Curtin College

In Association With Curtin University

Fundamentals of Programming (FOP1005)

Assignment 1

Due Wednesday, 8 December 2021 at 11:59 PM

Overview

You work as a Data Analyst for a cryptocurrency company that is wanting some insights on the Android app store market before it launches its new app onto the Google Play Store. Android is the world's most popular mobile phone operating system and has captured just over 71% of the total market. You have been asked to present an analysis of Google Play apps so that the team gets a good understanding of the different categories of apps, their ratings, and other metrics.

In this assignment, we will analyze the Android app market by comparing 10,842 apps on the Google Play Store across different categories. We will also use the user reviews and ratings and other attributes to draw comparisons between the apps.

The dataset you will use was scraped from the Google Play Store in September 2018 and was published on Kaggle (https://www.kaggle.com/lava18/google-play-store-apps/version/6). Please use the data files included on Moodle and not from this link since the Moodle files have been changed.

Specifications

The coding portion of this assignment is broken up into three main tasks. There is also a report section of the assignment. The three main coding tasks are:

1. Reading the app store's data stored in the apps.csv file.
2. Displaying a menu asking the user what filters they want to add to the app data.
3. Displaying the data matched by the set filters as a graph or by printing them to the terminal.

Reading The File (10%)

The apps.csv file contains all the details of the apps on Google Play as a comma-separated value (CSV) file. Before going on to the next tasks you'll need to read this file and store this data in a list or an array. Discuss which option you chose and why in the User Documentation.

There are 6 attributes that describe a given app in the apps.csv file, they are:

- **App:** The name of the app
- **Category:** The category of the app. Some examples are ART_AND_DESIGN, FINANCE, COMICS, BEAUTY etc.
- **Rating:** The current average rating (out of 5) of the app
- **Reviews:** The number of user reviews of the app
- **Installs:** The number of times the app was downloaded from the Google Play App Store
- **Price:** The price of the app in USD

Some of the apps are missing ratings. These apps have an empty value as a rating instead of an actual number. Handle this in a way you think best and discuss your solution in the User Documentation.

Displaying A Menu (5%)

Once you have read in the data from the file, your next task is to display a menu asking the user which of the following options they wish to perform:

1. **Add filters:** lets the user select a smaller portion of apps to print or graph
2. **Print app names:** prints the names of the filtered apps
3. **Graph apps:** graphs the filtered apps
4. **Exit:** exits the program

Adding Filters (20%)

This menu option allows the user to add filters to the data read in from the CSV file to help gain insights into the data. The user should have the ability to add filters on category, rating, reviews and installs. The user should also be able to add filters on multiple of these attributes. You don't need to have the option to remove a filter once it has been added, although you can if you would like to. This is how adding a filter to each attribute should work:

- **Category**: the user is shown the 33 different app categories available and can select one. Once a category is selected then only apps in that category should be used for future filter operations, for printing the app names and for plotting the data. These 33 categories should be retrieved from the data using code and should not be hardcoded. In other words, if the app categories changed in the file then should program should display the updated categories without changing the code.

- **Rating**: the user is given the option to choose a low rating and a high rating. Only apps with ratings between the low and high given should be used for future filter, printing and graphing operations. Ratings are always between 0 and 5 inclusive of both numbers (you can hardcode these boundaries).
- **Reviews**: the user can choose a minimum number of reviews and a maximum number of reviews. Only apps with reviews between the minimum and maximum should be used for future filter, printing and graphing operations. The number of reviews can never be less than zero. The maximum should be calculated based on the values in the file.
- **Installs**: this works in the same way as reviews but for installs.

For all the filtering operations above, you need to ensure the values entered by the user are valid. Invalid values should be shown an error message. When asking the user to enter a value you should display the lower and upper bounds that the value needs to be between.

Printing App Names (5%)

This option should print the app names remaining after the filters are set. Note that only the app names need to be printed and any of the other attributes.

Graphing The Results (20%)

This option should graph the app data remaining after the filters are set. When the user chooses this option they should be asked what data they want to plot on the x-axis. The options are categories, ratings, reviews, installs and prices. The y axis should always be the number of apps. The x-axis and y-axis should be labelled and the graph should have a title. It is up to you as to what type of graph you want to use to plot the data (bar, line, histogram etc). Bear in mind that some graph types are better suited for different types of data. Explain in your User Documentation what graph types you used and why.

Coding Standard (10%)

Your code submission must conform to coding standards emphasised in the lecture and practicals. Your code should also make use of multiple functions to reduce code duplication.

Your code should also follow the program layout structure on slide 52 of lecture 4. Note, your code won't look the same as the code on that slide but the order of the sections should be the same. Remember, consistency is key!

User Documentation And Report (30%)

You need to submit **User Documentation** as well as a **Report**, both of which should be in a doc/docx or pdf format.

User Documentation

Your User Documentation will be a minimum of 2 pages long and should include the following:

- An **overview** of each of your program's features.
- A **guide** on how to use your program.
- A **discussion** of your code, explaining the features you implemented, how you implemented them and why you implemented them the way you did.

The user documentation will be used and referred to by the programmers in the data analysis team at your ‘ workplace’ to get an understanding of your code.

Report

Your report will be a mini-paper that is 2-3 pages long and should follow the structure of a standard academic report. This is what will be used by the data analysis team at your work to make decisions about the app launch. As a result, this report should discuss the insights you found in the app data using your program. Such as, how competitive is the finance app category compared to other categories? What are the distribution of finance app ratings and installs like? What are some of the top apps in the finance category and so on? The required sections are:

- **Abstract:** Summarise your report's findings. Explain the purpose of the program, the questions you wanted it to answer and your outcomes/recommendations.
- **Background:** Discuss the purpose of the program and your choice of questions (see examples in the paragraph above).
- **Methodology:** Discuss how you went about gaining these insights from the program to answer your questions and why you chose to do it that way. Include commands, input files, and outputs – anything needed to reproduce your results.

- **Results:** Present the results of your analysis – include tables, plots and discussions.
- **Conclusion and Future Work:** Give conclusions and what further investigations could be done.
- **References**

(You can find examples on google but do not copy and paste. Just use the examples as a reference)

Submission Details

Submit electronically via Moodle. Make sure to submit early. You can submit multiple times – we will only mark the last attempt. Take care not to submit your last version late though. Read the submission instructions very carefully.

Submission Instructions

You should submit a single file, which should be zipped (.zip). The file must be named FOP_Assignment_<student id> where the <student id> is replaced by your student id ignoring the angle brackets. There should be no spaces in the file name; use underscores as shown.

The file must contain the following:

- **Your code**. This means all the files needed to run your program. That includes input files used as part of the assignment if that is required to run your program.
- **README** file, including short descriptions of all files and dependencies, and information on how to run the programs.
- **User Documentation and Report**, as described above.
- A signed and dated **Declaration of Originality**. This is available on Moodle and asks you to confirm that your work is your own. You can sign a hard copy and scan it in or you can fill in a soft copy and digitally sign it.

Make sure that your zip file contains what is required. Anything not included in your submission may not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct.

Requirements For Passing The Unit

Please note, as specified in the unit outline, it is necessary to have attempted the assignment in order to pass the unit. Section 2.5 of the Unit outline explains the weightage associated with the assignment, which is 30% of the overall unit. As a guide, you should be able to achieve around 50% for this assignment to pass the unit.

Plagiarism is a serious offence. This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au.

In the case of doubt, you may be asked to explain your code and the reason for choices that you have made as part of coding to the unit coordinator. A failure to adequately display knowledge required to have produced the code will most likely result in being formally accused of cheating.

Finally, be sure to secure your code. If someone else gets access to your code for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

Late Submissions

As specified in the unit outline, you must submit the assignment on the due date.

Acceptance of late submissions is not automatic and will require supporting documentation proving that the late submission was due to unexpected factors outside your control. See the unit outline for details as to the procedure for requesting that an assessment be accepted after the due date.

Also, note that IT-related issues are almost never a valid excuse.

In the event that you submit your assignment late and are deemed to have a valid excuse, you will be penalised 10% (that is, 10% out of 100%, not out of what you would have received) per calendar day that you are late, up to a maximum of seven (7) calendar days. Any work submitted after this time will not be marked and you will automatically fail the unit. Note that if you are granted an extension you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

Clarifications And Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced on the unit's Moodle page. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these by monitoring the Assignment section of the unit's Moodle page.