**FINAL PROJECT**

**Contributors:** YeeMey Seah, Yusuf Corr

Github - https://github.com/YusC3

**UWB - Dr. Hillesland Lab**

**DISCLAIMER: This code is meant to deal with outputs from Dr.** Yee Mey Seah’s code found here: <https://github.com/yeemey/dvh_mms2/blob/master/notebooks/data_exploration/mutation_coverage.ipynb>

IN PARTICULAR, the researcher should run this line:

SAMPLE\_LINEmut***N***, SAMPLE\_LINEevidence***N*** = subset\_gd\_to\_df('/path/to/annotated.gd', 'D2-0', 'D2', '0', cov=**True**), for each generational time stamp in a particular sample, using N + 1 to create different titles for each df creation.

Creation of csv file for all generational time stamps for a particular sample line: For each SAMPLE\_LINEmut***N***, use pd.concat([SAMPLE\_LINEmut***1***, SAMPLE\_LINEmut***2***, SAMPLE\_LINEmut***N***], ignore\_index= True) to get a combined df of all ACCEPTED mutations. Then use: Accepted\_mut.to\_csv to create a CSV file

USE SAME METHOD FORSAMPLE\_LINEevidence***N*,** to create Evidence\_df.csv

USE FINAL CSV’s for this program.

**Current Developments**:

* Implementation of statistical analysis: Will develop a statistical analysis to test the possibility of proposed rejected evidence in place of blanks across the time stamps found in the accepted mutations csv file
* Editing of help string and code in general (not as much function, but back end resources).
* The creation of test code. ( I didn’t create test code but test scenarios and jupyter notebook/powershell.)
* Automating the process of creating current Input files

**Program name: “**f2\_cdfg” for: find and filter while create data frame and graphs

**Code:** Mutation data frame filtration to find missing evidence for a mutation supported in one or more generational sequences.

**Libraries:**

* pandas as pd
* numpy as np
* seaborn as sns
* argparse import ArgumentParser
* matplotlib.pyplot as plt

**Description:** This codehas the sole purpose of filtering through a compiled data frame of ALL accepted mutation across all sequenced generational time points, find the mutations that do not have “accepted data” across all generational time points (and this can be specified by number of matches that SHOULD exist in the data frame), and finally, using another data frame (contain ALL the evidence for every generational time stamp, both accepted and rejected evidence for mutations) and create a new data frame with the rejected evidence that matches the data missing for the mutations that are not covered throughout each time stamp.

Inputs: example\_accepted.csv & example\_evidence.csv (MUST HAVE THESE TWO CSV’s BEFORE RUNNING)

Output: result.csv (if requested by -q2 yes),

Graph option: if requested by -q2 yes, will require the position# (location of mutation in genome AS A NUMBER), and genome id name (specified by column “genome\_id”) This will create a graph showing the visualization of polymorphism frequency for that mutation at position *n* and in genome id *some\_genome\_id* before the addition of the missing evidence and after (as two separate graphs).

Interface options: IF YOU DO WANT A GRAPH, set -q2 to “yes”, YOU MUST PROVIDE ACCURATELY: -p, -g

**REQUIRED ARGS**

Accepted\_df: (usage: path/to/file.csv, NO KEY SPECIFIER, MUST BE THE FIRST THING TYPED)a string that contains the path to the accepted mutations csv file

Evidence\_df: (usage: path/to/file.csv, NO KEY SPECIFIER, MUST BE THE SECOND THING TYPED) a string that contains the path to the evidence csv file

-i: (usage: -i 2, MUST BE A NUMBER) the number of time stamps that make up this data frame. This number represents the amount of rows that should be expected for each key identifier value(position # and genome id).or program will conclude data is missing

**OPTIONAL ARGS**

-q: (usage: -q2 yes, default = none) if ‘yes’, will create an output csv file containing all of the available missing evidences (due to rejection) titled “Missing\_Mutations”. ANY OTHER INPUT WILL NOT PROMPT FILE CREATION AND INSTEAD A STRING WILL APPEAR: “No output”. NOTICE: Out put created in current directory.

-q2: (usage: -q yes, default = none) if ‘yes’, will create and display a graph of polymorphism frequency for each sample line before and after the addition of the missing evidence. The “before graph” is denoted by a solid line while the “after” graph is denoted by a dashed line (---). ANY OTHER INPUT WILL NOT PROMPT GRAPH CREATION AND INSTEAD A STRING WILL APPEAR: “No graph”

-p: (usage: position#, MUST BE A NUMBER, default = 0) given the position # for the mutation (ie. Mutation happens at number 1023 in genome) it will find the row with that position in combination with specified genome id (-g).

-g: (usage: some\_genome\_id, ie: NC\_002937, default = none) given the genome id, it will find the position # of the mutation for that part of the genome.

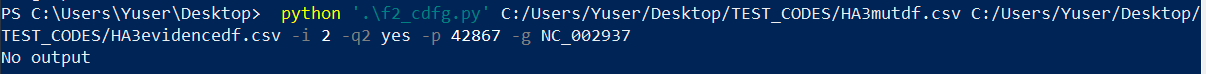
Otherwise you can leave -q, -q2, -p, and -g blank.

EXAMPLE RUN:

>python '.\ f2\_cdfg.py' C:/Users/Yuser/Desktop/TEST\_CODES/HA3mutdf.csv C:/Users/Yuser/De sktop/TEST\_CODES/HA3evidencedf.csv -i 2 -q gg -q2 yes -p 42867 -g NC\_002937

The first path is for the accepted mutations csv and the second is for the evidence csv.

Here -i is 2, which means this data frame is dealing with 2 generational time stamps and thus will expect 2 rows that contain the same genome id and position. -q is given “qq” which will not create an output file and instead prompt a string (“No output”). -q2 is given “yes” which will then create the graphs using key values -p and -g. -p is set to position 42867 and this position for the mutation is given the genome id (-g given “NC\_002937”) where the mutation resides.

Output scenarios (images): 

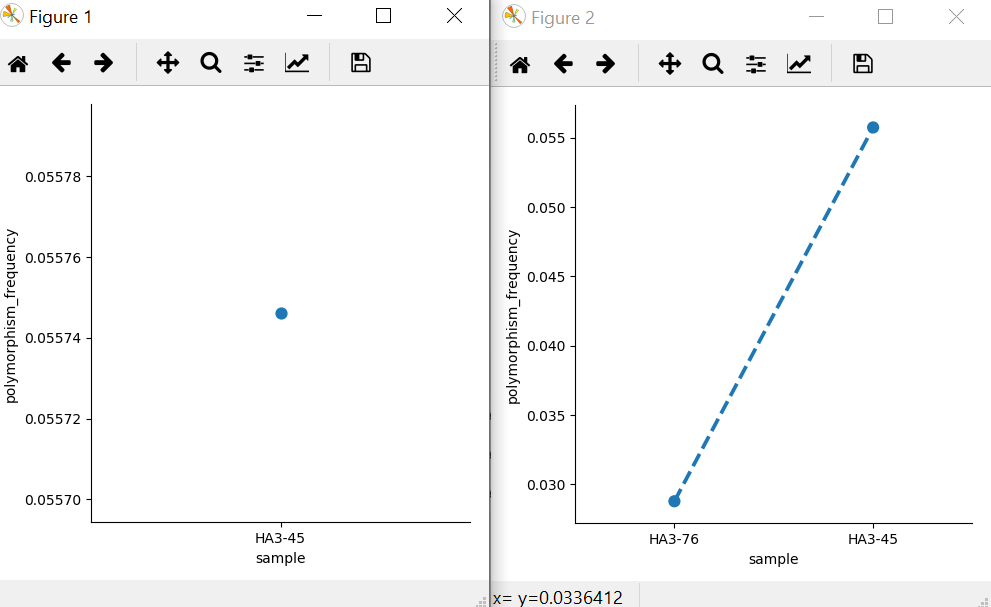


Figure 1 = before w/o missing evidence Figure 2 = after w missing evidence

MUST CLOSE GRAPHS TO CONTINUE USING TERMINAL (in windows: POWER SHELL).