Project Proposal: Genres of the Most Popular Books of All Time

Luong Tu Anh Bui and Eliza Azaro

March 2025

1 Introduction

For our project, we have explored the most popular novels of all time according to the website Goodreads, which since it's launch in 2007, has become "the world's largest site for readers and book recommendations" (Goodreads). We wanted to analyze how genres play a part in the books on this list. According to the Oxford Reference, the word genre in a literary context means "a grouping of texts related within the system of literature by sharing features of form and content" (Oxford University Press). Genres are an extremely effective method of categorizing books to help readers discover other books they'd find interesting based on things they've enjoyed in the past. While most people vary in their preffered genres, it does appear that certain categories of books are more likely to become popular and widespread. For example, according to Statistica, about 37% of children in the US have read at least one *Harry Potter* novel (Statistica). While there are several other fantasy series of similar popularily, you would hardly find a nonfiction book in similar standing. We have used a dataset from Kaggle called *Goodreads' Best Books Ever* and it contains a little under 50,000 books. The dataset is contained in a csv file where the first line specifies the values of each of the 23 different columns detailing various information about each book. However, for the purposes of our analysis we have chosen to only include data from 4 columns: Title, rating, genres and number of pages. **Our primary research goal is to analyze the genres among these most popular novels to determine what genres tend to be more popular among readers. Additionally, we would like to create an interactive environment for readers to find new book recommendations based on their preferences.**

Here is the link to our dataset: https://www.kaggle.com/datasets/arnabchaki/goodreads-best-books-ever

2 Computational Overview

To store the data for our program, we have written various attributes in our *BookData* class. The primary attribute is a Tree. The root itself doesn't contain any data, but each subtree of the root represents a tree with a root containing its title and a single subtree containing the other data we've chosen to include. We also use a dictionary called *genre_data* which maps every genre in the dataset to how many books are included within that genre. Since we wanted to be able to identify which genres had the greatest number of books in its category but dictionaries can't be sorted, we also created a list that stores tuples representing each genre and its number of books and these tuples are sorted in descending order by the number of books. To sort these genres into the *sorted_genres* list, we have implemented a selection sort algorithm with a helper method that searches the dictionary for the most popular genre and returns it to the original method so it can be appended to the list.

For loading data from the dataset into the attributes of our *BookData* class, it required a couple methods. The method *load_book_data* loaded the book data into our Tree described above. Since every book has multiple genres that it fits under, we wanted to include these genres in a list. However, since the data from our csv file is read as a string, we needed another method, $format\_genres$ to turn the string into a list that could be stored in the subtree of every book title. Lastly, we included a *load_genre_data* method which traversed through the subtrees of our Tree attribute, *book_data*, and created the dictionary described above to hold data on every genre and its number of books.

For the visual aspect of our project, we've included multiple graphs to view the data on genres. Using plotly, we have created both a bar graph and a tree map. The bar graph plots the genres on the x-axis and its corresponding number of books on the y-axis. The user is able to enter how many genres they'd like to see displayed on the bar graph so they can decide if they only want to look at the top couple genres, or see all of them. The tree map is another helpful visual representation of the genre data because it shows the top ten genres as squares where their size indicates how many books it corresponds to. In addition, each square contains ten books that are categorized under that category as examples. Every square can be selected and enlarged as well. The last visual piece we implemented was a weighted graph which shows a central node with edges to each of the ten most popular genres where each edge has a weight representing the number of books in that genre. The user interface is very interactive as the user can decide on the graphs they'd like to be displayed and we've also implemented a book recommendation system for the user as well. Users are able to input their book preferences including pages, genres, and ratings and we will use that information to calculate a similarity score that can then be used to recommend them books they'd probably enjoy. In addition, they can enter books they've already read so they don't show up in their list of recommended books.

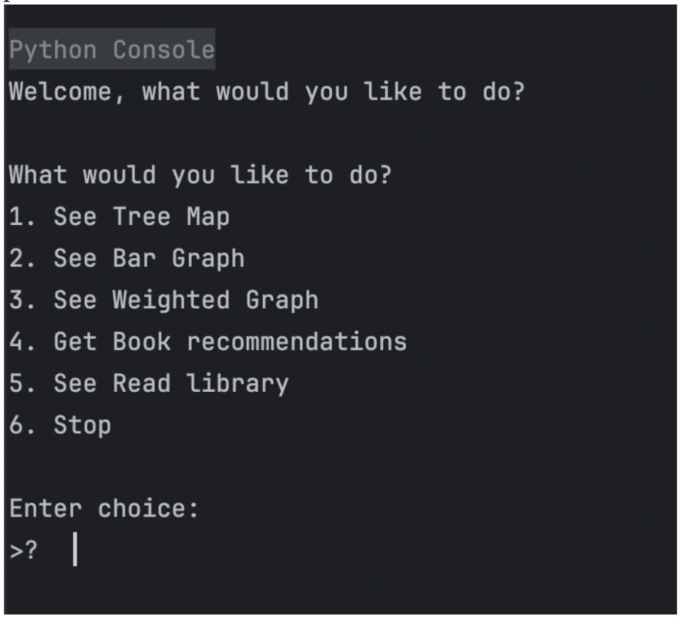
In order to create the graph visuals, we have used the Plotly and Networkx libraries. Plotly was used to create the bar graph and the tree map and we specifically used the *plotly.express* module which "contains functions that can create entire figures at once" (Plotly Graphing Libraries). The *bar*() method from this module created our bar graph using the dictionary we constructed which mapped the title "genres" to the list of sorted genres and "numbooks" to the list of corresponding number of books for each genre. Similarly, there is a *.treemap*() method in the module which we used to create the treemap. In addition, we used the methods *.update_traces*() and *.update_layout*() to format the colors of the tree and format the text inside each section. We then used networkx and matplotlib to create our weighted tree. We first used the *.Graph*() function in networkx to create a graph that we then added our nodes and weighted edges to. The method *.draw_networkx_edges*() took in the positions of all the nodes and the width of the edges to draw all the edges in. The methods *.get_edge_attributes*() and *.draw_networkx_edge_labels*() were then used to label each edge with the given weight and print those labels with the graph. Then we used *.draw_networkx_nodes*() and *.draw_networkx_labels*() to draw and format the nodes with their respective genres as labels. Lastly, matplotlib was used to finally display the graph.

3 Instructions

We have included a *requirements.txt* which names the libraries that must be installed for our program to work. We've included plotly and networkx, however those libraries were also included in the courses initial requirements file, so they are likely already installed for everyone. However, matplotlib and pandas will have to be installed as well.

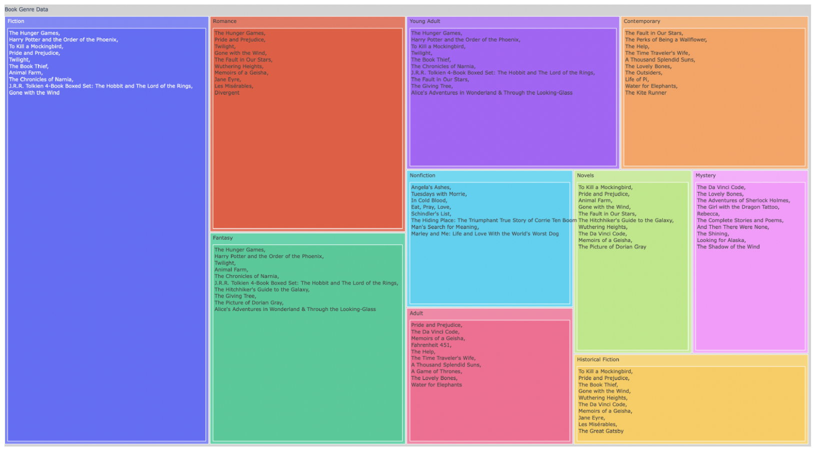
The zip file with our *main.py* file, code, and dataset is uploaded onto the repository.

When running the *main.py* file, you should expect to see a list of options for further steps. An image of this output is shown below:

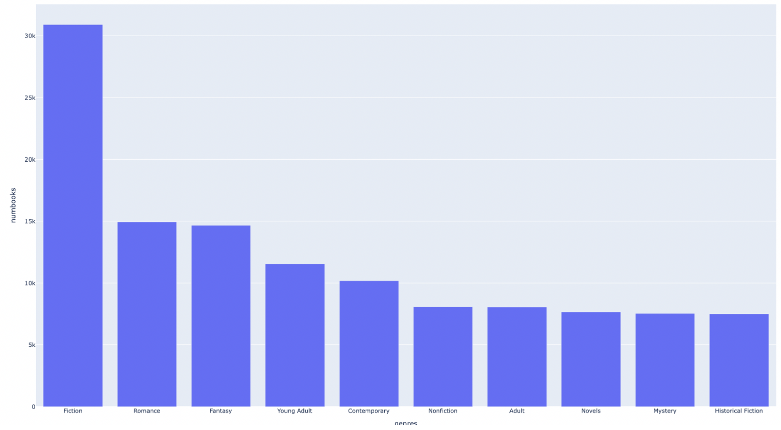


```
Python Console
Welcome, what would you like to do?

What would you like to do?
1. See Tree Map
2. See Bar Graph
3. See Weighted Graph
4. Get Book recommendations
5. See Read library
6. Stop

Enter choice:
>?
```

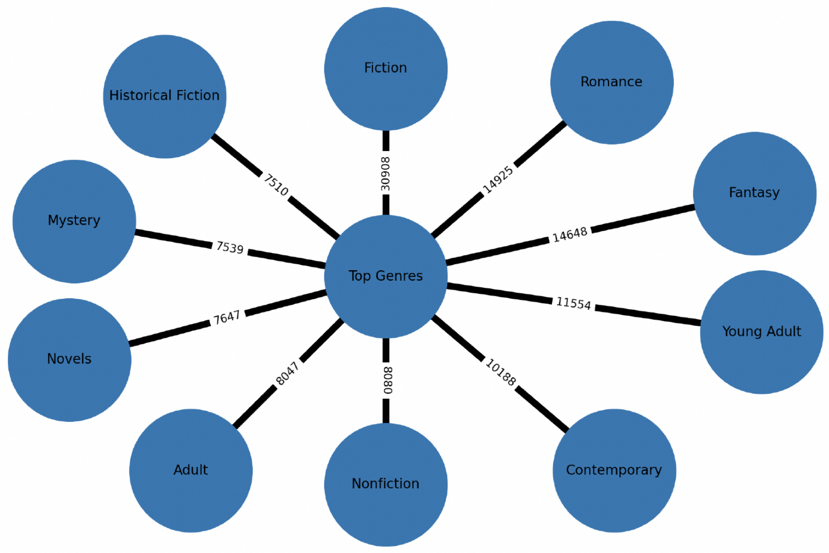
You may select any of the options given. If you select number one, you will see the tree map. That will look like this:



If you selected number two to see the bar graph, you will then be given the option to decide how many genres you'd like shown in the graph. Below is an example of a bar graph with 10 genres:



Number three will allow you to see the weighted graph, which is shown here:



You may also select to get book recommendations. If you select this option you will then be prompted to enter books that you've already read. This is an important step because no recommendations can be given if the user hasn't inputted at least one book. You can enter as many books as you'd like and when you have finished you can enter "stop" to move on to the next question. You will then be asked to enter a lowest rating bound so every book recommended to you will be above this threshold. Then you can enter the approximate number of pages you prefer.

You can then enter as many genres as you'd like and type "stop" when you're done. Lastly, you will enter the number of recommendations you'd like to receive and then you will be given a dictionary with that many books that have the highest similarity scores according to your data.

Another option is to see the books that you've put in as being read. Finally, you may type "stop" when you are done looking at the data and getting recommendations.

4 Changes to Project Plan

In our original proposal, we had planned on creating a decision tree like we'd done in our past exercises, and then use this tree to provide recommendations. Instead, we decided to use a tree that only split up the books based on their titles and then stored the data for each book below that book's title. For recommending books, we decided to implement a similarity score algorithm that found the best suited books for the user rather than using a decision tree.

We'd also originally planned on just displaying our data with a bar graph. However, we wanted to use other methods of visualizing our findings in addition to the bar graph, so we decided to also implement a tree map and weighted graph.

Lastly, there were certain parts of the dataset that we originally thought we would use, but in the ended chose not to. For example, we had planned on using information about the authors and book format as part of the recommendation algorithms. In the end, we chose to focus only on number of pages, rating, and genres for our recommendations.

5 Discussion

Our data and computations have revealed a lot about our initial project goal and research question. We went into the project wanting to learn more about which genres tended to be more popular among readers and by analyzing a dataset containing the best books according to Goodreads, we now have somewhat of an answer to this question.

We used three different visual representations to look at the data: a bar graph, a tree map, and a weighted graph. In all of these graphics, it was clear that the most popular genre was fiction and, more broadly, the top ten most popular genres were: fiction, romance, fantasy, young adult, contemporary, nonfiction, adult, novels, mystery, and historical fiction. The bar graph and tree map are very useful as they emphasize the scale at which fiction is much more popular than even the other 9 most popular genres because of how much larger its bar is in the bar graph and section is in the tree map. We can also see by reading the numbers on the edges of the weighted graph how great the difference is between fiction and every other genre.

A limitation of our data is that every book has a list of all the genres that it could be classified under. Therefore, it makes sense that a lot of the most popular genres are more broad and encompassing ones. However, in the marketing and printing of books, they are usually given one genre that best describes the book overall. It would be interesting to look at data that gives a single genre instead of multiple so we can consider more specific genres and the particular interests of readers across more niche categories of books. While there are more niche genres in the data that can be seen by increasing the number of books shown on the bar graph, they aren't really considered in our analysis because they are shadowed by those more popular genres that encompass them.

It would have been interesting to have had a better way of visualizing the tree we constructed and identify more popular genres from the imbalances of the tree. However, we ran into many difficulties with constructing tree and graph visuals and so we chose to make a graph that only represented the top ten most popular genres, rather than having something that displayed more options.

For further research, I think it would be interesting to look at genres as they relate to age. Do kids prefer fiction while adults prefer nonfiction? When looking at a lot of the most popular series among readers, they are often young adult fantasy series that readers of all ages enjoy. It would be interesting to look at what books are read across multiple generations versus ones that are primarily read by their marketed age group. Also, to give even better recommendations it would be helpful to utilize more of the data provided in our dataset such as book format, price, and authors. We could further narrow down the user's preferences and provide them with an even more tailored list of books for them to read.

6 References

- “Goodreads’ Best Books Ever.” Kaggle, 31 Mar. 2023, www.kaggle.com/datasets/arnabchaki/goodreads-best-books-ever.
- Goodreads. “About Goodreads.” Goodreads, www.goodreads.com/about/us. Accessed 31 Mar 2025.
- NetworkX Developers. “Weighted Graph.” NetworkX: Network Analysis in Python, https://networkx.org/documentation/stable/auto$_{e}xamples/drawing/plot_{w}eighted_{g}raph.htmlsphx-glr-auto-examples-drawing-plot-weighted-graph-py.Accessed 31 Mar$2025.

- Oxford University Press. “Genre: Quick Reference.” Oxford Reference, www.oxfordreference.com/display/10.1093/oi/authority.20110803095846831. Accessed 31 Mar 2025.
- Plotly Graphing Libraries. “Bar Charts in Python.” Plotly Graphing Libraries, https://plotly.com/python/bar-charts/. Accessed 31 Mar 2025.
- Plotly Graphing Libraries. “Plotly Express in Python.” Plotly Graphing Libraries, https://plotly.com/python/plotly-express/. Accessed 31 Mar 2025.
- Plotly Graphing Libraries. “Treemap Charts in Python.” Plotly Graphing Libraries, https://plotly.com/python/treemaps/. Accessed 31 Mar 2025.
- Statista. “Share of Children Who Have Read Harry Potter in the U.S. 2016, by Age.” Statista, 21 Mar. 2018, www.statista.com/statistics/689693/kids-read-harry-potter-books-by-age-group/: :text=Overall%2C%2037%20percent %20of%20kids,read%20a%20Harry%20Potter%20book.