Z-Lab Automation and Analysis Pipeline User Manual

Siddarth Raghuvanshi\*

\*Please direct all complaints to Andrew Hagner or John Edgar

**Goal of Document**:

Explain to a user the steps taken under the hood of the Z-Lab Automation Pipeline. No explanation of the code syntax will be provided, instead the just the purpose of the code will be explained. For this reason, as no data manipulation occurs in the EpMotion\_GUI.py script, it will be ignored in this document.

If you are interested in how to use the tool, I recommend the README on the gitlab link.

**Architecture Overview:**

The entire program is run from JMP\_to\_EpMotion.py, this file is responsible for sending and obtaining data from the other scripts in the program. The three other scripts which are important are Convert.py, Epmotion\_GUI.py, and Output.py.

JMP\_to\_EpMotion.py calls the EpMotion\_GUI.py file to create a GUI to get data from the user including the DOE table and specifications about plates and tubes for the experiment.

JMP\_to\_EpMotion.py calls the Convert.py file to find out how to dilute the stock concentrations and prepare the factors to be at the level desired, as well as create the commands which would prepare the plates. The goal of this file is to covert the DOE experimental matrix into commands for the EpMotion.

JMP\_to\_EpMotion.py calls the Output.py file to convert the lists and dataframes to a csv format for the EpMotion. In addition, this file is also responsible for writing up the protocol and the organising the folder structure.

**Outputs:**

The program creates an Experimental\_Files folder marked with the date and time of the program. The program stores a summary excel file which contains the DOE experimental matrix provided and the excel sheet which the user has used to enter the concentration information into.

Each block of the user’s experiment contains its own folder. Within the block folder there is a text protocol file, an EpMotion folder contain the CSVs for the EpMotion, and a Cytoflex folder which should be used to store flow data.

**Program Actions:**

**JMP\_to\_EpMotion.py:** The EpMotion program first creates a numpy array from 1 to 24 to represent the 24 well racks on the EpMotion.

The program then uses the Get\_Data function to get information from the user about the specifics of the experiment. The data includes:

* The experiment matrix
* The plate type
* The volume of liquid in a well
* The number of edge rows
* The dead volume in a tube which the EpMotion cannot access
* The cell volume which will top up the liquid to the well volume

The program then reads the experiment matrix and divides the experiment per its blocks. Each block is treated as individual experiment from this point forward. For each block, a new EpMotion called Handler\_Bing is created and the above variables except the experiment matrix are set as its attributes. In addition, every block has its own folder structure as described above.

The blocks are sent to Convert.py through a function called Rearrangement which produces EpMotion and user commands. These EpMotion commands are sent to a function called Epmotion output, and the user commands are sent to the Protocol\_Output function. In addition, the concentrations csv which is also moved into these folders. All of these are placed into the folder of each block.

After each block is run, the program then moves a copy of original command excel file into the folder.

**Convert.py:** The script takes the Experiment matrix provided by the program and finds the “X” variables and the “Y” variables. The program creates a variable called Factor\_Vol by dividing the volume of a well by the number of factors. This is the volume of each factor which will be added to the well, the same is done for cell volume. The cell volume is subtracted from the factor\_vol as that volume of media will be created added later in the form of cells. The program then finds the names of the Factors and Levels.

The program builds a dataframe which contains the number of times each condition is tested. As this is a count, by multiplying the volume of the factor added in each well, you can determine the amount of liquid needed for each condition. The same is done to find the volume of liquid which will be added by the cells.

The program then runs checks on the space, volume required and gets the concentration information of the factors and levels in a csv from the user.

The user provides the concentration of factor as stored in source, and the program calculates a dilution of that which is sufficient for the experiment and does not produce waste. To calculate this value, the program multiplies each condition volume which was calculated earlier by the concentration and sums them together for reach factor. This gives the mass of each factor used in the experiment. This mass is then divided by the largest concentration for each factor. This gives the volume for each factor’s manual dilution. The manual dilution is then checked to ensure it would fit in a tube.

As we now have the mass of the factor used and the maximal volume possible, we can find the manual concentration by dividing them and multiplying by the number of factors. This is done as each factor will be diluted when they are deposited into each well. The volumes to generate this concentration are then calculated to be passed to in the protocol.

This information of the manual concentration and the user’s inputs was created into a dataframe which is passed to the function Dilution\_Prep. This function finds the volume of the manual concentration needed to reach the concentration of the level at the volume of the level required. If this is too small for the EpMotion, the volumes are scaled up to the max epitube volume.

If even after the scale up, there are still volumes below the minimum cutoff, the program creates serial dilutions. The serial dilutions are calculated by taking the log of the (minimum value divided the volume needed) with the base of the (volume of the epitube divided by the minimum volume). This give number of serial dilutions needed for each factor.

The program then creates a dataframe of the locations of the manual dilution tubes, and adds the location of the serially diluted tubes. In addition, the program also makes a set of commands for the EpMotion to create these serial dilutions. These commands contain a variable called rank, the commands will later be sorted by this variable, as if a factor needs to be serially diluted twice, the first dilution must occur before the second.

The dilution\_prep program produces commands for the epmotion as well as the volumes to add and total volume. These are passed on to the program Factor\_Dilution\_Commands. This program takes the volumes from the appropriate cereal dilution or source and creates all of the levels for each of factors in commands for a CSV. As this function has no effect on the data other than placing it into the CSV, no further detail will be provided.

The commands for the factorial design are done by creating a plate class. A list called Plates stores the number of plates needed for the experiment. Each plate class will store the commands generated by the program.

The Plates, Volumes for Manual dilution, factor commands, and serial dilution commands are sent back to the JMP\_To\_EpMotion script. From this point, this information is converted to CSVs in the Output.py Script.

**Output.py:** The script takes information and creates the folder structure and converts commands to CSVs. As information is not manipulated in this script, it should be straight forward to read. Therefore no explanation beyond the Output section is given.