Ex Mente

Python Assignment

Data Analysis Python Package

Authors:
R. Hayman

15/02/2024

Contents

1 Introduction

This project involves the design of a Python package module that can be installed and imported for use in other projects. The Python package module is focused on the processing and analytics of the data frame class type of Pandas, a statistical Python module.

Besides the data analysis module being a more encapsulating module for important statistical and plotting functions, for the Pandas [1] data frame class, it also brings with it some extra technical advantages like multi-threading function execution with multiple data frames, as well as file syncing with the implementation of rsync.

The developed module were also used with test data to demonstrate it's functionality and advantages. Lastly, a version control system has been setup for the data analysis module on Github [2] to ease the development and improvements of the module in different categories.

2 Implementation

2.1 Problem Statement Overview

The functions of the data analysis package can be summarized as follow:

- The user will specify a csv file path to have the file imported as a Pandas dataframe.
- The user will give a dataframe and specify a file path to export the dataframe to a csv file..
- The user will give a csv file/dataframe to have statistics being calculated for every column. The data analysis package will have to detect by itself whether a particular column is numerical or categorical before calculating the appropriate statistics.
- The user will give a csv file/dataframe with the column(s) to be graphed by the data analysis package. The data analysis package will by itself have to detect the appropriate graph to plot, based on the column type(s).

The problem statement is visually represented in Figure 1:

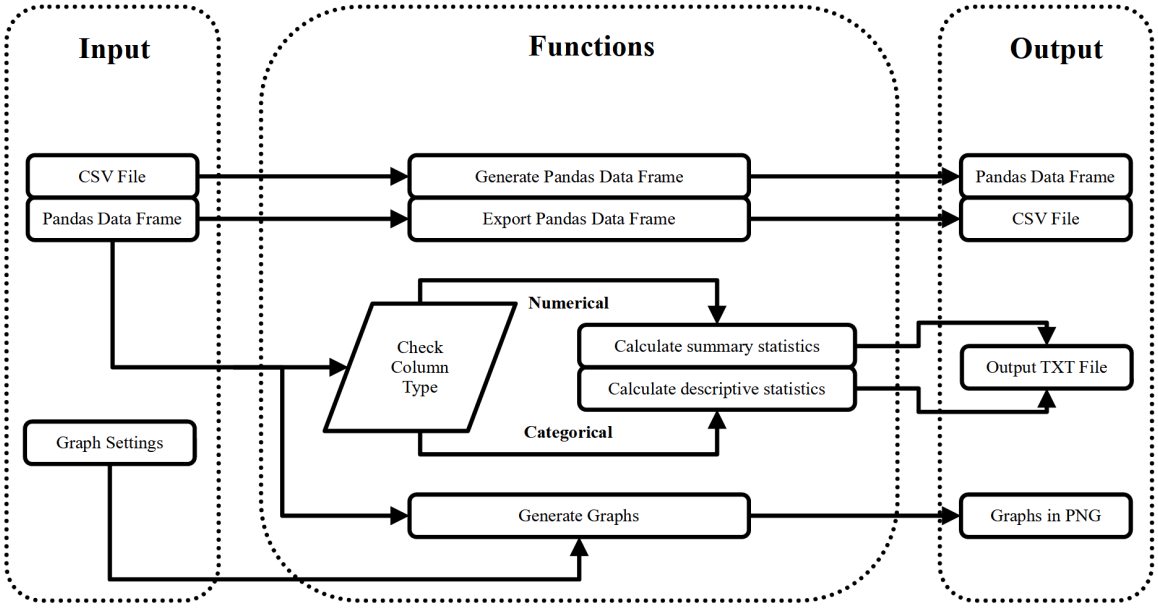

Figure 1: Problem statement of data analysis package

2.2 Implementation Overview

The data analysis package consists of 2 class modules, a Dataframe class and a Data-Analysis class. The user just needs to import the data analysis class as the data frame class is imported by the data analysis module.

2.2.1 Dataframe Class

The Dataframe class was developed as a means to encapsulate all the variables and functions required by the problem statement, in Subsection 1, into a single data frame blueprint (class) that can be made a object.

2.2.2 DataAnalysis Class

The data analysis class was developed as for the purpose of first, storing and handling multiple data frame objects in a multi-thread manner and secondly to control non-dataframe operations/ functionality.

2.3 Multi-threading Overview

Certain selected functions of the data analysis class will utilize the multi-thread functionality. With these functions, the user can specify the amount of threads that should be active at any one time. Each dataframe that has been created's desired function will be run in parallel to the amount of threads specified by the user.

The main thread of the program will constantly run in a loop, checking if any of the threads is unused and then load the thread with the next data frame function until all date frames desired function has been called. The user will get boolean flag return to indicate which of the data frame threads has failed.

A visual representation of the multi-threading system can be seen in Figure 2:

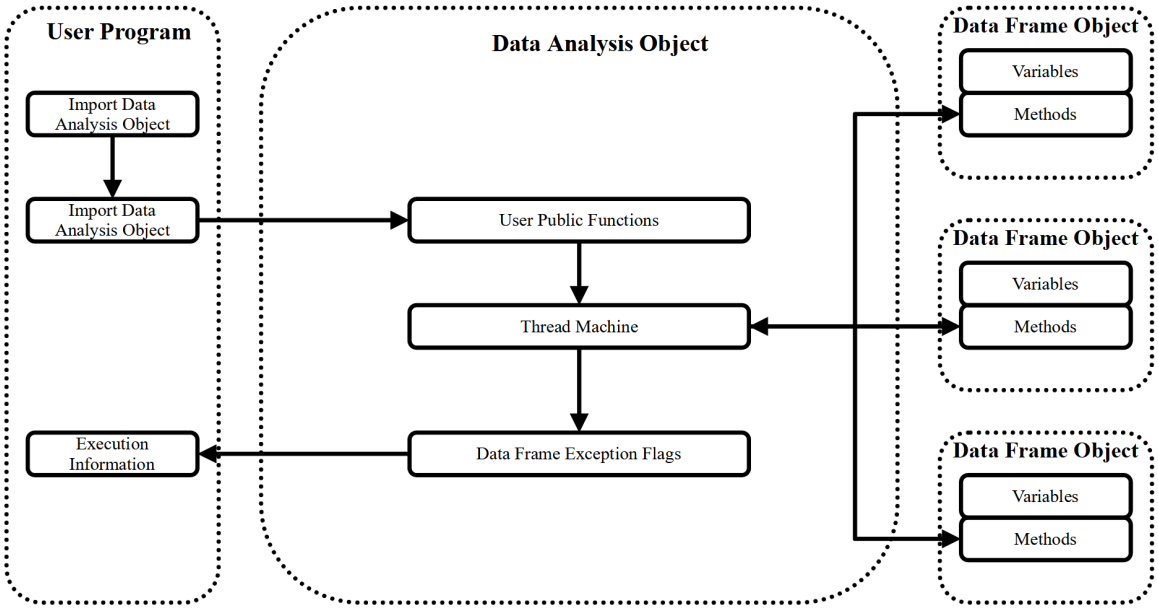

Figure 2: Multi-thread implementation

3 Challenges

With the development of the data analysis Python module, some challenges were met. The 3 most prominent challenges were learning the intricacies of the Pandas module, multi-threaded function implementation with multiple data frames and lastly the implementation of rsync file syncing.

3.1 Pandas Module

The intricacies of the Pandas module were pretty new and had to be learned, especially the CSV importing and data accessing format of pandas data frames. It was learned that to access pandas data frames data, it was better to use the column text keys, rather than to use traditional numeral indexes as with arrays/lists

3.2 Multi-Thread Function Implementation

Multi-threaded implementation of functions will always be the more challenging parts of a project due to special care having to be taken on locking and unlocking different resources for each thread to use, in order to prevent potential problematic occurrences.

Examples of such occurrences are thread race condition and deadlocks. Thread race conditions occur when multiple threads are writing to the same memory location without there being any organisation/syncing between threads. Thus the final result will be reliant on the finishing conditions of the threads.

Deadlocks is another problematic occurrence that can happen in multi-thread programming. This happens when for example, one thread locks certain resources that another threads needs while at the same time the other thread locks resources that the first thread needs. The result is that the two threads end up waiting for each other to unlock each others required resources, creating an endless wait cycle.

Lastly, it should also be mentioned that debugging multi-thread programs is more challenging due to the parallel nature of the program making stepping through code more of a choir.

4 Version Control

4.1 Overview

In order to ensure that development on the Data Analysis is as progressive as possible, the Git version control system has been set in place to divide the project in clear logical development areas and to medicate potential regressions that might occur as development continues.

For this project it has been decided on, to have 3 main branches that branch of from the master branch. The 3 main branches, being development, bug-fix and documentation, each has their respective purpose as explained in sub-subsection 4.1.1, 4.1.2 and 4.1.3, but all 3 main branches are permanent branches that will still be kept after their changes has been merged with the master branch.

Sub-branches can be created from any of the 3 main branches but these branches will only be temporally, meaning that they will be deleted after their changes has been merged with the respective main branch. A visual representation of the Git repository branch structure can be seen in Figure 3:

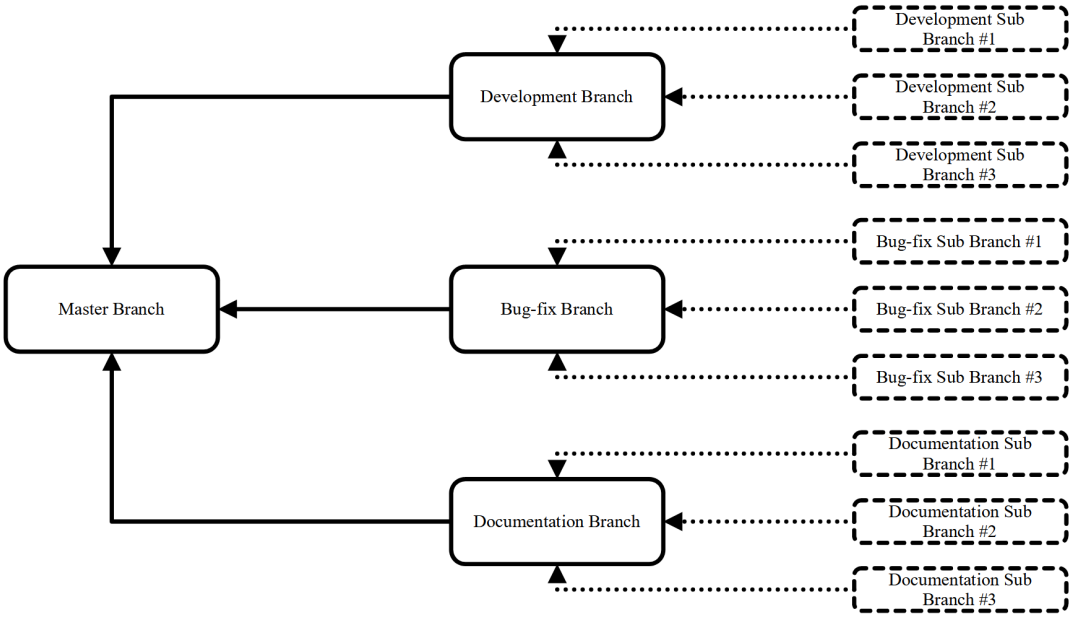

Figure 3: The Git repository branch structure for the project

4.1.1 Development Branch

The main purpose of the development branch is to sandbox new features/functions being developed from the master branch. This strategy prevent the new features from potentially causing bugs/regressions in the master branch.

4.1.2 Bug-fix Branch

The main purpose of the bug-fix branch is to fix bugs that have been discovered, on existing features of the Data Analysis package. These fixes are then tested in the bug-fix branch before being merged with the master branch.

4.1.3 Documentation Branch

The purpose of the documentation branch is to update the package manual and doc strings comments of each component of the package. This means that all changes made in this branch should not in any way alter the functionality of the package, when compared compared to the master branch.

4.2 Examples

Examples of pull requests can be seen in Figure 4, where commits done on the development and documentation branch were merged with the master branch.

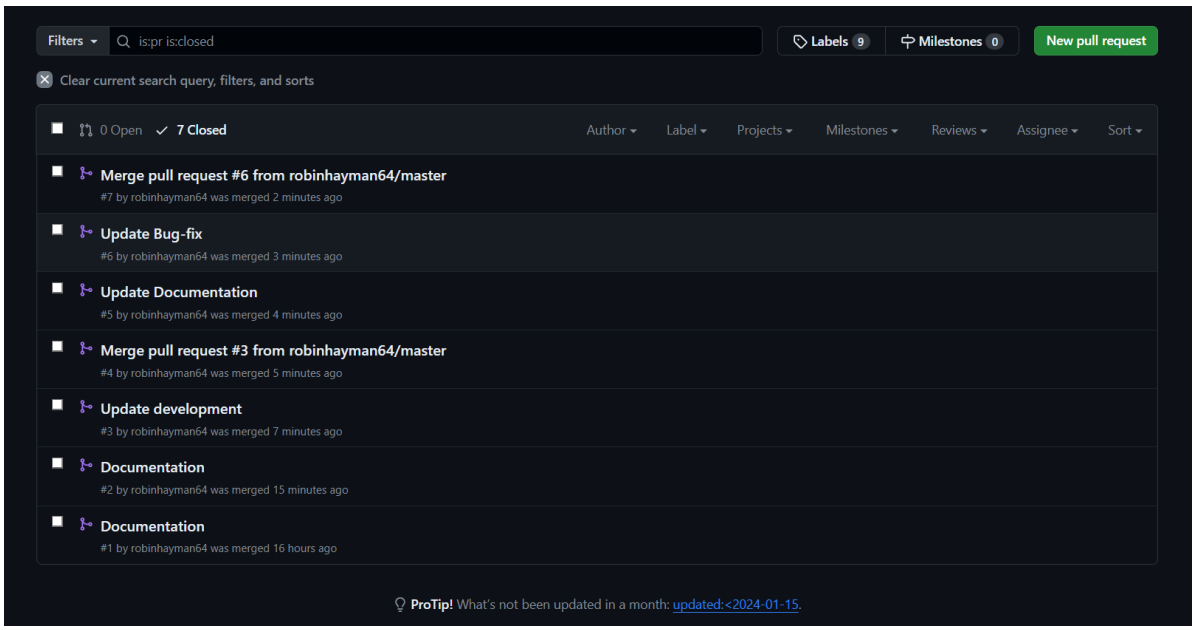

Figure 4: Github pull-request examples

5 Data Analysis

5.1 Dataset

Two datasets were used for the testing of the data analysis. The first dataset, "sales-data.csv" were used for the data analysis part of the python module. The dataset consist of type of products, the quantity sold and the revenue the sales accumulated for a particular day. The data analysis results can be seen in Subsection 5.2.

The second dataset were "447-monica-power-july-2023.csv" which is a more complicated dataset. 32 Copies of this dataset were made and the statistics results were calculated on all 32 copies simultaneously with varying amount of thread counts as a way to test the multi-thread advantage of the data analysis. The multi-thread advantage results can be found in Subsection 5.3.

5.2 Analysis on Dataset

5.2.1 Quantity of Product Types Sold

A bar graph was generated on the amount of units sold from the 3 different product types, as can be seen in Figure 5:

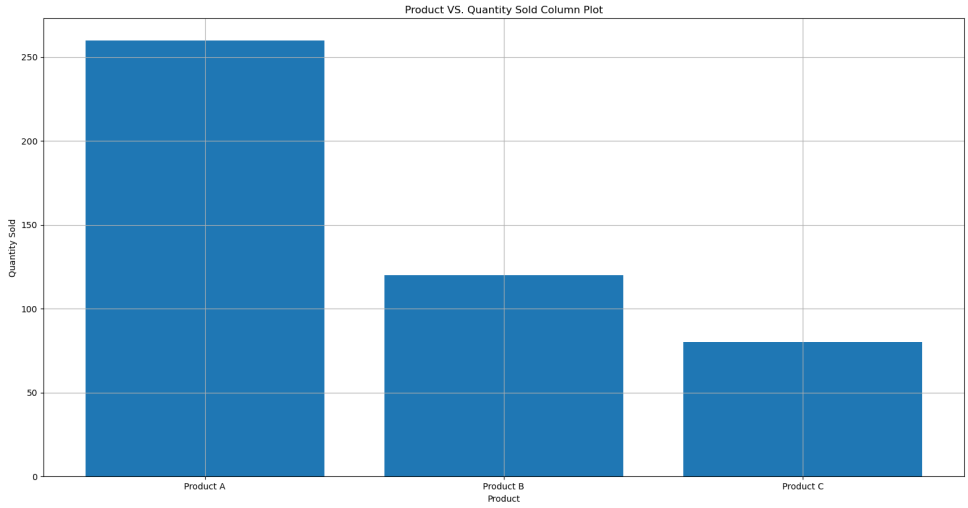

Figure 5: Quantity sold of each product type

From the graph, it can be seen that "Product A" is by far the most popular product with over 250 units sold. "Product B" is second with 120 units and "Product C" is third with 80 units.

5.2.2 Revenue Distribution

A box graph was generated on the revenue distribution of the dataset, as can be seen in Figure 6:

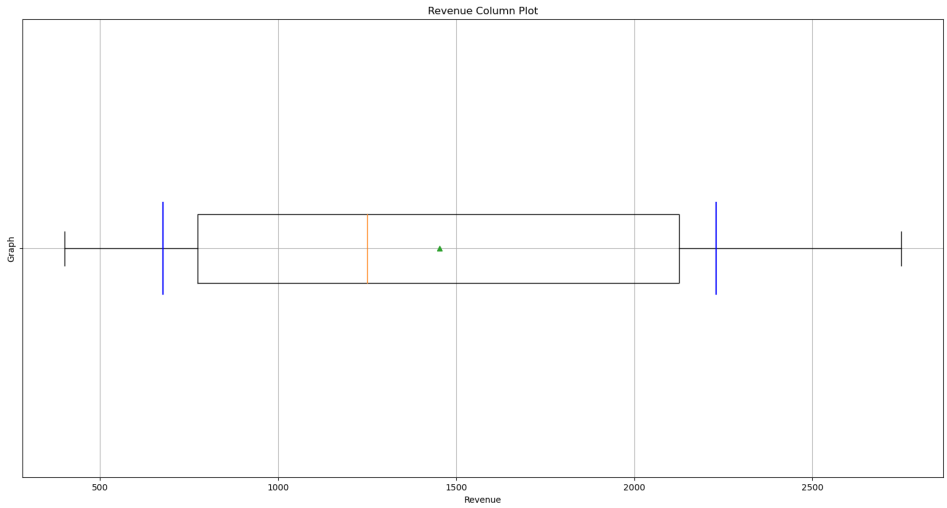

Figure 6: The revenue distribution of sales done

From the graph, it can be seen that the minimum revenue made was about 400, the maximum was 2750, the mean was just below 1500, the first standard deviation is about 680 and 2230 respectively, the median was 1250, the first quartile is 750 and the third quartile is 2250.

5.3 Multi-Thread Advantage

A line graph was generated on the amount of threads vs the amount of time it takes to finish processing the statistics of 32 "447-monica-power-july-2023.csv" data frames. The graph can be seen in Figure 7:

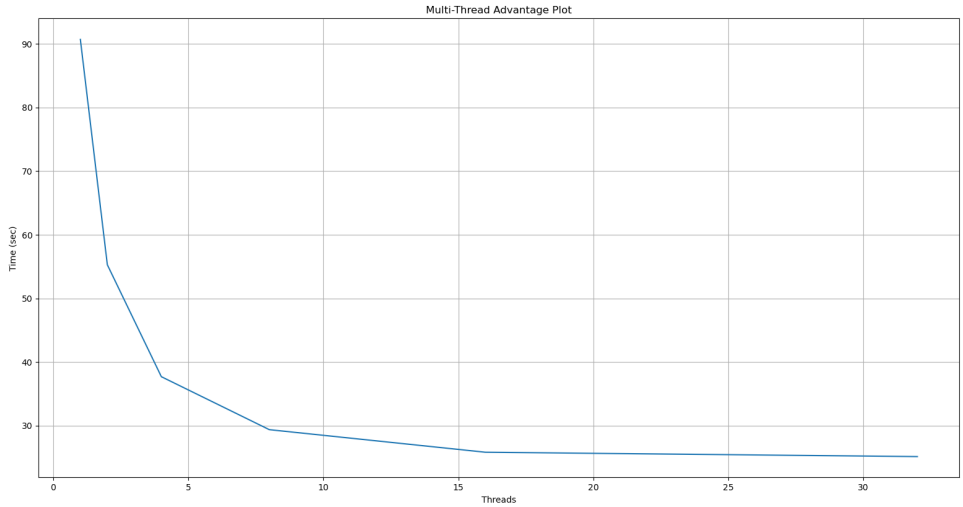

Figure 7: Threads vs. Time graph to finish 32 data frames

6 Conclusion

Overall the project was a success, with the core functionality of the data analysis package being fully developed with the inclusion of multi-threading functionality, plotting capability and a file sync system. The functionality of the package has also been quite thoroughly tested with test data.

Lastly, a git repository has been setup to organise the development areas of the data analysis package in a logical manner and to prevent potential regressions that might occur with future development.

Some improvements that could be made in the future to the data analysis package a more robust plotting module that has more features/plotting options. The plotting time could then also be further enhanced by integrating multi-threading into the plotting functions.

References

[1] "Pandas documentation," pandas documentation - pandas 2.2.0 documentation, https://pandas.pydata.org/docs/ (accessed Feb. 15, 2024).

[2] R. Hayman, "Robinhayman64/Python_data_analysis_package," GitHub, https://github.com/robinhayman64/Python_Data_Analysis_Package (accessed Feb. 15, 2024).