Anthony Stem
CS 124 | Fall 2021
Project 1 Report

Dataset

This dataset is a subset of the dataset maintained by the Extrasolar Planets Encyclopaedia as of February 28, 2018. It contains data for 3732 confirmed exoplanets from various sources and missions such as Kepler, CoRoT, K2, Kelt, OGLE, and more.[1] This dataset was found on Kaggle and originally contained 97 different columns of data; however, only five columns from the original dataset were used for this project. In addition, a new column called "row" was introduced to create a unique identifier for each row in the dataset. The six fields used in the project are as follows:

• *row* - a unique integer; indicates the row in the database.
• *name* - a string; indicates the name of the exoplanet.
• *mass* - a double; represents the mass of an exoplanet (in terms of Jovian mass J_M).
• *planetaryRadius* - a double; represents the radius of the exoplanet (in terms of Jovian radii J_R).
• *orbitalPeriod* - a double; represents the orbital period (in Earth days) of the exoplanet around its host star.
• *discoveryYear* - an integer; represents the year in which the exoplanet was discovered.

The entries in this dataset are ordered numerically by *row* in ascending order.

The primary reason I chose this dataset is because I have always been fascinated by space, especially exoplanets and extrasolar objects. I wanted a dataset that would allow me to analyze and explore the various properties and attributes of different exoplanets.

Calculation Function

The calculation function *showEstimatedSurfaceGravityValues()* computes an estimated surface gravity value, which is expressed as a product of the Earth's standard surface gravity $g = 9.8067\ m/s^2$, for each exoplanet. This function uses a specially modified form of the formula $g = \frac{Gm}{r^2}$, which uses the *pow()* function from <cmath>, to ensure the calculations are a product of $g = 9.8067\ m/s^2$.[2] This function uses each exoplanets' mass (*mass*) and radius (*planetaryRadius*) to make each estimation. In addition, the constant for G (*GRAVITATIONAL_CONSTANT*) in the formula, as well as the constants (*JUPITER_MASS*, *JUPITER_RADIUS*, and *EARTH_GRAVITY*) which are used to make the calculation be in terms of g are defined as follows:

[1] Original Source: http://exoplanet.eu/
Kaggle: https://www.kaggle.com/eduardowoj/exoplanets-database
[2] <cmath>: https://www.cplusplus.com/reference/cmath/

- *GRAVITATIONAL_CONSTANT* = 6.6741×10^{-11}
- *JUPITER_MASS* = 6.641×10^{-11}
- *JUPITER_RADIUS* = 7.1492×10^{7}
- *EARTH_GRAVITY* = 9.8067

```
double GRAVITATIONAL_CONSTANT = 6.6741e-11;
double JUPITER_MASS = 1.8981e27;
double JUPITER_RADIUS = 7.1492e7;
double EARTH_GRAVITY = 9.8067;
```

In addition to computing the estimated surface gravity for each exoplanet, the function also keeps track of the number of exoplanets with sufficient and insufficient data required to make the calculations, as well as the total number of entries attempted. The function then displays a table with the estimated surface gravity values for the planets where an estimation could be made. At the end of a table, the counts for the number of successful and unsuccessful entries, as well as the total number of entries is displayed as shown below:

```
Name                         Estimated Surface Gravity (g)
------------------------------------------------------
1RXS 1609 b                          12.2435 g
2M 0122-24 b                          50.548 g
------------------------------------------------------
684 Entries Successful.
3048 Entries Couldn't Not Be Computed.
3732 Total Entries.
------------------------------------------------------
```

Note: The actual output is too large to be properly displayed on this document; therefore, a snippet of the top and bottom combined is used to represent the output of the function.

Validation and Testing

How can you demonstrate in your code that all your class methods function correctly?

To test whether the Exoplanet class and its methods function correctly, I created the *testExoplanetClass()* method. This method uses assertions to test object instantiation with both default and alternative constructors, as well as the getter methods, setter methods. If any test fails, one of the assertions would cause the program to exit, and an “Assertion failed!” message would be displayed on the console along with the expression that failed the test. Otherwise, if the class pasts all the tests, the message “Exoplanet Class Test Successful!” is printed to the console.

How can you demonstrate in your code that your program successfully reads and stores 1000+ objects from your data file?

To test whether the program successfully reads and stores 1000+ objects from the data file, I created the *verifyFileInput()* method. First, this method checks to make sure that the row values are set correctly for every row. If there is an improperly set row, the method prints out that row value in an error message. This method also displays the number of exoplanets stored in the vector (this number should be 3732), as well as the row number of the first and last exoplanets stored in the vector to make sure there is the correct number of entries. The following example shows the output if there is an error in row setting:

```
3733 entries read from the file.
1 is the first row read from the file saved into the vector.
3732 is the last row read from the file saved into the vector.
Error: Improper row set at entry 3733.
```

Otherwise, the output is the following:

```
3732 entries read from the file.
1 is the first row read from the file saved into the vector.
3732 is the last row read from the file saved into the vector.
```

How can you demonstrate in your code that your calculation function works correctly?

To check the functionality of this function, I verified the results of my function by performing independent calculations using Excel and comparing the results. All of the surface gravity calculations made for each exoplanet are in agreement, as well as the number of exoplanets that could (684 entries) and could not (3048 entries) have a calculation performed on them. In addition, the total number of exoplanets passed into the function matches the total number of entries in the dataset.