Video Game Player Statistics for Game Development Planning

1. Introduction

The project addresses player count tracking and predictions over video games, which can be filtered to analyze data for specific games over time. The file reads a CSV which should have headers for date, active users, and title, and will reduce the CSV to a struct containing only those 3 as fields, eliminating other unnecessary CSV headers. Furthermore, the CSV file's rows can be filtered, to include only a specific game of interest. For my project, I decided to analyze the player count for Murder Mystery 2, a long-standing game that existed since 2014 on Roblox. This data not only can help analyze what affects player count, but also the demographic of certain games. The data set used in this analysis can be downloaded at https://www.kaggle.com/datasets/databitio/roblox-games-data, and is not included in the repository due to being over the 100 MB size limit.

2. Process

To filter the rows, I used some inspiration from a previous homework assignment where we were implementing DataFrames from the pandas module. It uses a closure and iterates over all the rows of the vector of structs, and only keeps those that match the input.

I realized that my CSV tracked player counts multiple times a day, at different hours, and if used with other CSV files, the date format might not be in yyyy-mm-dd. I decided to introduce the chrono crate to easily convert dates to the same format. In order to exclude the time from the date field, I decided to cut off the field's value at a space character. Converting the dates will parse the string, and will detect it if it is in mm/dd/yyyy format, and reformat it to yyyy-mm-dd. This was necessary for calculating the average of the day, in case there was instances where the same day entries in the CSV were in a different format. It also was necessary for sorting the days in order.

After that, I combined all the structs with identical date fields, and summed the player counts together. I used a HashMap to do this, rather than a struct, because I wanted to exclude the game title from the output, and keep it only to day and player averages. I create a new key if the day doesn't exist, and if it does, the player count is added. The keys have two values, one for the sum and one for the count, which the sum would be divided by to get the average, because I noticed the data was not gathered the same amount of times every day, which would make a total player count misleading.

I also decided to get averages for days of the week, and of the months in the dataset. I used the chrono crate to identify days of the week and month, and used a similar style to hourly_average_users to get the average; using a HashMap, creating keys for weekdays/months that did not exist, and adding the player count to the existing key if they did. I kept increasing the count by one for every loop iteration, which would be necessary to get the average, as the data did not include the full month of January or May, so total player counts would be misleading.

With all the data available, I wanted to make it somewhat visualizable, so that it could be graphed. Unfortunately, my main intention was to implement bar charts for weekly and monthly averages, and scatterplots with linear regression for the daily hourly averages, but I was not able to do this. I decided to continue with linear regression instead. Using a vector as input, I only used the average player count for the y axis, and incremented the x-axis by 1 for every data point, which was a day. I summed the y-values, and got the sum of the squares of n, the number of days. I then looked up the formula to calculate the slope and intercept, and implemented them. Then, the linear regression is used to create predictions. The prediction function will continue and stop before the player count will reach 0 on the next day. It only uses the intercept given from the linear regression function, and computes the player count for every day based on the slope. I used chrono to increment the days by 1 as well, so that the date can be identified, rather than just something like “day 1, day 2, …” In the event that the game’s player count is increasing or decreasing too slowly, or is stable, I also have the loop stop if 1 year has passed.

3. Results

The module can be used to analyze linear regression, predictions, and average player count statistics for any game. However, my main function will analyze this for MurderMystery2 as an example, and it is the game of interest. First noticing that the slope is negative, based on the linear regression output, I decided to write the main function to output days until the game is empty. Other average statistics, such as day of the week and month averages make sense with my logic, with the player count being higher than average on weekends. This implies that the majority of players for this game are children, despite it being labeled as a horror genre in the CSV.

4. Interpretations

The player count decreasing was predicting for the game to die in 346 days. While this is an exaggeration, as games have loyal player bases, it was somewhat accurate, as predicted player counts for the month of October were accurately reflecting the actual player counts at that time (this is historical data from 2022). However, this prediction does not imply anything about the player demographics. The first day recorded in the CSV file for Murder Mystery 2 is 2022-01-22, which is the day of an update in the game. Although the actual daily averages can be observed to be increasing at first, over the next 4 months of data collection, player counts decreased gradually.

This can be interpreted as the fault of the game's developer. The data can all be explained; an update is released, and the player count is stable/increases for a few days or weeks, then declines as players find the update "stale," or the game no longer gets updated. The linear regression data and predictions can be used by developers to know how often they should be releasing content updates, as a sign of successful games is a returning and loyal player base that never goes down. This also depends on how "addicting" the game is. A player count that decreases overtime shows that there is an issue with player retention, and there is little incentive or dopamine for players to constantly return, which is especially the case for Murder Mystery 2, which has had little changes since 2014.

5. Conclusions and future usage

The code can be used for other games, especially ones with increasing player counts for developers to see their success, and see what is behind sudden jumps in average daily player count, and also to see the games that perform poorly and decline in player counts sharply. This can be used by developers as a guideline on how to guide game development, by finding games that do well and following their example, and avoiding the paths that failed games took. Player counts do not magically decline or increase for no reason; they always are driven by external factors. The average player count by day of week and month can also be used to identify the player demographics; if certain months have spikes, such as summer, it can tell that the majority of players are children who live in countries with a school break in the summer. Consistent player counts could be telling of a more mature demographic, which is possible, as this code is not limited to Roblox tracking (and there are games with adult playerbases on Roblox); it only requires a proper CSV with the necessary fields.