Additions to Netflix Platform Over Time Rust Project

In my project, I aimed to use rust and different modules in order to process, analyze, and visualize the findings in a graph. I used Visual Studio Code to code the project, and used Graphviz in order to process the dot file I created from the code to produce a png image. The data comes from Kaggle, an online resource with many different datasets. The dataset is titled "Netflix Movies & Shows Dataset".

The five modules in my project are titled "analysis.rs", "date_added.rs", " graph_construction.rs", "parser.rs", and "visualization.rs". Also included is my main.rs file. In "analysis.rs", I am specifying that I am using a directed graph (Digraph) from the petgraph library. The 'analyze_graph' function has each node representing a year and each edge as having a weight, which would be the additions of shows for that year. It then prints the statistics I would like to find out from the graph in order to analyze it. These statistics include 'total_nodes', 'total_edges', 'total_additions', and 'avg_additions_per_year'. These calculate the total number of nodes, or years, that are in the graph, the total edges, or total transitions from one year to the next, the summation of all the weights of the edges, and the average number of additions to the platform per year. The function 'year_with_greatest_change' finds the year that has the greatest change, and the function 'print_additions_per_year' prints information similar to 'year_with_greatest_change'.

The 'date_added.rs' module was important for me to include in this project, as the original csv file contained many columns and I wanted to avoid the unnecessary columns getting in the way. The function 'remove_extra_columns' focuses on the date_added column in the original csv file titled "netflix_data.csv", which is the input file, and uses"c'leaned_netflix_data.csv" as the output file that I will be using for the rest of the code in the main.rs file. In th is function, I specified that the date_added column was at index 6, or the 7th column of the original file. Also in this module, I included a test that validates that the function correctly extracts only the date_added column. It compares what the file should look like to what it actually looks like as a result of the function above it.

The 'parser.rs' module parses and analyzes the dataset. The struct 'NetflixRecord' focuses on the 'date_added' field and uses Serde for deserialization. date_added can either hold a date from the file, or be 'None' if there is no date or it is invalid. The 'deserialize_optional_date' function parses the string for the date and is set to 'None' if it is missing.. 'parse_netflix_data' reads the csv file and uses the other methods to process the data. It then creates a HashMap that counts how many times a year occurs, and then counts and reports the number, and then sorts it. Afterwards, there is a testing modules called 'test_parse_netflix_data' that makes a temporary csv file with a sample of data that has both valid and invalid dates. Using 'parse_netflix_data' on it, it checks to see if it will count the valid dates and skip over the invalid ones. This was important since the information about the original dataset mentioned that there were exactly 10 missing dates. This module was extremely important as it is parsing the data I want to analyze and if it is incorrect, I will not be able to see any accurate data and any inferences would be inaccurate.

The 'graph_construction.rs' module utilizes 'YearGraph' that uses a directed graph from the petgraph crate. There are nodes that represent years (i32) and edges with weights, which represent the number of additions of movies/tv shows added. I used 'connect_all_years' so that each consecutive year was connected, which makes sense for what this project is about. There is a test for the construction of ‘YearGraph’ that makes sure the nodes are correctly connected, as makes sure the weights between two consecutive nodes are correct.

‘visualization.rs’ includes the function ‘visualize_graph’ that initializes a string in the DOT format and specifies the node shape, edge color, and graph layout direction (left to right). Nodes with over 1000 additions are red, and nodes with 0 additions are a diamond. In this project, it would be the last year since it would not have additions into the next year (2021 would not have a value of 0 since there are no additions listed in the data for 2022). Included are also labels for the edges that display the weights, which is the number of additions to the platform according to the dataset. After, the function returns a DOT string for representing the actual graph.

My ‘main.rs’ file uses the five modules. ‘input_file’ is defined as the ‘netflix_data.csv’ and ‘cleaned_file’ represents ‘cleaned_netflix_data.csv’. The ‘remove_extra_columns’ function is used to process the original CSV file to make it just the date_added column. Then, the data from that is parsed using ‘parse_netflix_data’ to count the auditions per year using HashMap. After, ‘YearGraph’ is created and the parsed data is added using ‘add_sorted_years’ that adds nodes and edges. Then ‘connect_all_years is used to make sure that all the years are connected. ‘analyze_graph’ and ‘print_additions_per_year’ are used to graph and print important statistics. ‘year_with_greatest_change’ is used to find the year with the greatest number of additions to the platform. Lastly, it is all made into a DOT format string and written to ‘netflix_graph.dot’ so that it can be visualized using Graphviz.

To run the project, using a terminal or command prompt, I can navigate to the root of the directory of the project, which is named ‘netflixgraph’, and run it by typing ‘cargo run’ to execute ‘main.rs’. Additionally, to run tests, I can type ‘cargo test’. (To visualize using Graphviz, I typed ‘% dot - Tpng -o netflix_graph.png netflix_graph.dot’ in order to take the DOT file created from the project and show it as a png file.)

The output is as follows (using cargo run):

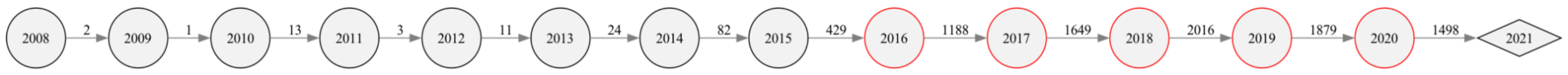
```
Records with missing dates: 10
Total years: 14
Total transitions: 13
Total additions: 8795
Average additions per year: 628.21
Additions per year:
Year 2008: 2
Year 2009: 1
Year 2010: 13
Year 2011: 3
Year 2012: 11
Year 2013: 24
Year 2014: 82
Year 2015: 429
Year 2016: 1188
Year 2017: 1649
Year 2018: 2016
Year 2019: 1879
Year 2020: 1498
Year 2021: 0
Greatest change in additions was between 2017 and 2018: 2016 additions
```

The test (using cargo test):

```
running 3 tests
test graph_construction::tests::test_graph_construction ... ok
test parser::tests::test_parse_netflix_data ... ok
test date_added::tests::test_remove_extra_columns ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

The graph (using Graphviz):



I can tell from this graph that Netflix's catalog was growing more rapidly starting at around 2015 and on, with their highest number of additions to their platform being in 2018 with 2016 additions.