read in each annotated.gd file; as a pd.df or as a dict

initiate a blank\_df

for each df:

initiate a row in blank\_df with rowname = samplename

parse out the freqs

parse out the position

if intergenic; throw it away

elif 3rd column is a .; throw it away

optional step: ignore synonymous mutations

if it’s a nonsynonymous or frameshift, keep it

parse out the locus tag (note---for mycoplasma, only using the last 4 digits of the locus\_tag, thereby merging the syn1.0 and syn3B. Later, you can make a version of the matrix that ONLY has genes that are SHARED)

in blank\_df check whether a column for this locus\_tag exists

if not; create it at the end; then add the frequency to its value in this row

else; add the frequency to its value in this row

if this row has any empty values, assign them = 0

output to csv

#########one option would be to modify the values to be length-corrected values. I don’t think i agree with the motivation for doing this, though... in ecology, when you take a sample and do Bray-Curtis, you’re not assuming that “all else being equal, all species should have equal abundance”. Of course not all else is equal.

For parsing the columns, use Will’s methods in the get\_multiplicity.py file.

for dir in dirs:

for i, line in enumerate(open(mt.get\_path()+'/data/'+dir+'/annotated.gd', 'r')):

line\_split = line.strip().split('\t')

if line\_split[0] not in output\_to\_keep:

continue

if line\_split[3] + '\_' + line\_split[4] in sites\_to\_remove:

continue

frequency = float([s for s in line\_split if 'frequency=' in s][0].split('=')[1])

if frequency != 1:

continue

if line\_split[0] == 'SNP':

if [s for s in line\_split if 'snp\_type=' in s][0].split('=')[1] == 'nonsynonymous':

locus\_tag = [s for s in line\_split if 'locus\_tag=' in s][0].split('=')[1]

frequency = float([s for s in line\_split if 'frequency=' in s][0].split('=')[1])

if ';' in locus\_tag:

for locus\_tag\_j in locus\_tag.split(';'):

if locus\_tag\_j not in gene\_count\_dict:

gene\_count\_dict[locus\_tag\_j] = 0

gene\_count\_dict[locus\_tag\_j] += 1

else:

if locus\_tag not in gene\_count\_dict:

gene\_count\_dict[locus\_tag] = 0

gene\_count\_dict[locus\_tag] += 1

else:

continue

else:

if len([s for s in line\_split if 'gene\_position=coding' in s]) >= 1:

locus\_tag = [s for s in line\_split if 'locus\_tag=' in s][0].split('=')[1]

frequency = float([s for s in line\_split if 'frequency=' in s][0].split('=')[1])

if ';' in locus\_tag:

for locus\_tag\_j in locus\_tag.split(';'):

if locus\_tag\_j not in gene\_count\_dict:

gene\_count\_dict[locus\_tag\_j] = 0

gene\_count\_dict[locus\_tag\_j] += 1

else:

if locus\_tag not in gene\_count\_dict:

gene\_count\_dict[locus\_tag] = 0

gene\_count\_dict[locus\_tag] += 1