GUI for PluMA: Plugin-Based Microbiome Analysis

Student: Cesia Bulnes
Mentor: *Dr. Trevor Cickovski,* Florida International University
Instructor: *Dr. Masoud Sadjadi*, Florida International University

Problem

Many of PluMA's users aren't developers, therefore it can be a difficult process to download plugins from the pool and forming the pipeline. The GUI for PluMA was developed in order to facilitate this process, and make the user experience as easy as possible, for both skilled and unskilled users. My project's goal was to establish a Graphical User Interface, where users did not have to code anything at all from the start to finish of installing plugins to forming the pipeline.

My goal as a developer was to display the plugins that have been installed, and scrape the whole plugin pool and make it accessible for users to install more plugins without ever using the command line. I also helped with the UX of the pipeline format.

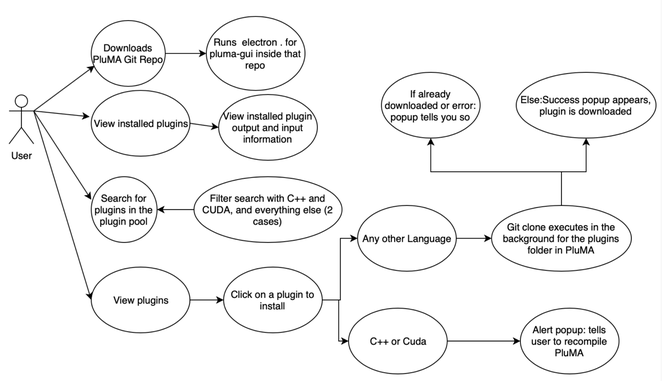
Current System

Currently, Users have to go to the BioRG website to access the plugin pool. They must also go to each plugin's GitHub and download using the terminal. It is difficult for users if they have not used the terminal or don't know any git. Users also have to form the pipelines through the terminal.

Object Design

User Story Diagram for User Story #27 & #30
- As a user, I should able to view all plugins from the pool in a popup window
- As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping.

USER STORY DIAGRAM



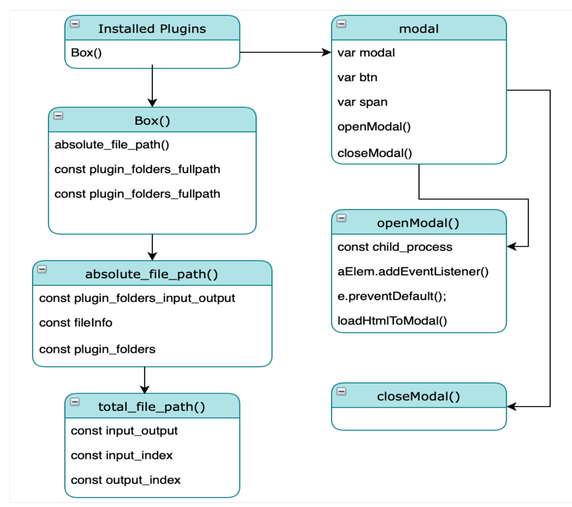
Requirements

Development of the project goes as following:
- Setting and creating the project using git and electron
- Conversion from static to dynamic interface
- Building and compiling webscraping of plugins and having the pipelines be dynamic

Integration of webscraping and installing pipelines required the following:
- Dynamically read the html from the plugins folder, in order to let the user know what pipeline has been installed
- Dynamically webscraping the BioOrg pipeline pool using cheerio
- Dynamically letting users click on plugin, if they wish to install it, and writing scripts as child processes of the event in order to execute git clone commands in the background
- Displaying a warning box if the plugin has already been installed
- Displaying a warning box should the user want to git clone a plugin written in C++ or CUDA, since the whole system of PluMA would have to be recompiled

System Design

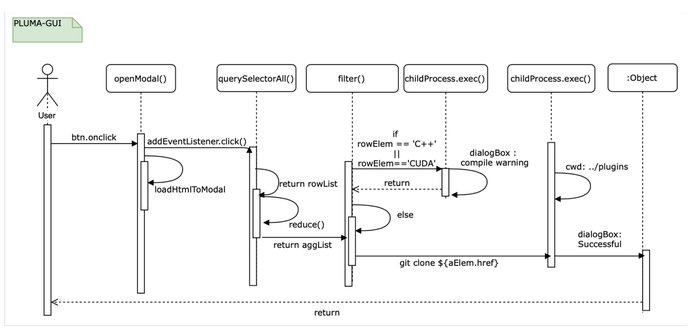
UML CLASS DIAGRAM



Implementation

Sequence Diagram for user story #30
•As a user, I want to git clone the repo of a selected plugin inside my plugin folder in the background of my webscraping.

SEQUENCE UML DIAGRAM

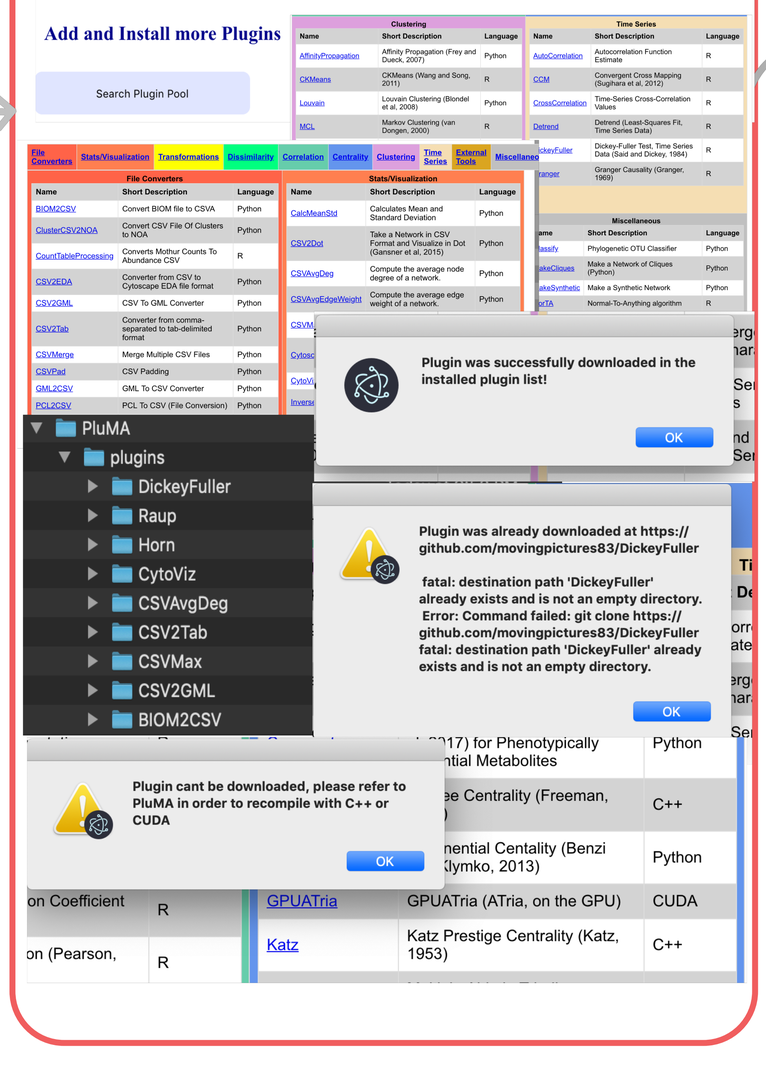


Screenshots

Installed Plugins

BIOM2CSV	Input: BIOM file,Output: CSV file
CSV2GML	Input: CSV (file to convert),Output: GML (converted file)
CSV2Tab	Input: CSV file,Output: CSV tab-delimited file
CSVAvgDeg	Input: CSV (edge weights),Output: none (average degree printed to screen)
CSVMax	Input: CSV (samples),Output: screen
CytoViz	Input: GML (network),Output: none (Cytoscape visualizes network)
Horn	Input: CSV (abundances),Output: CSV (dissimilarities)
Raup	Input: CSV (abundances),Output: CSV (dissimilarities)

Add and Install more Plugins



Verification

Manual Testing was performed with PluMA

Test Case ID: PluMA_30	When not C++ or CUDA
Purpose	To test the functionality when a user was clicking a plugin from the plugin pool to install, the plugin would not need recompiling therefore it could not be C++ or CUDA
Preconditions	The user should have a plugins folder The user should have git installed The user should have cheerio installed The user should have electron js installed
Input	Click -> git clone <plugin name> cwd: '../plugins'
Expected Output	If the plugin had already be downloaded previously, a dialog box will let them know it exists already in the plugin folder. If the plugin was downloaded for the first time in the plugins folder, a dialog box well let the user know it was successful.

Technologies

HTML, CSS, JS, Chrome, Node, Electron, cheerio

Summary

- Users can use the GUI in order to view their downloaded plugins in their plugins folder
- Users can view the input and output files needed for each plugin in order to form a plugin
- Users can search the plugin pool dynamically by web scraping the website (it can be updated dynamically)
- Users can choose to download plugins without having to execute commands on the terminal

Acknowledgement

The material presented in this poster is based upon the work supported by FIU BioOrg Research Lab, Dr.Giri Narasimham and Dr.Trevor Cickovski. I thank Dr.Cickovski for assistance, cooperation and mentorship that I received throughout this process and my teammates Rishab Vaidya and Bhavyta Chauhan for their teamwork.

School of Computing & Information Sciences
Spring 2019 Senior Design Project

FIU Engineering & Computing
School of Computing & Information Sciences