Plugin Based Microbiome Analysis Graphical User Interface (PluMA-GUI) Version 0 - User Manual

**Team Members: Rishabh Vaidya, Cesia Bulnes, Bhavyta Chauhan**

**Product Owner(s)**: Trevor Cickovski

**Mentor(s)**: Trevor Cickovski

**Instructor**: Masoud Sadjadi

***Abstract***

*This document presents the information necessary to gain a good understanding of PluMA: Plugin-based Microbiome Analysis. Our goal with PluMA is to facilitate the construction of flexible and lightweight analysis pipelines through which a developer can implement a new algorithm in their programming language of choice, and easily test and debug within a larger pipeline alongside stages in different languages that potentially use different file formats.*

*PluMA accomplishes this through plugins, and has a large collection available in its plugin pool, implemented in various programming languages for both the CPU and GPU. Plugins can be run sequentially to form a pipeline, and can be easily added, removed or substituted through our user interface. Since plugins are dynamically loaded, you can assemble a pipeline by downloading PluMA and only the plugins that you need from the pool.*

*Currently, PluMA can be intricate for those who don’t have coding experience or have never used the terminal. With the GUI, users can create the same pipeline formation and install plugins with a very easy user experience that does not require any coding whatsoever.*

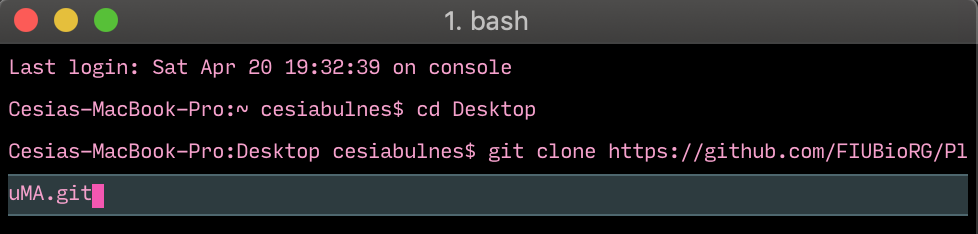
**Installing Dependencies:**

Before we begin, I want to make sure you have git installed. Please refer to this [link](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) to install git on your terminal.

In the command line, we will be installing the PluMA project under the Desktop folder.

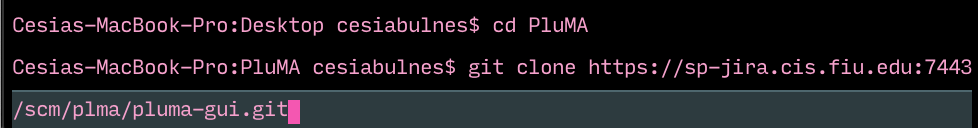
The following command will be used to install

git clone https://github.com/FIUBioRG/PluMA.git



Inside the PluMA folder we will be installing our project pluma-gui

git clone https://sp-jira.cis.fiu.edu:7443/scm/plma/pluma-gui.git

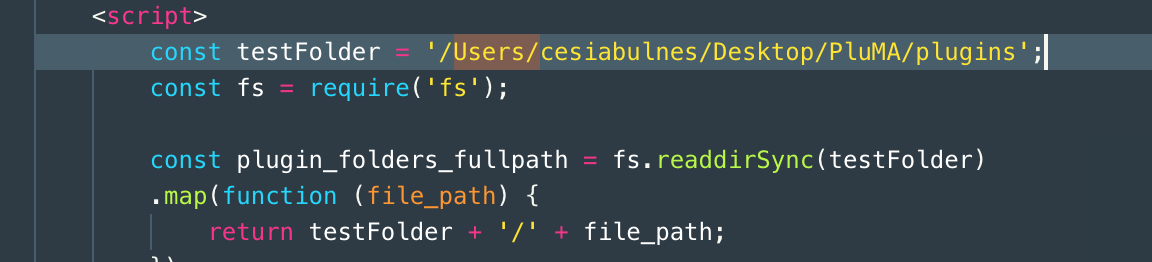


From there, we will be installing any dependencies you may not currently have, such as [electron](https://electronjs.org/docs/tutorial/installation), [cheerio](https://cheerio.js.org), [node](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm), etc. If there are any dependencies you may not have, you can also do npm install -g.

Before launching the application this step MUST BE DONE IN ORDER FOR IT TO WORK. The team had trouble in storing information with our different routes to where PluMA and pluma-gui was stored.

In readmeparser.js, please edit the first line, and replace with your user name after /Users/<your user name>/Desktop/PluMA/plugins

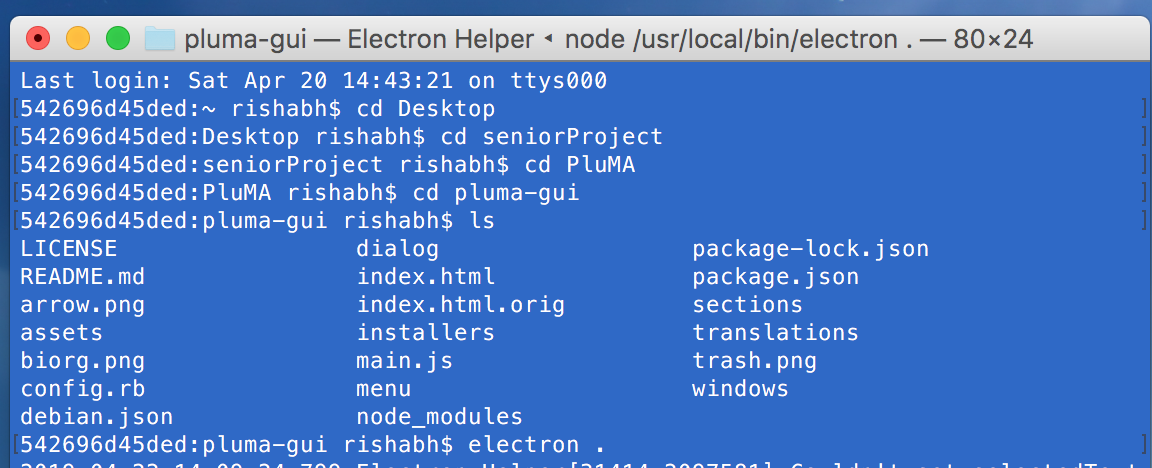


Please do the same steps in index.html, in lines 60,211 & 255



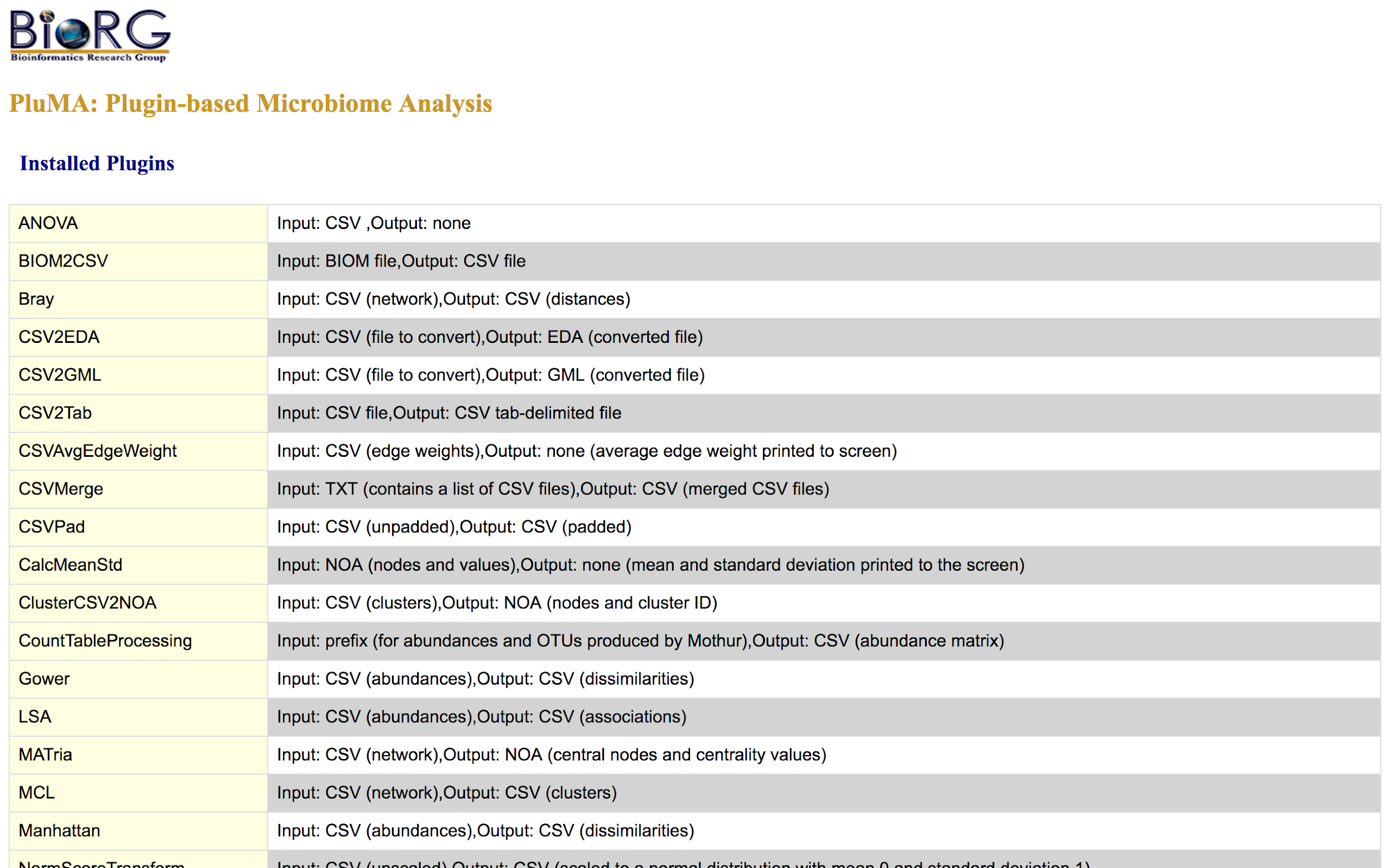
**Launching the Application:**

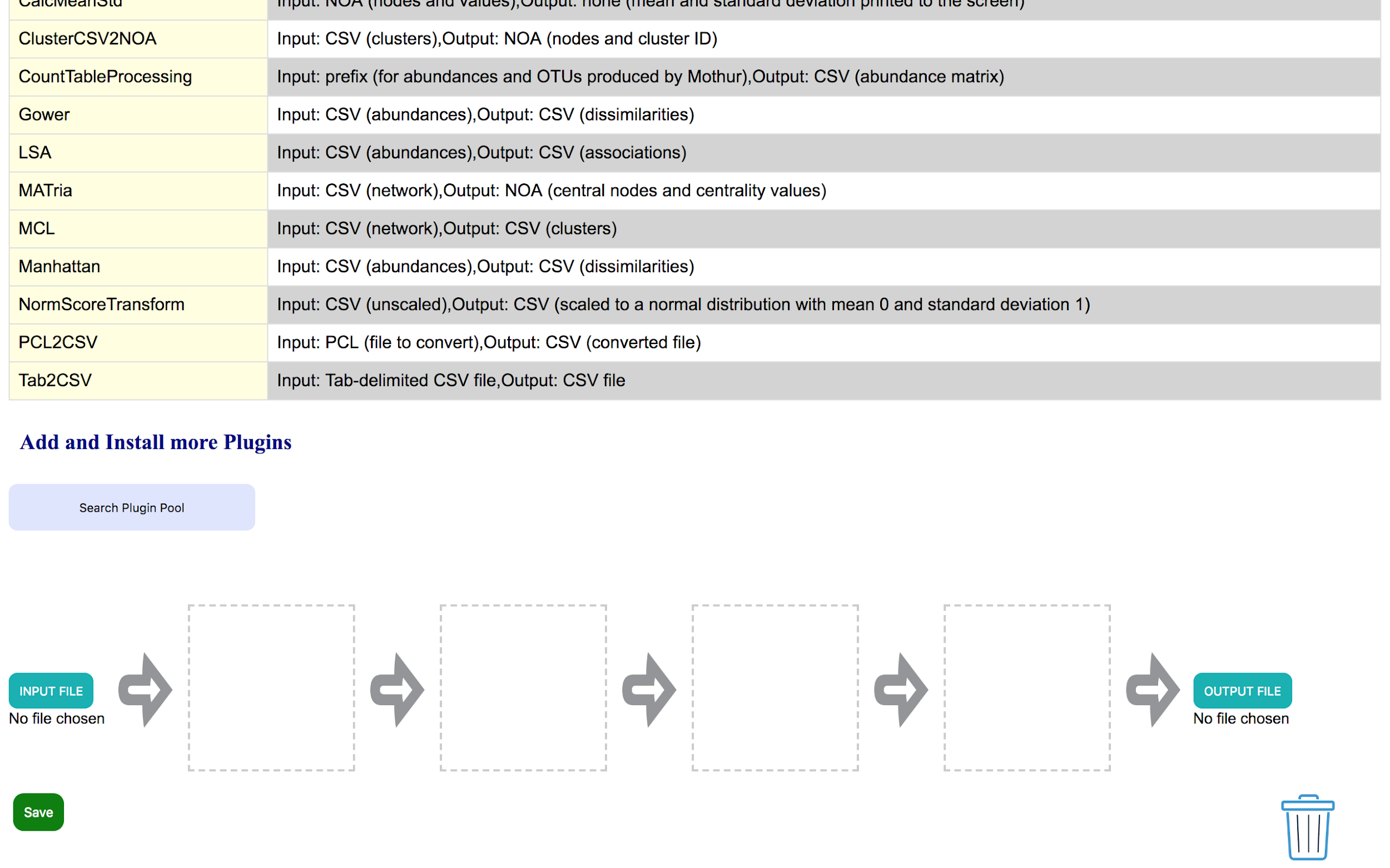
After following the installation instructions, the user can launch the application by typing “electron . ” in the command prompt or terminal.



After running this command, the application should start:

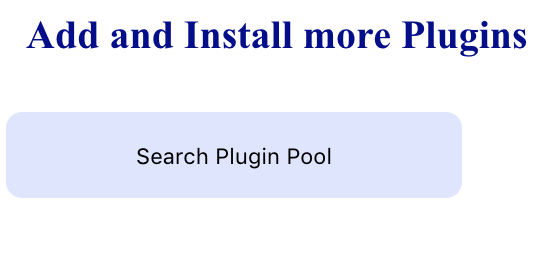
The following installed plugins feature was coded by Cesia Bulnes, this table is reading the plugins folder, and reading all the installed plugins. It is also going inside the README of each plugin and parsing the input and output information from each one displaying.



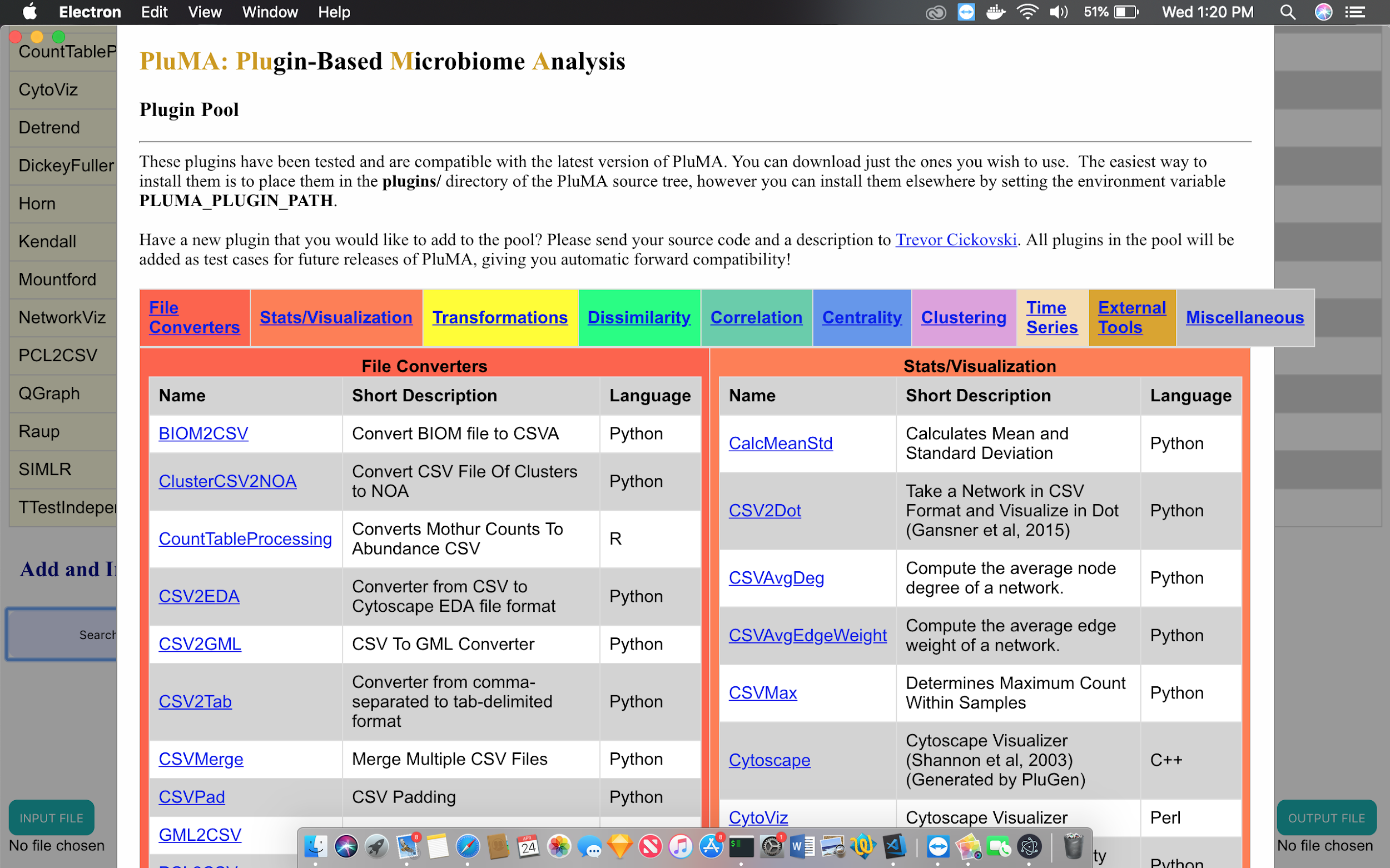


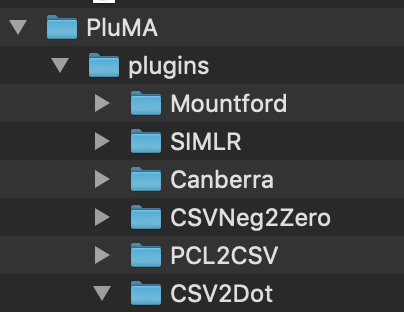
**Web Scraping Feature:**

Here you can search for plugins displayed on BioOrg’s website!

****

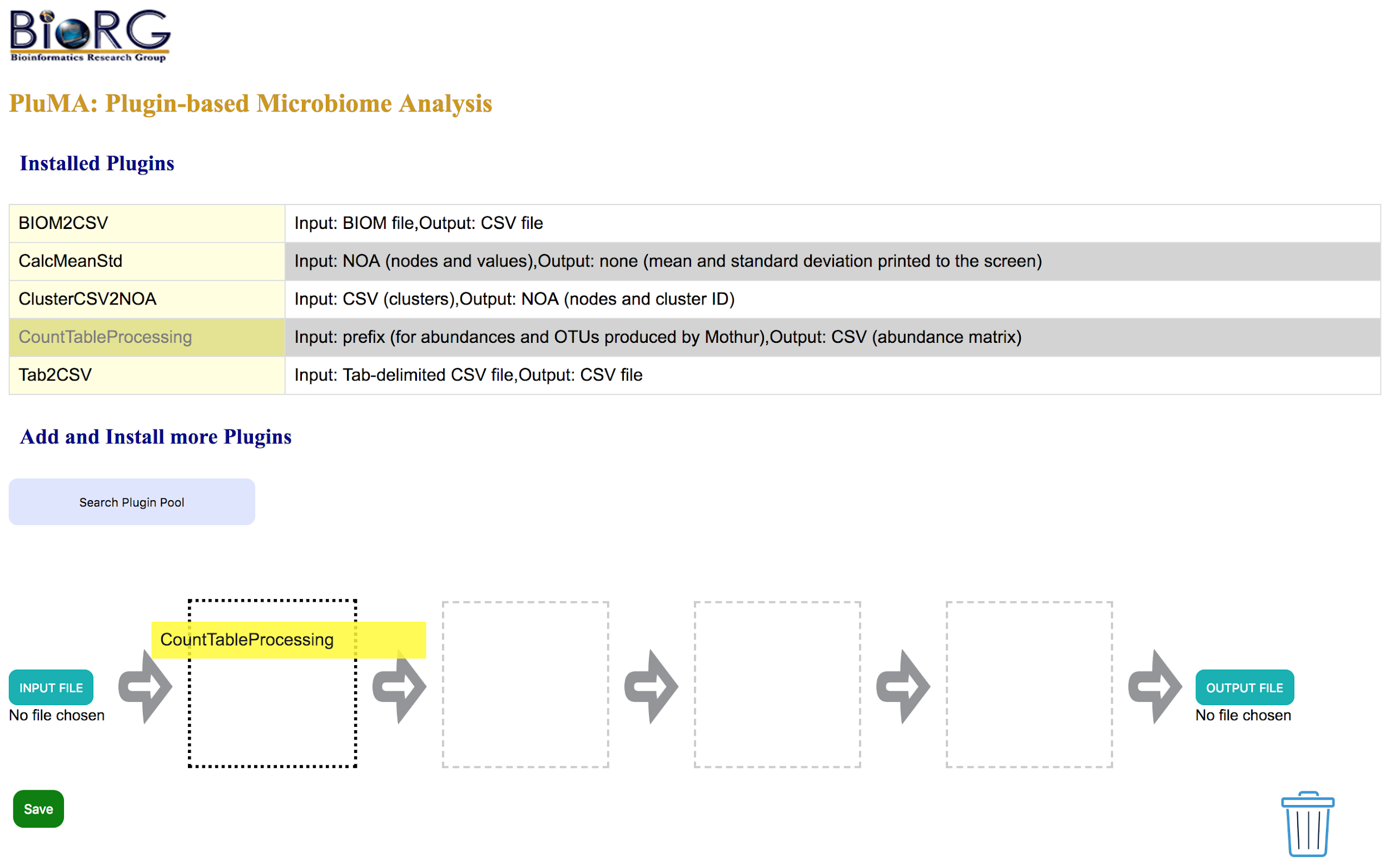
Once the Search plugin pool button is clicked, the whole bioOrg plugin pool url will be scraped. Users then will get to click on each plugin which will automatically install it into the plugins folder inside PluMA, rather than going into each plugin’s github and git cloning each url.

****



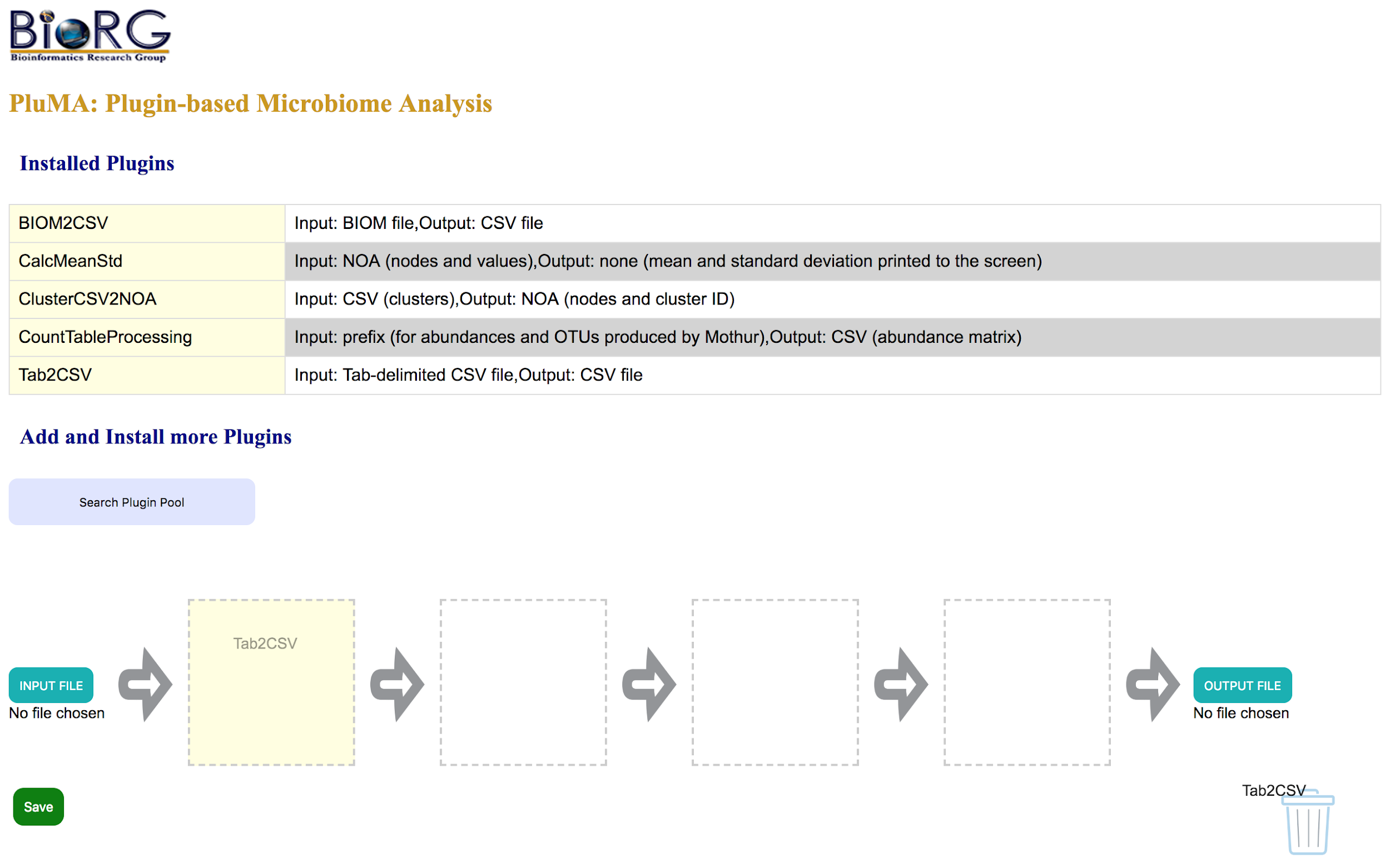
**Drag and Drop Feature:**

The user can drag and drop the plugins of their choice into the pipeline.



**Delete Plugins:**

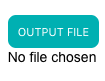
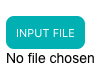
The user has an option to delete any unwanted/unnecessary plugins by dragging them into the trash can on the page.



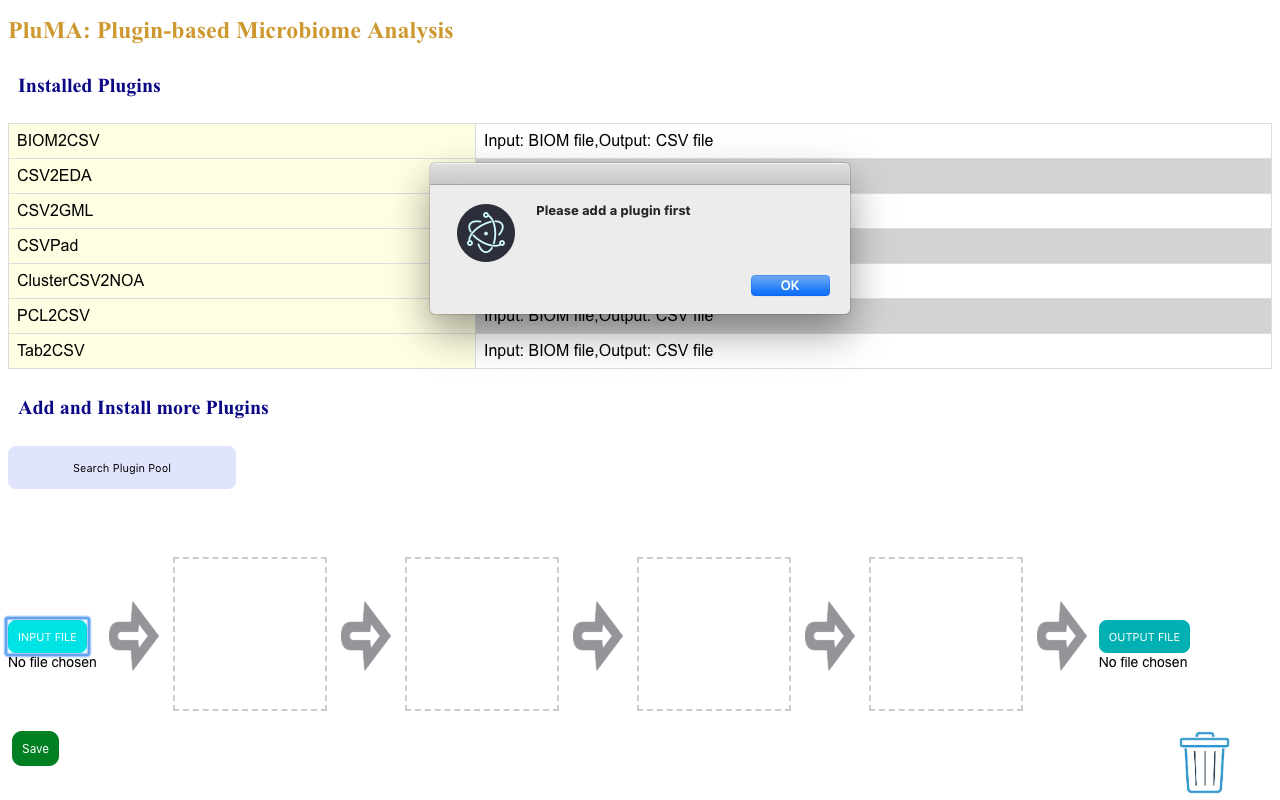
**Upload Input/Output Files:**

Each plugin has an input and output file associated with it. For the PluMA software, it is possible for the user to provide input and output files for the intermediate plugins in the pipeline since it is not necessary that the pipeline operates linearly.

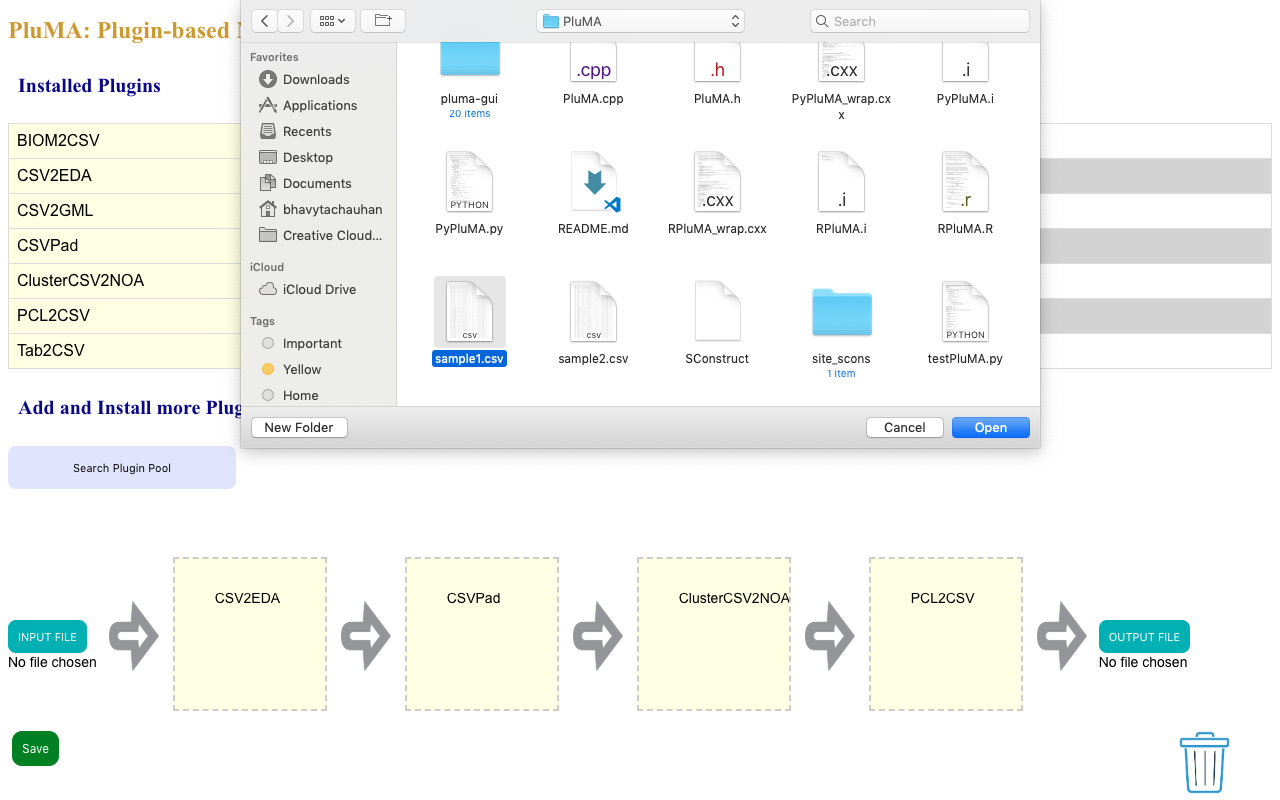
Based on the current implementation of the interface however, it is assumed that only linear pipelines are allowed. Therefore, the user only needs to upload an input file for the first plugin and an output file for the last plugin. This is done through the input and output file upload buttons (shown below) at the beginning and end of the pipeline.

****

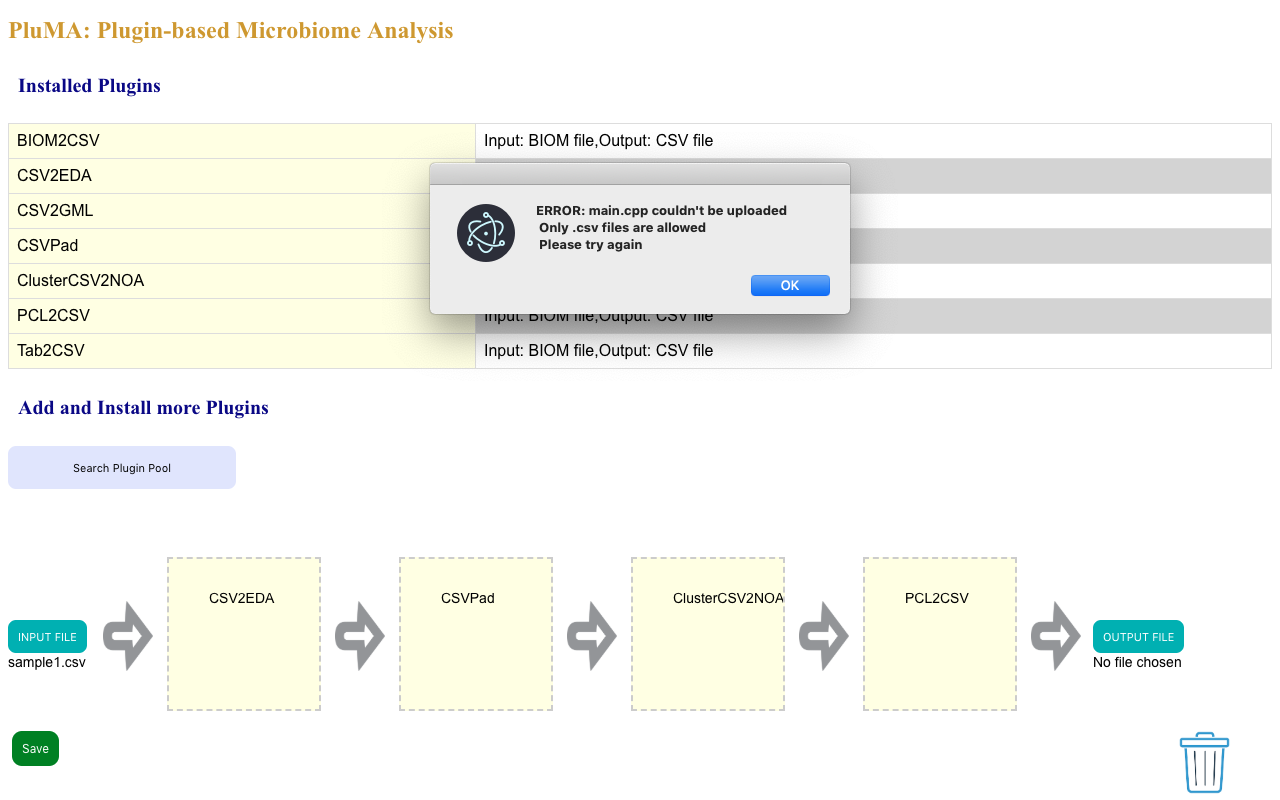
The user may not upload a file unless the associated plugin has been dragged to the respective boxed area. An alert is displayed if the user tries to do so.

****

After adding the desired plugins and clicking on the upload file button, a local file browser window is displayed from where the user can select the desired file for upload.

****

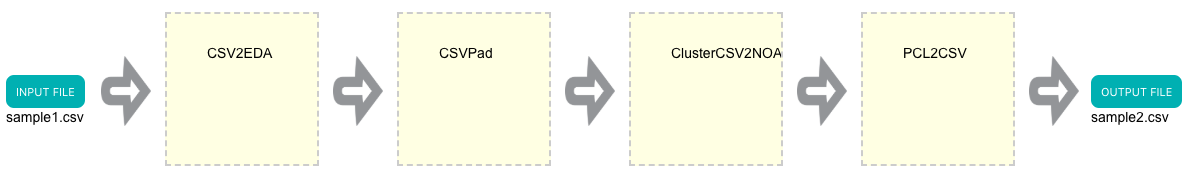
If the type (extension) of the chosen file is incorrect, the upload is canceled and an alert is displayed informing the user that the upload failed due to incorrect file type and the correct file type that is allowed for upload.

****

If the upload is successful, the name of the uploaded file appears under the button.

****

Here is a view of a completely assembled pipeline:

****

**Save Information:**

The users have the option to save all the information such as the input/output filenames and the plugin names. Once the save button is clicked a config.txt file is generated.

