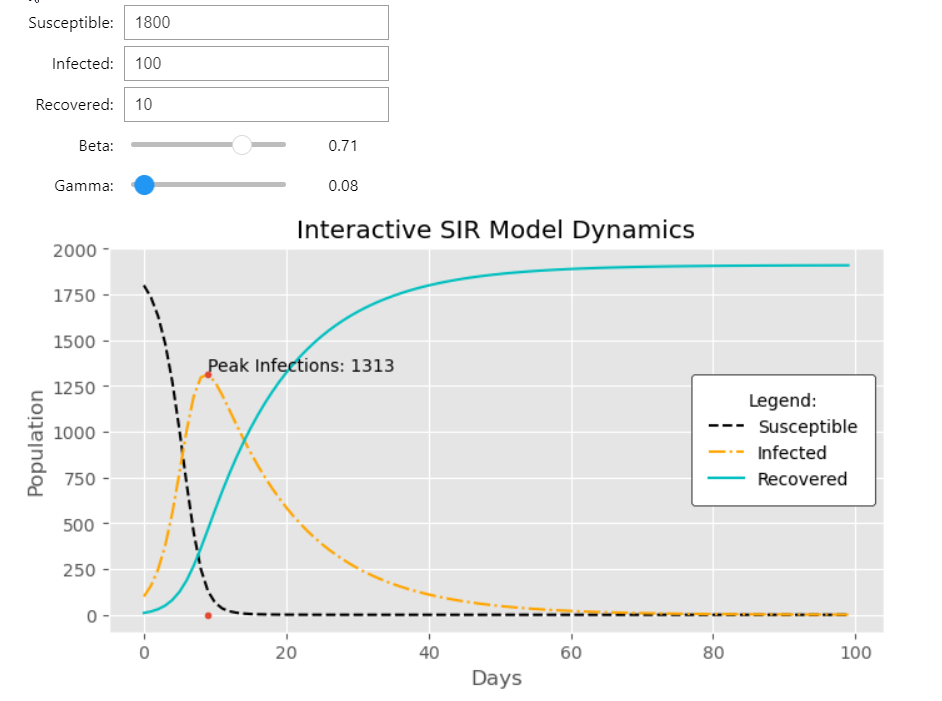
I coded a contour plot to visualize the final number of infected individuals as a function of two parameters: the infection rate and the recovery rate. This helps to understand the sensitivity of the SIR model with changes in *`beta`* and *`gamma`* parameters. Areas on the graph with higher contours indicate that it will lead to more significant final infected counts. Figures 14 and 15 show the code and the output.

Figure 14: Contour Plot of Final Infected Individuals Code

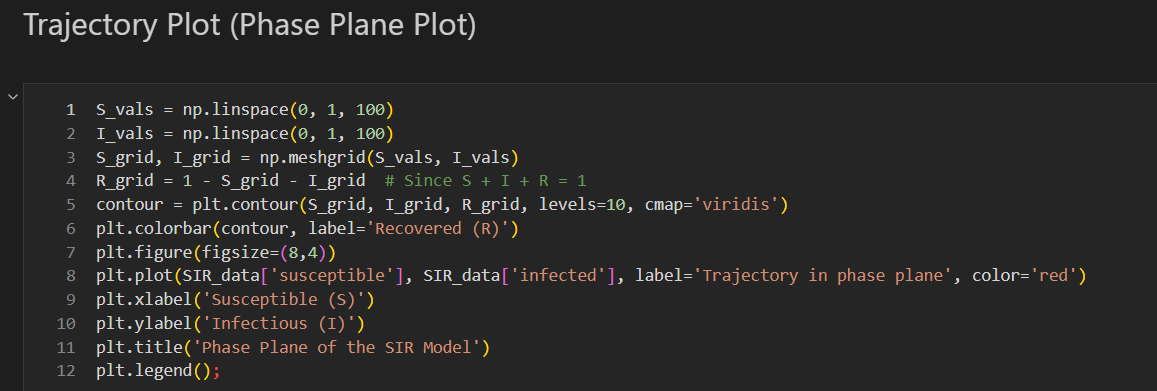
Figure 15: Contour Plot of Final Infected Individuals

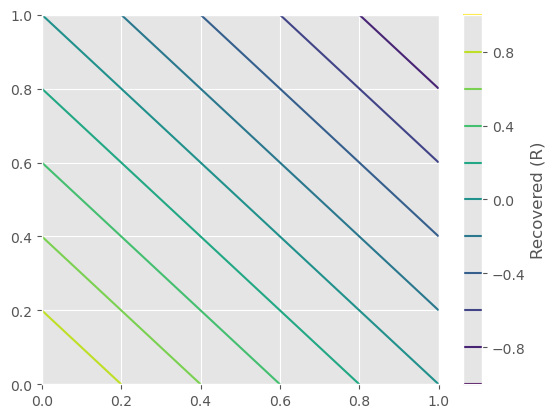
The SIR model is an excellent tool to simulate and learn about disease spread. Though it is limited, it is still widely used in the real world to model infections and in classroom environments to teach differential equations. For a fun experiment, I wanted to create a more interactive graph that would 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. At first, I used sliders for all the parameters. But eventually I decided to use *BoundIntText* so that the user could easily type in whole numbers for the number of people that would be susceptible, infected and recovered. But I left the beta and gamma sliders as *FloatSliders* so the user would understand what values are acceptable and to allow easy manipulation. Figures 16 and 17 show the code and the output with new parameters entered to show a different line plot graph. Additionally, I attempted to allow the user to change the number of days. However, because of how I set everything up, I continued to get errors. Unfortunately, I wasn’t able to work out the errors to show this but, maybe at a future date, I’ll have made a completely interactive model.

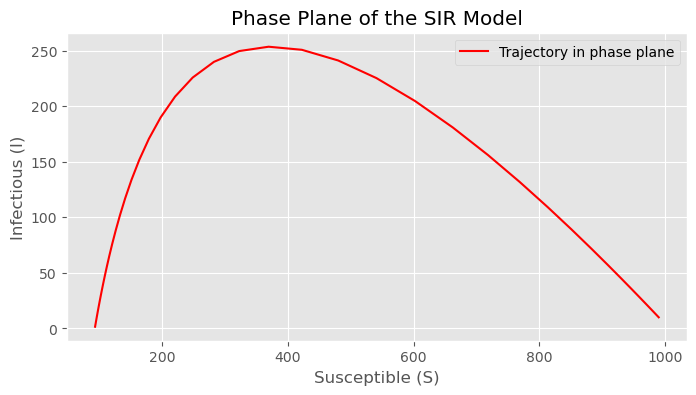
Figure 16: Interactive SIR Model Dynamics Code

Figure 17: Interactive SIR Model Dynamics Plot and Layout

Another important concept in studying parameter relationships in differential equations is the trajectory plot also known as the phase plane plot. A phase plane plot visualizes the trajectory of the susceptible and infected populations in the SIR model. This code shown below in Figure 18 calculates this relationship over time. The purpose of this graph is to help understand the dynamic interactions between susceptible and infected individuals using *plt.contour and plt.plot*. There are better ways to produce this plot, however I wanted to attempt to recreate this plot using Matplotlib. Figure 19-20 shows the ouptut. I attempted to overlay the graphs on top of each other. However, I kept getting bad results that did not make since. So I simply kept the two plots separated. s

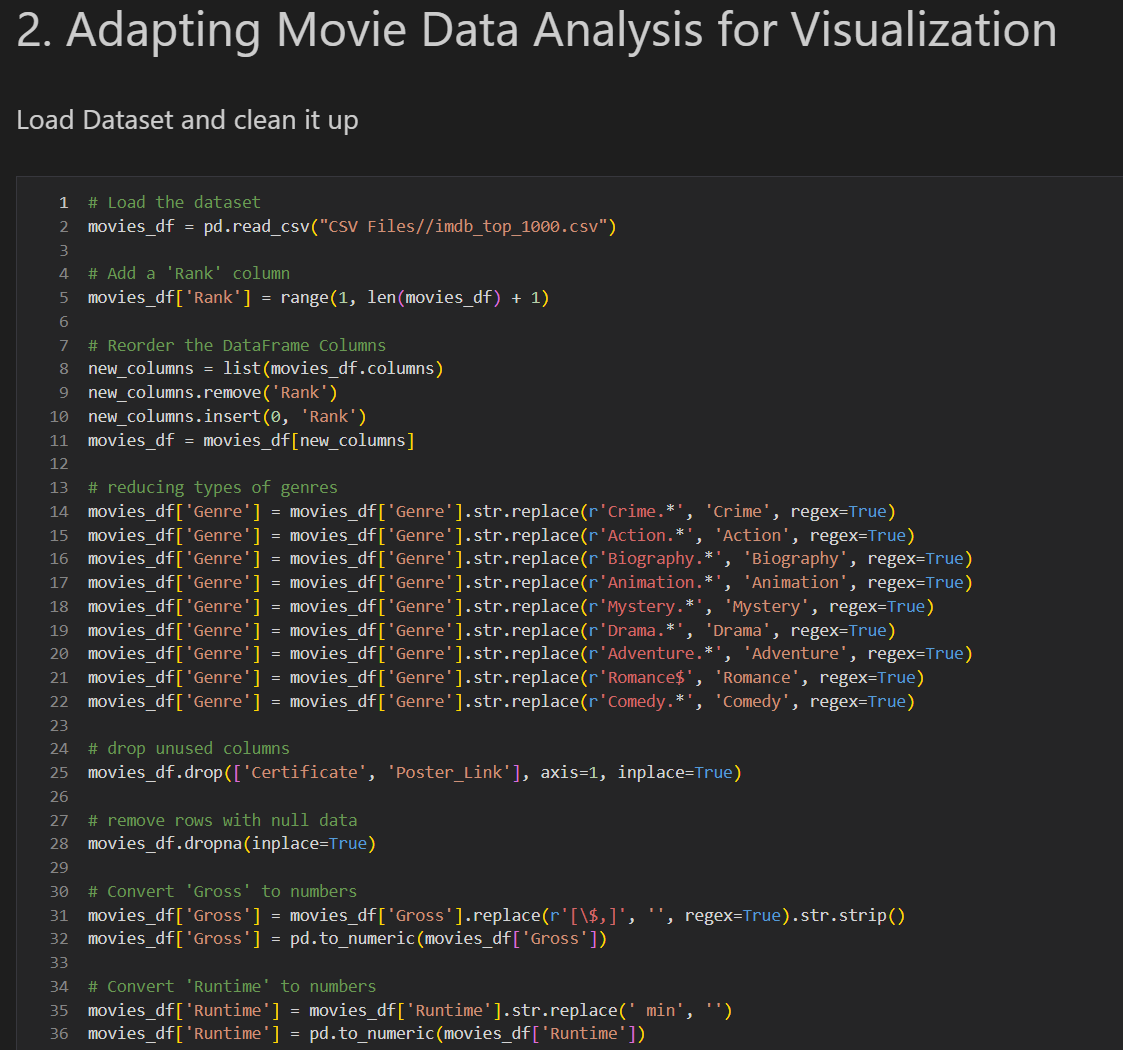
Figure 18: Trajectory Plot Code

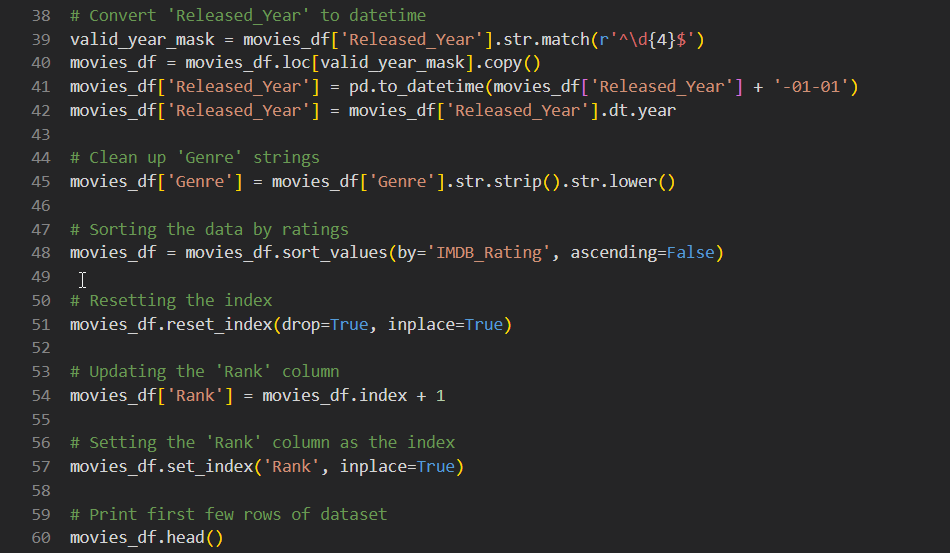
Figure 19: Trajectory Plot (a)

Figure 20: Trajectory Plot (b)

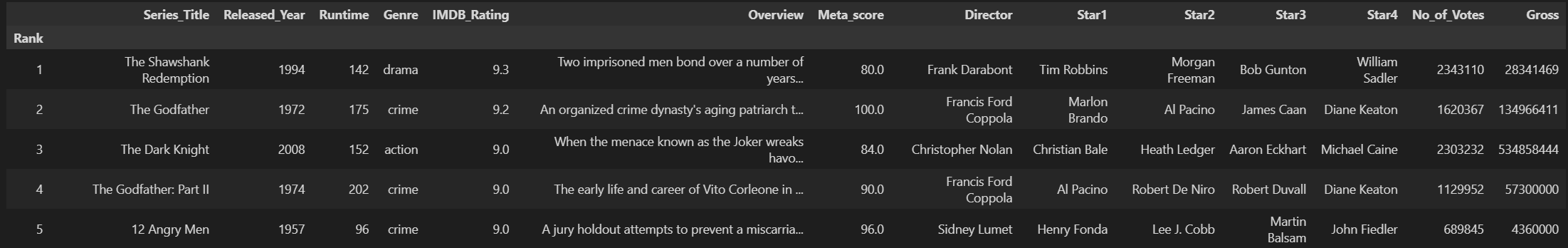
# 3. Adapting Movie Analysis Model for Visualization

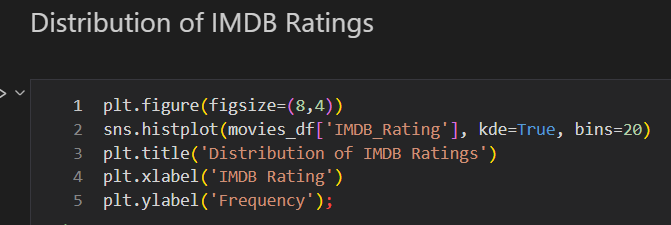
To continue showing how Matplotlib can be used to visualize data and provide a more comprehensive understanding of data, I used my previous project that demonstrated using Pandas by analyzing a movie data and converting it to a dataFrame for analysis. I first began by reading in the CSV of top 1000 movies, a dataset I obtained from Kaggle. 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 Figures 21-23.

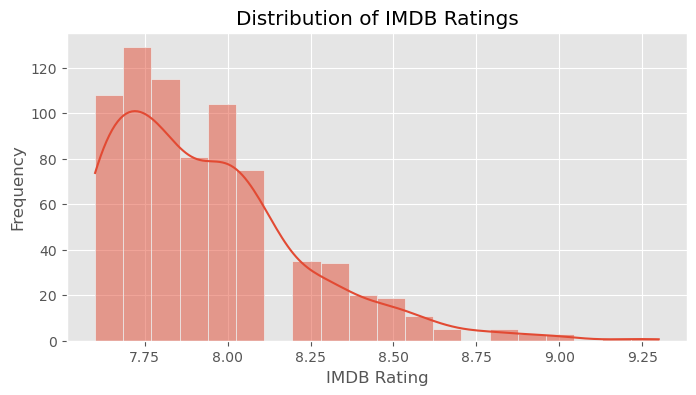
Figure 21: Movie Analysis Code (a)

Figure 22: Movie Analysis Code (b)

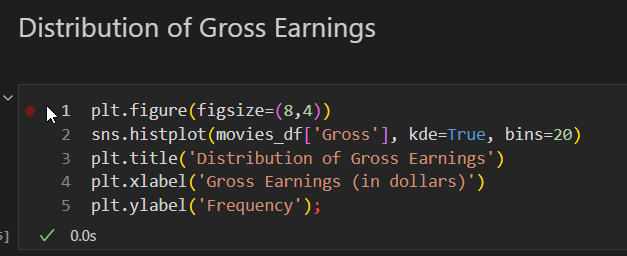
To begin to better understand the data, I plotted the IMDB\_Rating in a histogram to better understand the distribution of ratings. Here we see that the majority of top movies tend to have a rating of 7.6 to 8.20. It is rare to have a movie with a rating above a 9. Figures 24 and 25 shows the code and the results.

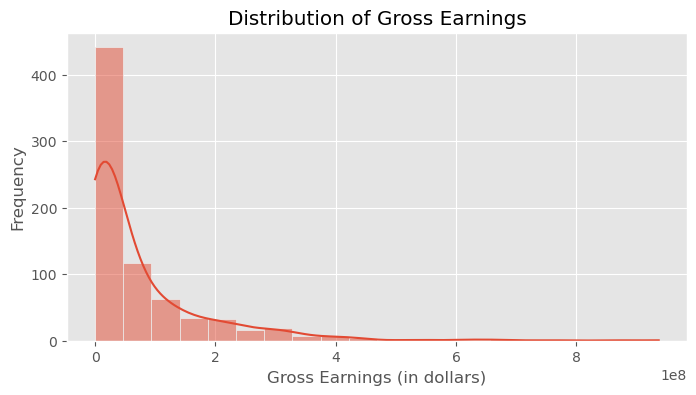
Figure 23: Movie Analysis Code (c)

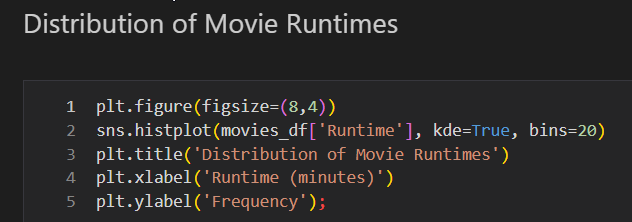
Figure 24: Distribution of IMDB Ratings Code

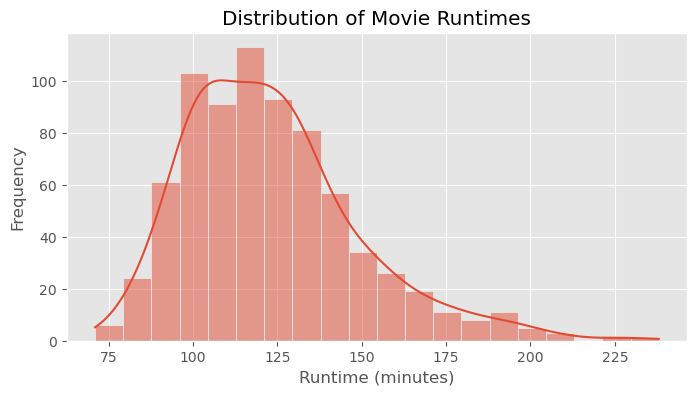
Figure 25: Distribution of IMDB Ratings Histogram Plot

Next I used a similar histogram plot to show the distribution of gross earnings, along with showing the distribution of runtimes. Figures 26-29 shows the code and output. The majority of the top movies tend to stick around a runtime of 100 to 125 minutes.

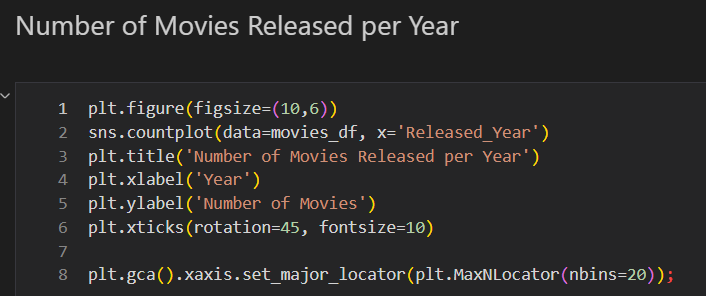
Figure 26: Distribution of Gross Earnings Code

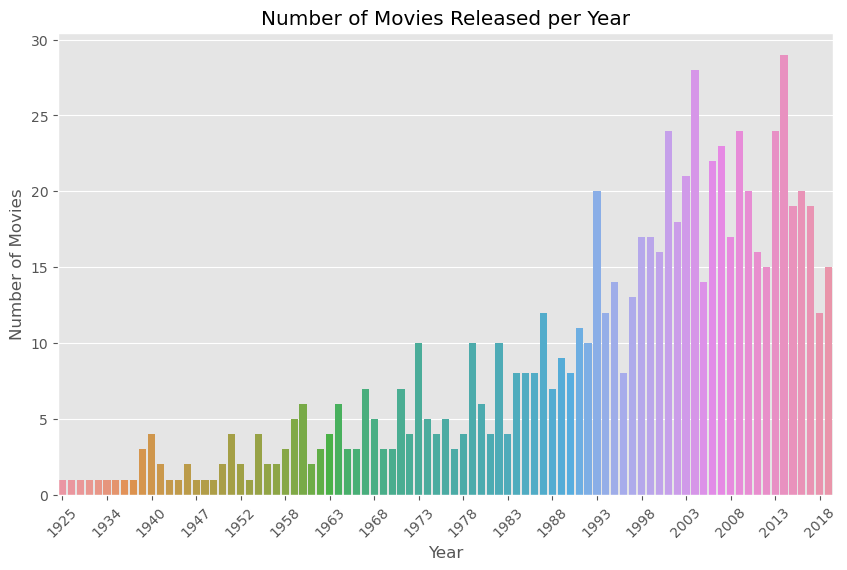
Figure 27: Distribution of Gross Earnings

Figure 28: Distribution of Movie Runtimes Code

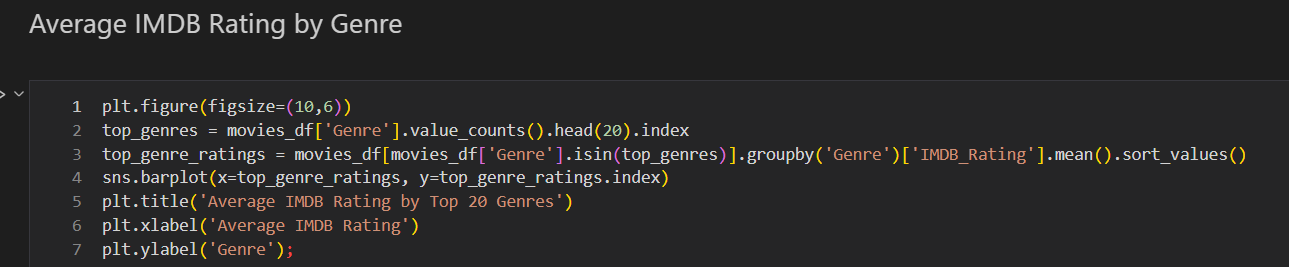
Figure 29: Distribution of Movie Runtimes

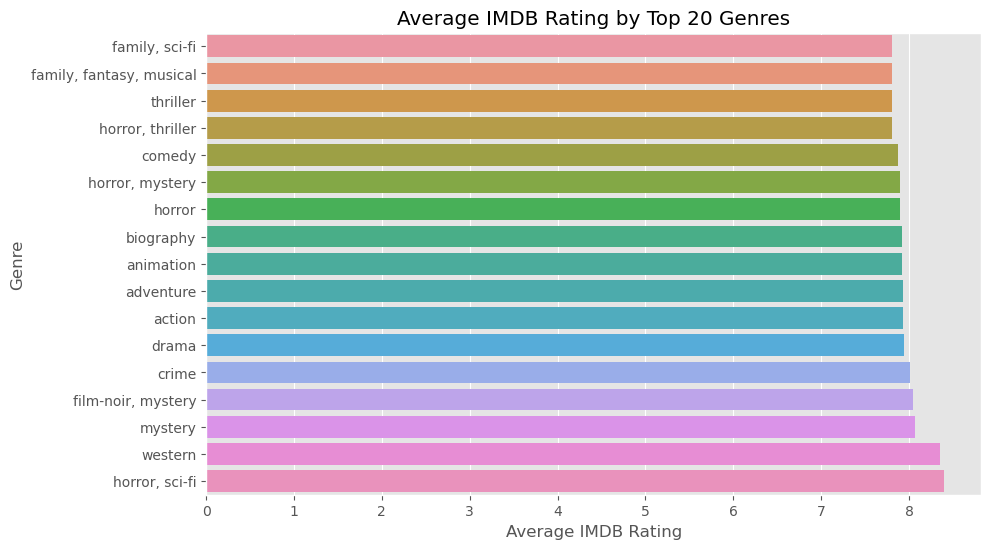
The next bar plot displays the number of movies that are considered the top movies for each year. The code processes the dataset to extract release years, handling missing and incorrect data, and the plots the counts using *`seaborn`*. This graph provides insight into trends and patterns in movie production over time. Figures 30 and 31 show the code and output. One issue I encountered was having the ticks not to overlap each other. So to solve this issue I customized the ticks by rotating them at an angle for display.

Figure 30: Number of Movies Released per Year Code

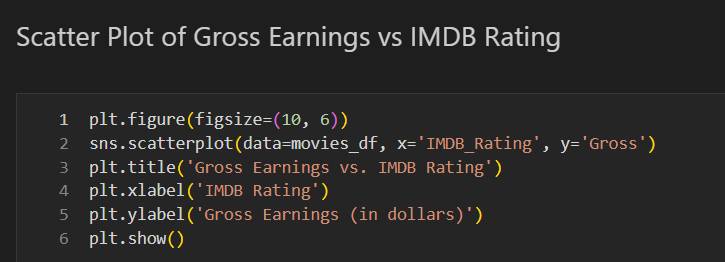
Figure 31: Number of Movies Released per Year Plot

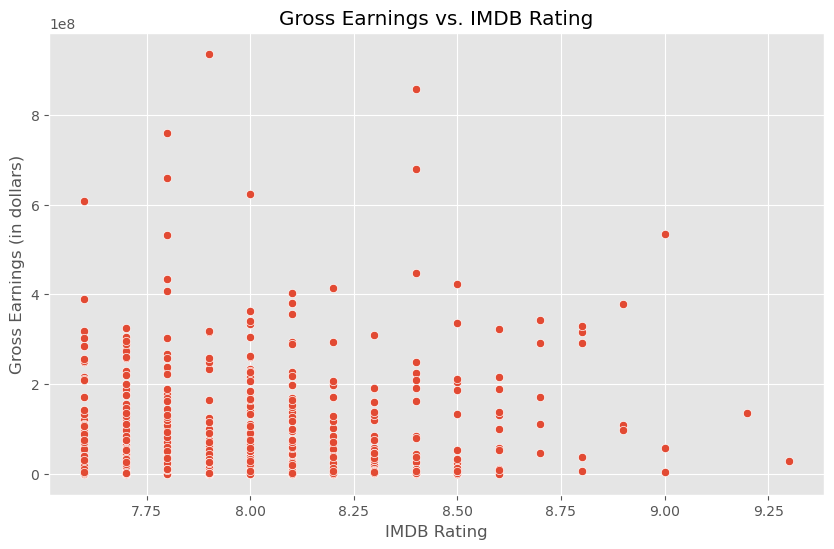
The next graph is another bar plot that shows the average IMDB ratings for the different genres. The code simplifies genre names using regex, groups the data by genre, and calculates the average rating for each. This visualization highlights differences in audience reception across various genres. Figures 32 and 33 show the code and output.

Figure 32: Average IMDB Rating by Genre Code

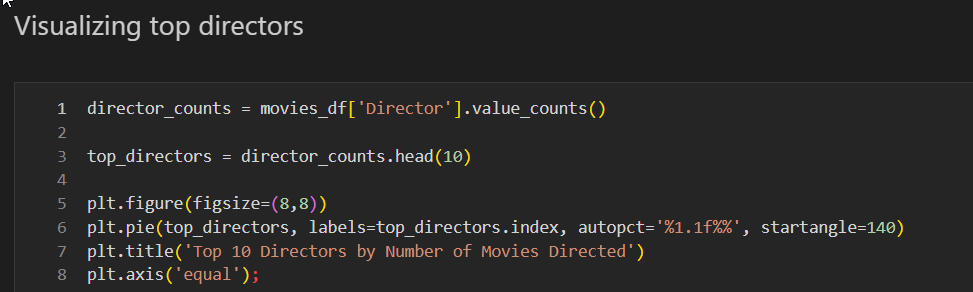
Figure 33: Average IMDB Rating by Genre Plot

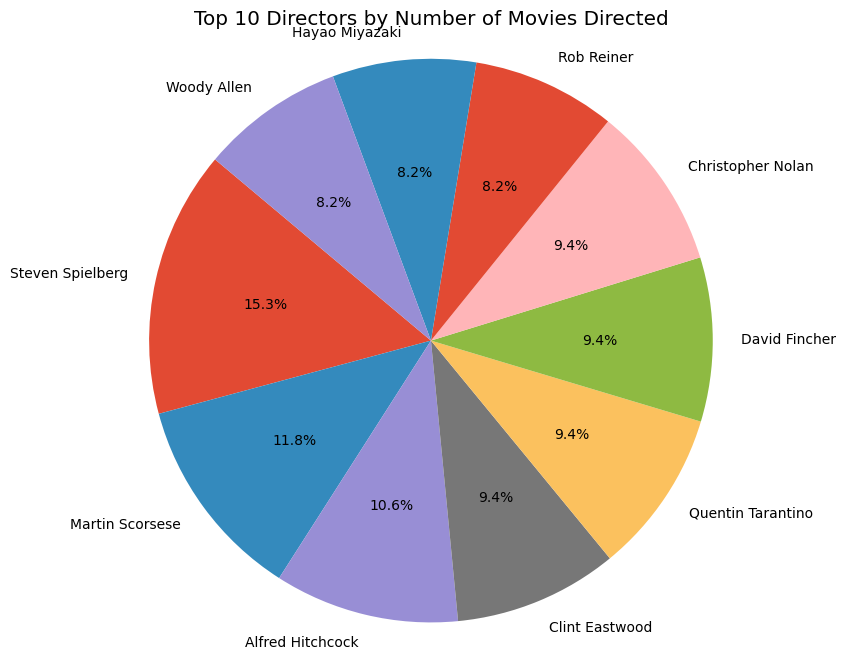
The next graph is a scatter plot to examine the relationship between gross earnings and IMDB ratings. The code plots these two variables, showing how movie earnings correlate with their ratings to identify potential patterns in the data. Figure 34 and 35 show the output and code.

Figure 34: Scatter Plot of Gross Earnings vs IMDB Rating Code

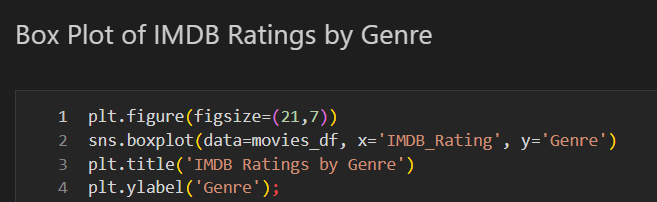
Figure 35: Scatter Plot of Gross Earnings vs IMDB Rating

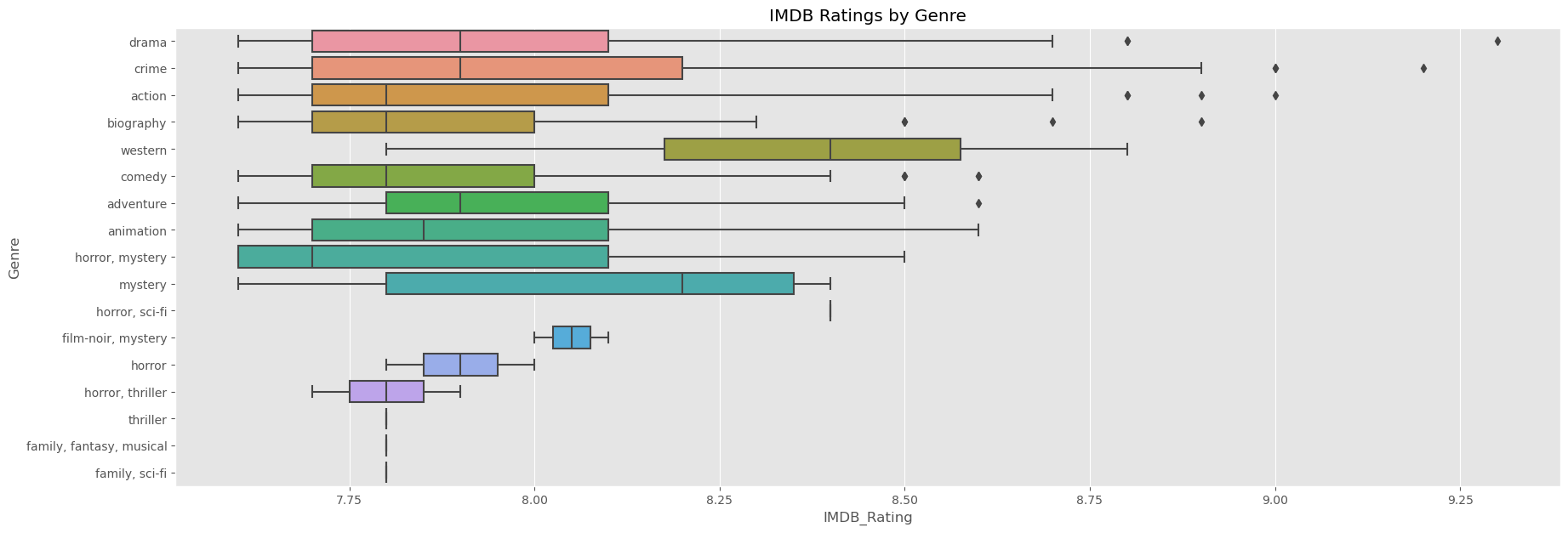
Next I used a pie chart to represent the top 10 directors by the number of movies directed. The code counts the occurrences of each director in the dataset and visualizes the top 10. This graph gives a view of which directors have the most significant presence in the top 1000 movies (Fig. 36-37).

Figure 36: Visualizing Top Directors Code

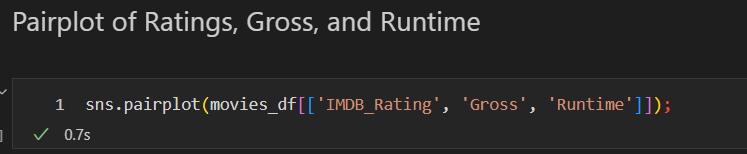
Figure 37: Visualizing Top Directors Pie Chart

The next graph shown below was a box plot constructed using *`seaborn`* to visualize the distribution of IMDB ratings across different movie genres. This graph provides insight into how ratings vary within each genre and to compare the central tendency and spread across genres. Figure 38 and 39 shows the output and code.

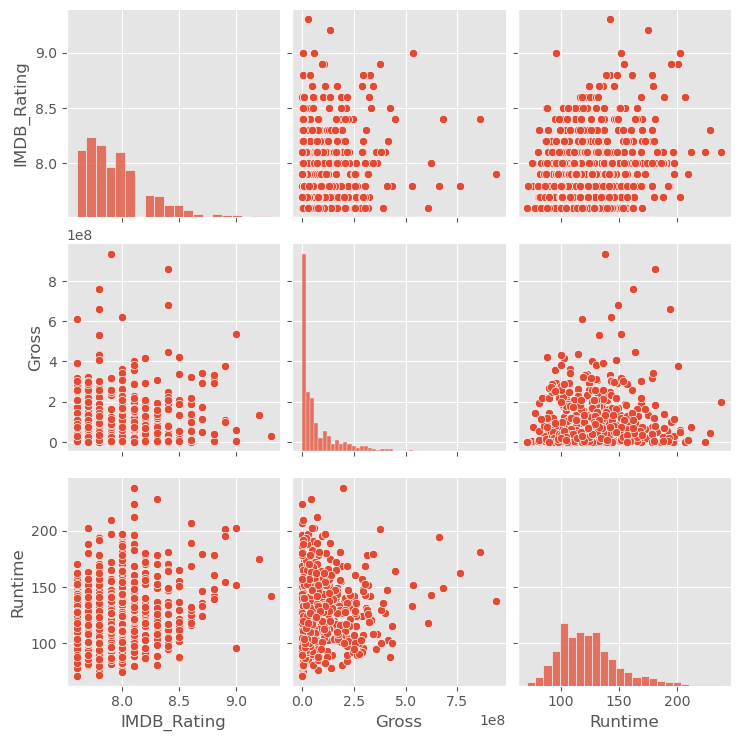
Figure 38: Box Plot of IMDB Ratings by Genre Code

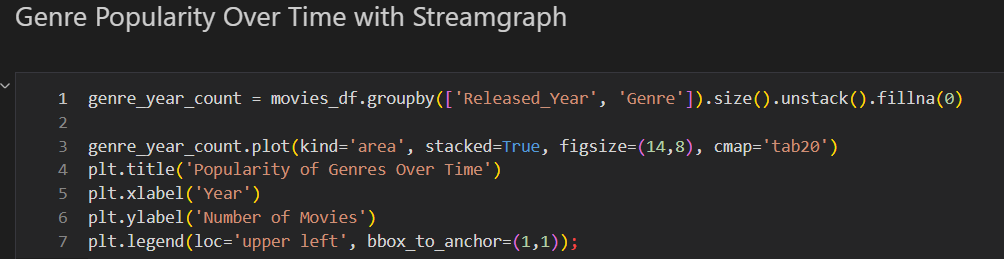
Figure 39: Box Plot of IMDB Ratings by Genre

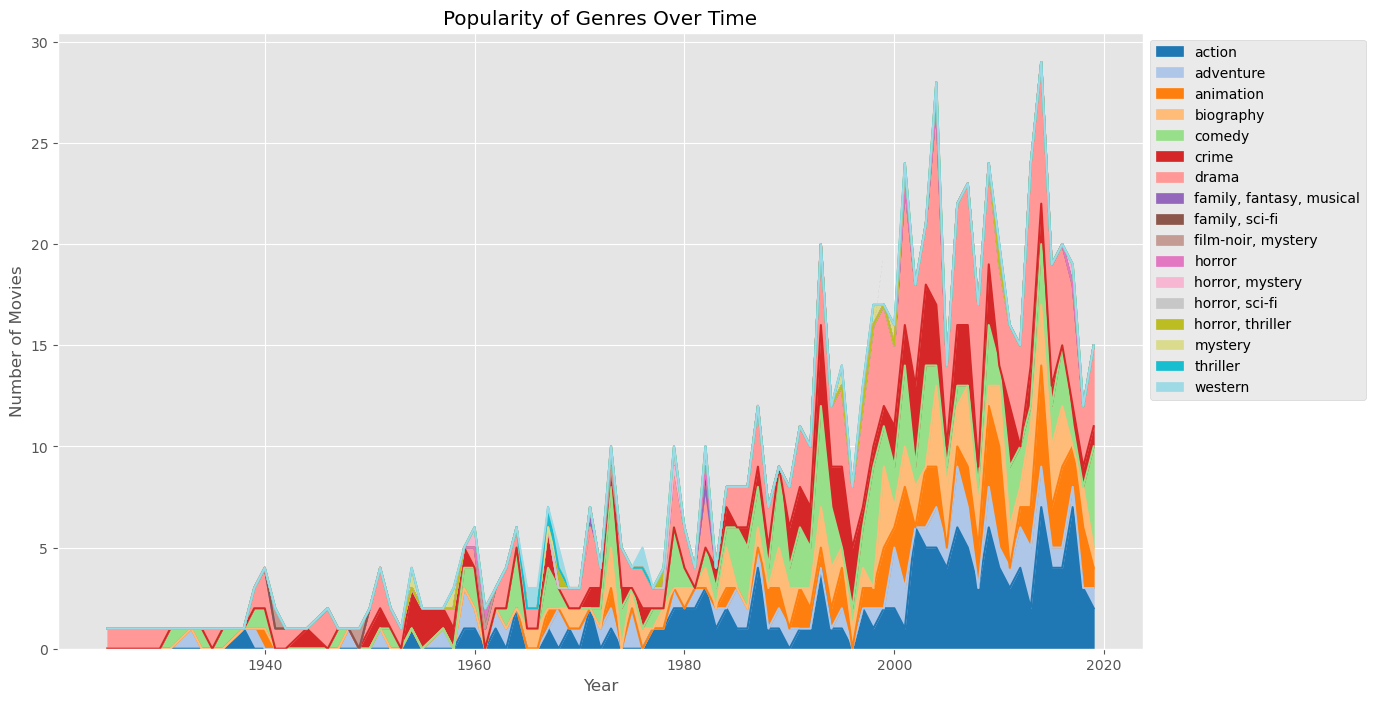
Using *`seaborn`* again, a pairplot was used to visualize and assess the relationship between the rating, gross earnings, and runtime. Figure 40 and 41 shows the output and code.

Figure 40: Pairplot of Ratings, Gross Earnings, and Runtime Code

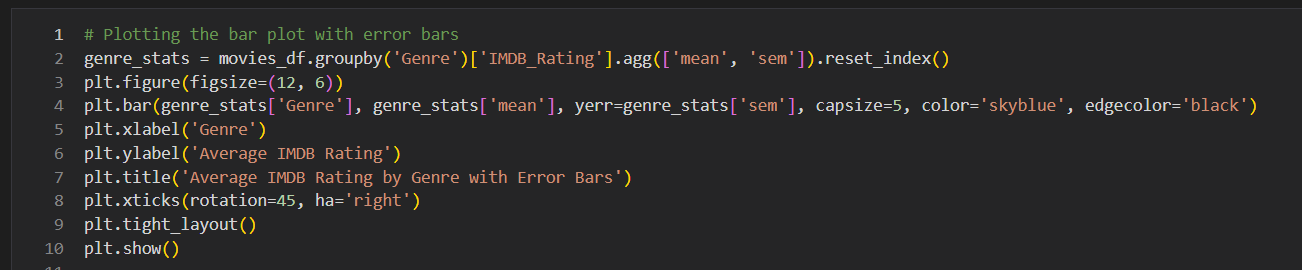
The next graph is an area plot that shows the popularity of different genres over time. The code groups the data by release year and genre, then plots the number of movies in each genre per year to reveal any trends in genre popularity over the years. Figure 42 and 43 show the code and output.

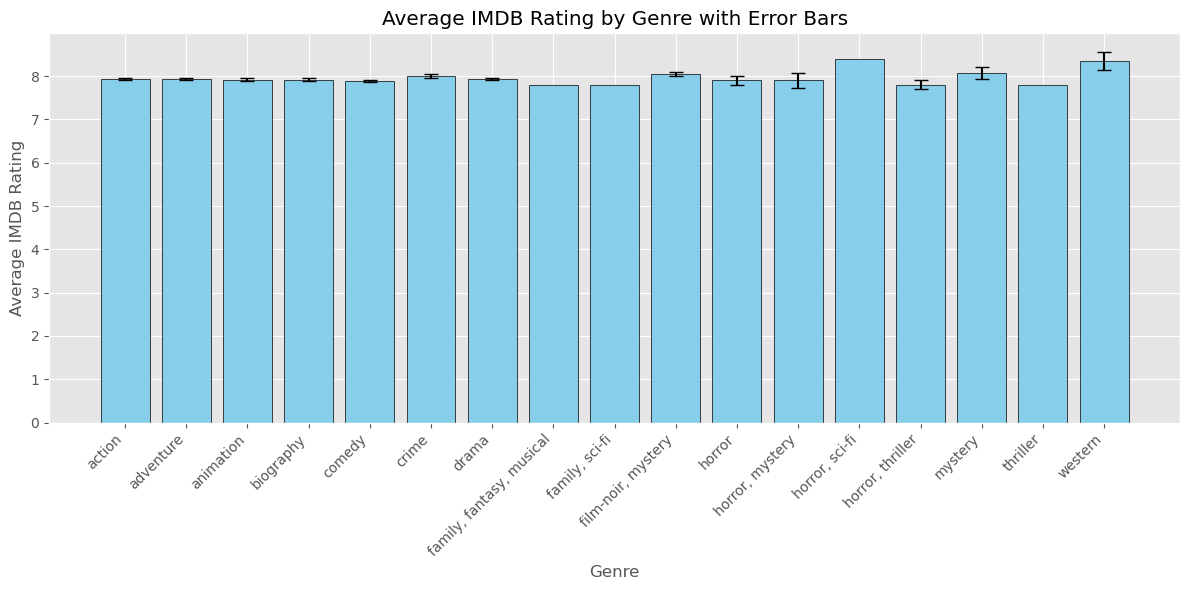
Figure 41: Pairplot of Ratings, Gross Earnings, and Runtime

Figure 42: Genre Popularity over Time with Area Plot Code

Figure 43: Genre Popularity over Time with Area Plot

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 44 and 45 shows the code and output.

Figure 44: Average IMDB Rating by Genre with Error Bars Code

Figure 45: Average IMDB Rating by Genre with Error Bars

# 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. Figure 46 and 47 shows the code and the output. I used the Basemap toolkit to plot the geographic data, which helps provide a spatial understanding of COVID-19’s impact. Figure 48 and 49 shows the code and output. Additionally, I constructed subplots using *`sns.barplot`* to visualize the average number of confirmed COVID-19 cases and deaths in the US. Figure 50 and 51 shows the code and the results.

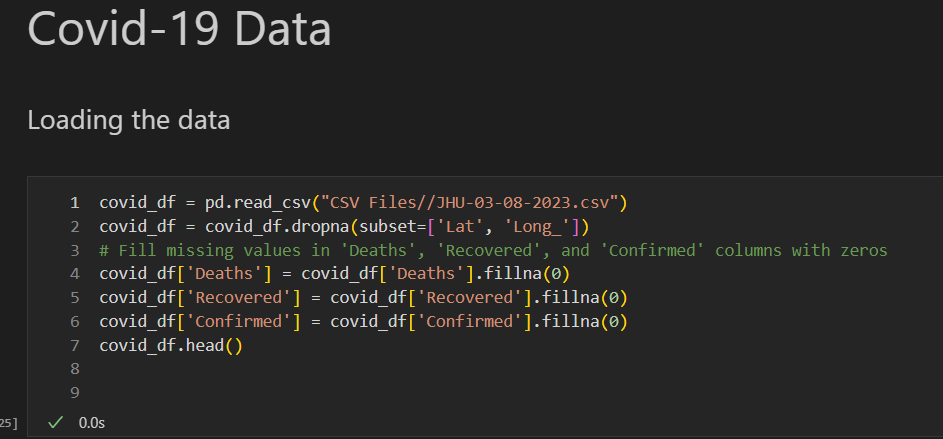
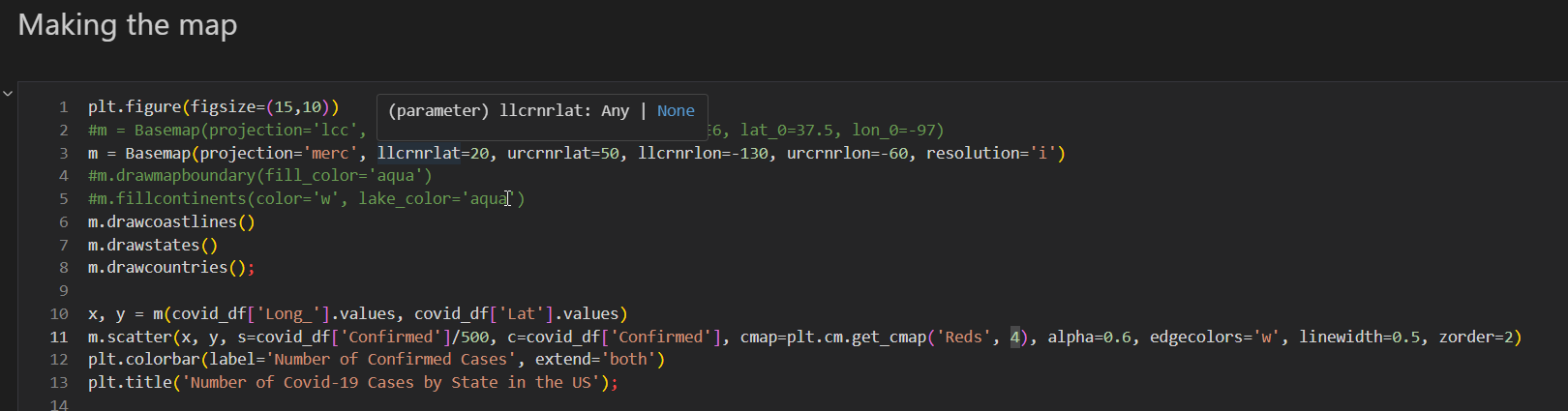
Figure 46: Importing and Cleaning COVID-19 Dataset Code

Figure 47: Importing and Cleaning COVID-19 Dataset Output

Figure 48: Visualizing COVID-19 Cases in the US with Basemap Code

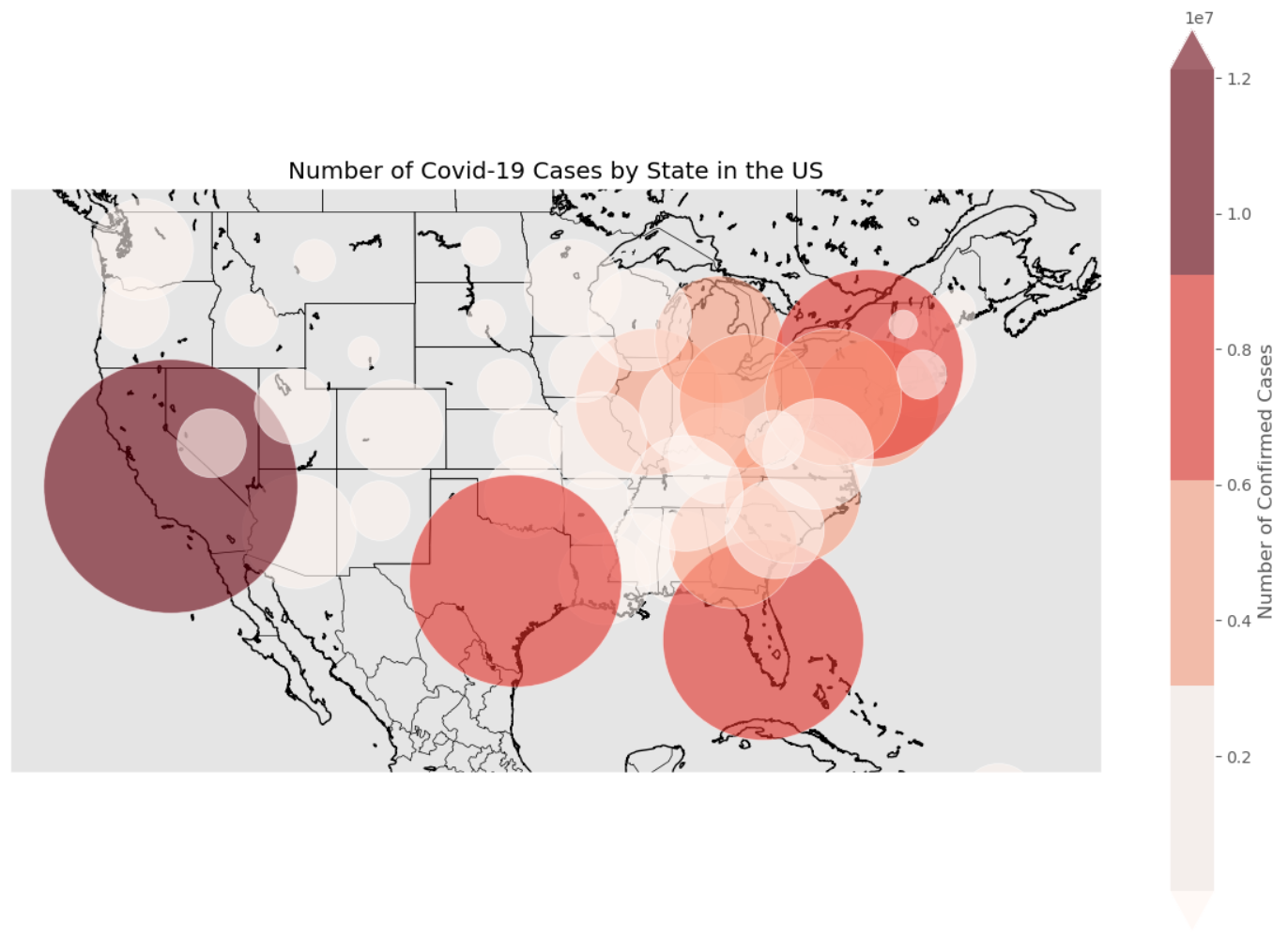
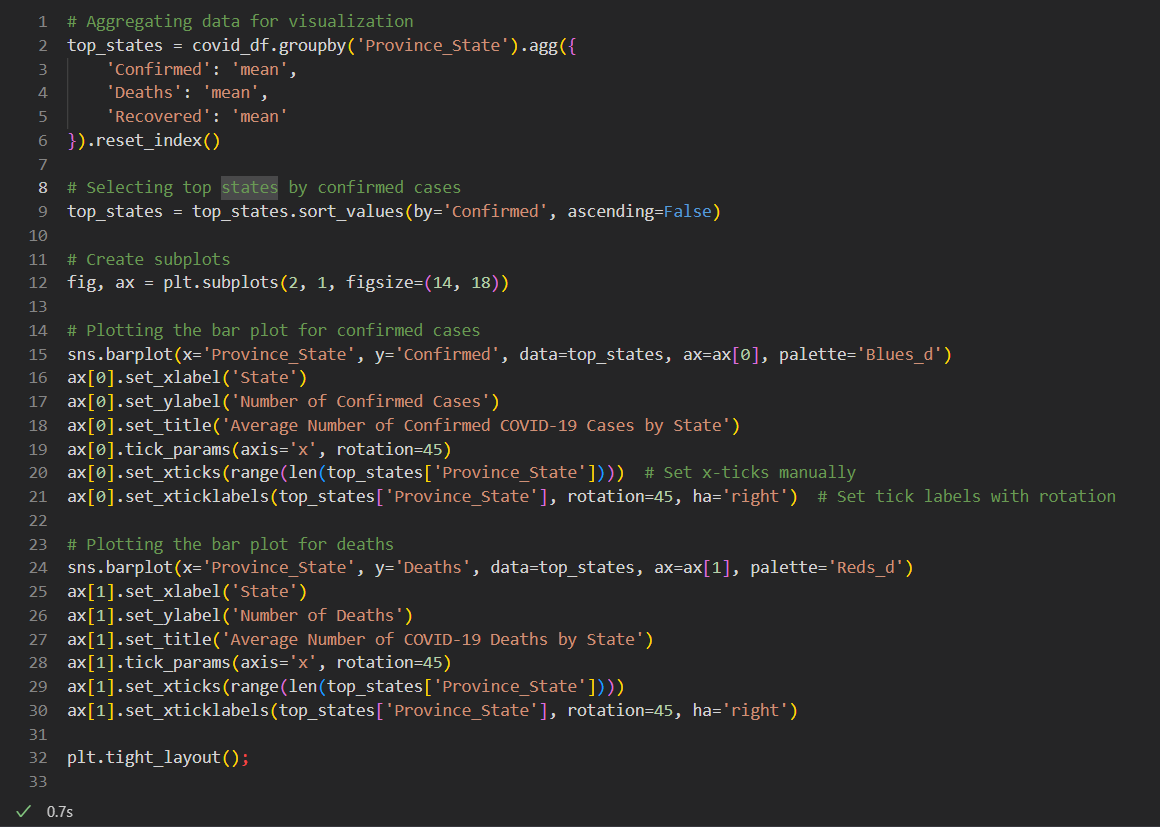
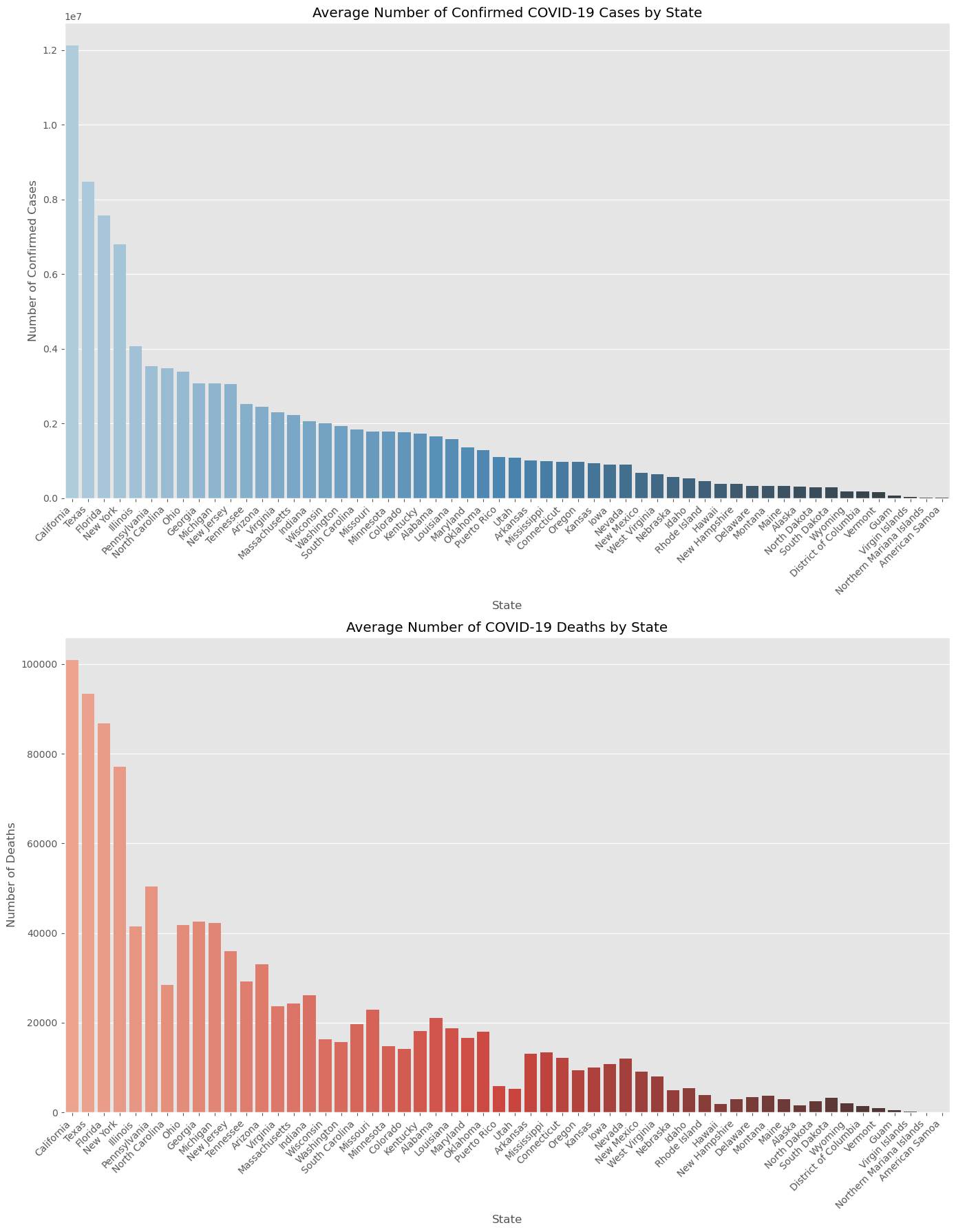
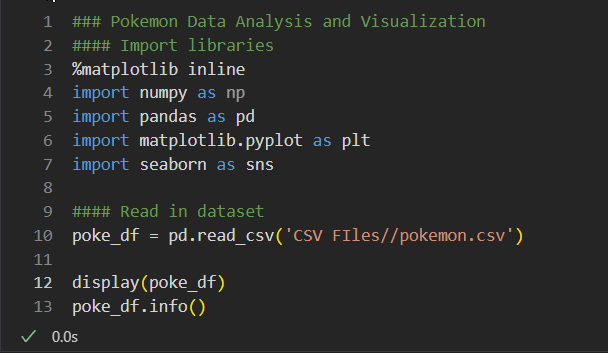


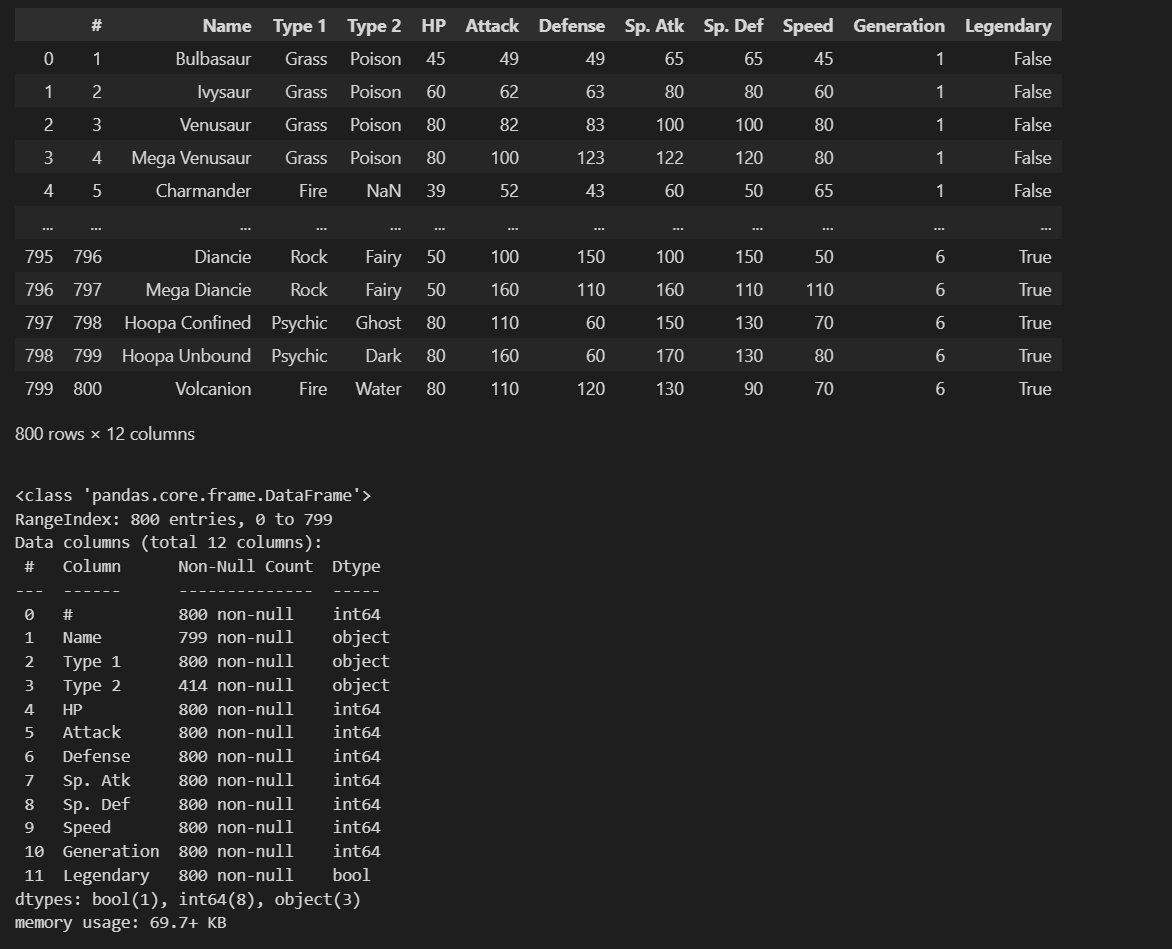
Figure 49: Visualizing COVID-19 Cases in the US with Basemap

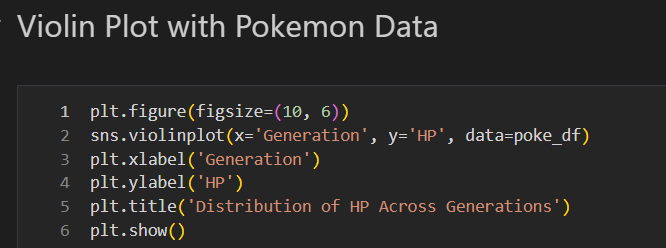
Figure 50: Subplots of Average Number of Confirmed COVID-19 Cases and Deaths by State Code

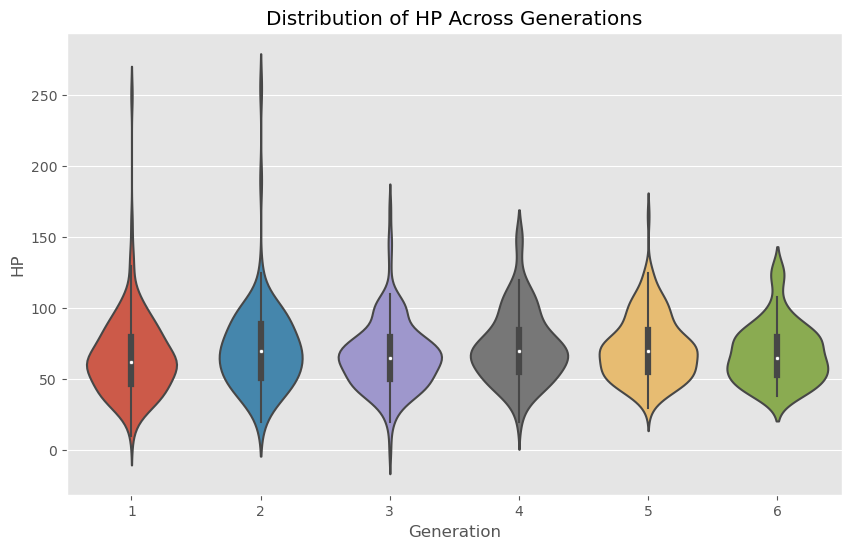
Figure 51: Subplots of Average Number of Confirmed COVID-19 Cases and Deaths by State

I also imported a Pokemon dataset from Kaggle that contains basic information such as the Pokemon name, Type, HP, Attack, and other stats (Fig 52-53). 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 54 and 55 shows the code and output.

Figure 52: Importing Pokemon dataset code

Figure 53: Output of Pokemon dataset

Figure 54: Violin Plot with Pokemon Data Code

Figure 55: Violin Plot with Pokemon Data

# 5. Conclusion

This report documents my journey in learning Matplotlib. Matplotlib is a versatile plotting library in Python. I explored concepts that include fundamental plotting techniques essential for visualizing data and customizing the titles, labels, legends and colors of charts and graphs to enhance clarity and aesthetics. Key concepts covered include creating various types of plots such as line plots, bar charts, scatter plots, histograms, and more. By using real data and previous projects, I was able to explore how to go about creating graphs and charts.

# 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. Johns Hopkins University Center for Systems Science and Engineering (2024). COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. GitHub. Retrieved Jun 14, 2024, from https://github.com/CSSEGISandData/COVID-19/tree/master/csse\_covid\_19\_data/csse\_covid\_19\_daily\_reports\_us
3. Terminus7. (2024). Pokemon Challenge dataset. Kaggle. Retrieved June 14, 2024, from https://www.kaggle.com/datasets/terminus7/pokemon-challenge?select=pokemon.csv
4. VanderPlas, J. (*2016*).  *Python Data Science Handbook*. O’Reilly Media. Retrieved from https://jakevdp.github.io/PythonDataScienceHandbook/index.html