***Java Final Project***

**Introduction**

The purpose of this program is to filter out any aligned samples from mass spectrometry data that may or may not be useful in interpreting the results form a metabolomic study, particularly in the case that there are false positives, meaning there is thought to be a valid metabolite but in reality there isn’t, it is merely a flaw within the alignment algorithm. This is especially important when trying to understand how clean or closely related the data is between different batches run on the instruments to obtain succinct intensities between these different replicates. Since instrumental drift can occur, along with column degradation, it is important to make sure that those features aligned for a single sample are actually for the same sample and not an issue with the batch. That is the purpose of this program, to filter those out and be left with those samples or metabolites that are true positives from statistical analysis.

**Data**

The data that needs to be passed through this program is a CSV file that is a feature table, where the rows are individual samples or metabolites and the columns are individual features from each run through the instruments, whether it be an internal standard, a blanks sample, a preinject sample, or an unknown sample. From here the program can parse the data and perform the necessary statistics across the different batches.

**batch\_anova\_effect Program**

*ArrayList<String[]> read\_file\_nist(File myFile, String subString, int featuresPerBatch, double anovaCuttoff)*

Description:

This method organizes most of the other methods within the program, dictating the majority of the workflow.

Parameters:

myFile = input file given in the main function

substring = String given for the internal standard common name within the header of the input file’s feature table

featuresPerBatch = integer used to describe how many internal standard feature columns are in each batch to ensure grouping within the various arrays is correct when conducting OneWayAnova test.

anovaCuttoff = the double value given to dictate the P-Value cutoff as to whether a sample is omitted or inserted into the new feature table

Returns:

This method returns an ArrayList<String[]> of all the new, cleaned rows the of new csv file.

*List<String> loopRow(String[] splitRows)*

Description:

This method is called within the read\_file\_nist to populate an ArrayList<String> with each row of the input file, by splitting each row into its component for the samples only and not the header, where each line is an entry and returned as a list of strings

Parameters:

splitRows = is a single sample from a non-header row within the feature table, which has been parsed into a String array

Returns:

This method returns an ArrayList<String> of each feature contained within a single sample passed through

*List<double[]> findIndices(List<String> header, List<String> body, String subString, int featuresPerBatch)*

Description:

This method loops through the header list that was created previously and iterates through to find the substring given by the user within the header, taking those indices and finding the feature index match within the sample row. From there, the featuresPerBatch integer passed by the user, that groups the number of features per batch within an “if” statement that then keeps track of how many features have been added to a double array list, if that number is met without any blank or missing values, that array is appended to a List<double[]>. This process repeats until the method has iterated through the entire length of the header. If there are blank values found, the indexBody should be cleared in the rare event this happens.

Parameters:

header = is a list containing the string of each header in the first row the feature table

body = is the list containing the string of each feature for a single row or feature from the feature table

subString = is the target string that the method is looking for within the header list

featurePerBatch = is the integer give that tells how many features per sample of the internal standards, that are found using the subString, are in each batch.

Returns:

This method returns a List<double[]> that can be passed into the anovaTest method.

*double anovaTest(List<double[]> indexBody)*

Description:

This method conducts a One Way Anova on the classes of data that are within the list double array passed in.

Parameters:

indexBody = is a list with each entry containing a double array, which is holding each of the features per batch

Return:

This method returns a double P-Value, which is then checked in the read\_file\_nist method to see if the P-Value is greater or less than the specified tolerance.

*void newBetweenBatchBlank(String filePath, ArrayList<String[]> cleanedArray*)

Description:

This method reads in the new array string for each row of the filtered and cleaned feature table based off the P-Values and returns a CSV file to the specified location from the user. This should be the new cleaned version with P-Values appended in the final column.

Parameters:

filePath = this is the specified path given by the user for where the CSV file should be saved to.

cleanedArray = the string array list that contains all the new sample rows for the new file

Return:

This method doesn’t return anything, but it does read out the new file to the path specified by the user