

```

import sys

#constants
DEBUG = True

INPUT_FILE_NAME_TRAIN = "ALL_AML_gr.thr.train.csv"
OUTPUT_FILE_NAME_FOLD_VALUES = "ALL_AML_gr_no_one_folds.thr.train.csv"
OUTPUT_FILE_NAME_GENE_DISTRIBUTION = "ALL_AML_gr.distribution.train.txt"

FOLD_DIFF_LT_2 = 'LT2' #Val <= 2
FOLD_DIFF_2_4 = '2_4' #2 < Val <= 4
FOLD_DIFF_4_8 = '4-8' #4 < Val <= 8
FOLD_DIFF_8_16 = '8-16' #4 < Val <= 8
FOLD_DIFF_16_32 = '16-32' #...
FOLD_DIFF_32_64 = '32-64'
FOLD_DIFF_64_128 = '64-128'
FOLD_DIFF_128_256 = '128-256'
FOLD_DIFF_256_512 = '256-512'
FOLD_DIFF_GT_512 = 'GT512'

#globals
countFoldDiffPerRange = {FOLD_DIFF_LT_2 : 0, FOLD_DIFF_2_4 : 0, FOLD_DIFF_4_8 : 0,
FOLD_DIFF_8_16 : 0, FOLD_DIFF_16_32 : 0, FOLD_DIFF_32_64 : 0, FOLD_DIFF_64_128 : 0,
FOLD_DIFF_128_256 : 0, FOLD_DIFF_256_512 : 0, FOLD_DIFF_GT_512 : 0}

def debug(logMsg):
    if DEBUG:
        print(logMsg)

def computeFoldValues(input_file_name, output_file_name, log_file_name):
    with open(input_file_name) as f:
        resultLines = [] #will need to remove all genes with fold values of 1, so store a list
        of all genes which do not have that value
        gnuplotLines = []
        genesWithOneFoldRatio = {}
        geneFoldValues = {}

        #create a list of lines, stripped of the newline
        content = [line.rstrip('\n') for line in f]

        #open the file which will have the lines without one ratios; we don't want to append
        out_file = open(output_file_name, "w")

        #open the file which will have the lines without one ratios; we don't want to append
        out_file_gnuplot = open(log_file_name, "w")

        #just add the first line back to the result lines
        idLine = content.pop(0)
        resultLines.append(idLine + "\n") #writelines requires newlines

        #for every line, compute the fold difference
        for line in content:

```

```
if len(line) <= 2:
    continue

#we need every integer
int_strings = line.split(',')

#the first value is the name of the gene
geneName = int_strings.pop(0)

#set the max and min to the opposite end of the range
maxVal = 20
minVal = 16000

#compute the maxima and minima of each gene
for val in int_strings:
    val_int = int(val)
    if(val_int >= maxVal):
        maxVal = val_int

    if(val_int <= minVal):
        minVal = val_int

#compute the fold difference
currFoldDiff = maxVal / minVal

#if maxVal eq minVal, then it has a ratio of one
if maxVal == minVal:
    genesWithOneFoldRatio[geneName] = currFoldDiff
else:
    resultLines.append(line + "\n")

#add the computed fold difference to its range
if currFoldDiff <= 2:
    countFoldDiffPerRange[FOLD_DIFF_LT_2] += 1
elif currFoldDiff > 2 and currFoldDiff <= 4:
    countFoldDiffPerRange[FOLD_DIFF_2_4] += 1
elif currFoldDiff > 4 and currFoldDiff <= 8:
    countFoldDiffPerRange[FOLD_DIFF_4_8] += 1
elif currFoldDiff > 8 and currFoldDiff <= 16:
    countFoldDiffPerRange[FOLD_DIFF_8_16] += 1
elif currFoldDiff > 16 and currFoldDiff <= 32:
    countFoldDiffPerRange[FOLD_DIFF_16_32] += 1
elif currFoldDiff > 32 and currFoldDiff <= 64:
    countFoldDiffPerRange[FOLD_DIFF_32_64] += 1
elif currFoldDiff > 64 and currFoldDiff <= 128:
    countFoldDiffPerRange[FOLD_DIFF_64_128] += 1
elif currFoldDiff > 128 and currFoldDiff <= 256:
    countFoldDiffPerRange[FOLD_DIFF_128_256] += 1
elif currFoldDiff > 256 and currFoldDiff <= 512:
    countFoldDiffPerRange[FOLD_DIFF_256_512] += 1
else:
    countFoldDiffPerRange[FOLD_DIFF_GT_512] += 1
```

```

        #store the fold difference in the dictionary
        geneFoldValues[geneName] = currFoldDiff
    #end for line in content

    #find the largest and smallest fold diffs; also compute the range
    largestFoldDiff = -1
    smallestFoldDiff = 16000000
    for key, value in geneFoldValues.items():
        geneName = key

        if(value >= largestFoldDiff):
            largestFoldDiff = value
        if(value <= smallestFoldDiff):
            smallestFoldDiff = value

    #now find the number of genes which have these values and record them
    numGenesWithLargestFoldDiff = 0
    numGenesWithSmallestFoldDiff = 0
    for key, value in geneFoldValues.items():
        if(value == largestFoldDiff):
            numGenesWithLargestFoldDiff += 1

        if(value == smallestFoldDiff):
            numGenesWithSmallestFoldDiff += 1

    for key, value in countFoldDiffPerRange.items():
        gnuplotLines += key + "\t" + str(value) + "\n"

    #end with open

    debug("largestFoldDiff: " + str(largestFoldDiff))
    debug("smallestFoldDiff: " + str(smallestFoldDiff))
    debug("numGenesWithLargestFoldDiff" + str(numGenesWithLargestFoldDiff))
    debug("numGenesWithSmallestFoldDiff" + str(numGenesWithSmallestFoldDiff))
    # debug("geneFoldValues (dictionary):\n" + str(geneFoldValues))

    # debug("genesWithOneFoldRatio (dictionary):\n" + str(genesWithOneFoldRatio))

    # debug("countFoldDiffPerRange (dictionary):\n" + str(countFoldDiffPerRange))

    out_file.writelines(resultLines)
    out_file_gnuplot.writelines(gnuplotLines)

    return geneFoldValues
#end computeFoldVals

geneFoldValuesMain = computeFoldValues(INPUT_FILE_NAME_TRAIN, OUTPUT_FILE_NAME_FOLD_VALUES,
OUTPUT_FILE_NAME_GENE_DISTRIBUTION)

```