Assignment 1

Jialin Kang

email: jkang58@jhu.edu

Question 1: Chromosome structures

The code used to solve this question:

```
import numpy as np
import pandas as pd

file_list = ['./ce10.chrom.sizes',
             './dm6.chrom.sizes',
             './ecoli.chrom.sizes',
             './hg38.chrom.sizes',
             './TAIR10.chrom.sizes',
             './tomato.chrom.sizes',
             './wheat.chrom.sizes',
             './yeast.chrom.sizes']

col_list = ['Total','Number', 'max_size', 'max_name', 'min_size', 'min_name', 'me
an']
row_list = ['ce10','dm6','ecoli','hg38','TAIR10','tomato','wheat','yeast']

def information(file_list, col_list, row_list):
    '''
    this function is used to produce a excel table of the chromosome information

    Parameters
    ----------------------
    file_list:list
    the files path
    col_list:list
    the table column name list of the excel table
    row_list:list
    the table row name list of the excel table
    ----------------------
    '''

    # open a df data frame
    df = pd.DataFrame(columns=row_list, index = col_list)

    value_total = []
    value_num = []
```

```python
    max_value = []
    max_value_name = []
    min_value = []
    min_value_name = []
    mean = []
    i=0
    for name in file_list:
        context = {}
        chrom = open(name, 'r')
        line = chrom.readline().replace('\n', '')
        while line != '':
            line_list = line.split()
            context[line_list[0]] = int(line_list[1])
            line = chrom.readline().replace('\n', '')
        value_list = []
        for value in context.values():
            value_list.append(value)

        # calculate the information of the chromosome of a species and put it int
o a list
        value_total = np.sum(value_list)
        value_num = len(value_list)
        max_value = np.max(value_list)
        max_value_name = max(context, key=context.get)
        min_value = np.min(value_list)
        min_value_name = min(context, key=context.get)
        mean = np.mean(value_list)
        arr_value = [value_total, value_num, max_value, max_value_name,
            min_value, min_value_name, mean]

        # write the list into dataframes
        df[row_list[i]] = arr_value
        i += 1


    df.to_excel('excel.xls')

if __name__ == "__main__":
    information(file_list, col_list, row_list)
```

the result table is save into an excel.xls file, the screen shoot of that file context is:

	ce10	dm6	ecoli	hg38	TAIR10	tomato	wheat	yeast
Total	1E+08	1.38E+08	4639211	3.09E+09	1.19E+08	7.83E+08	1.45E+10	12157105

Number	7	7	1	24	5	13	22	17
max_size	20924149	32079331	4639211	2.49E+08	30427671	90863682	8.31E+08	1531933
max_name	chrV	chr3R	Ecoli	chr1	Chr1	ch01	3B	chrIV
min_size	13794	1348131	4639211	46709983	18585056	9643250	4.74E+08	85779
min_name	chrM	chr4	Ecoli	chr21	Chr4	ch00	6D	chrM
mean	14326581	19649709	4639211	1.29E+08	23829270	60193849	6.61E+08	715123.8

Question 2: Sequence content

Question 2.1

The code used to solve this question is:

```
import sys

def fafile2dict():
    '''
    this function can be used to calculate the As, Cs, Gs, Ts in entire genome

    STDIN:
    ----------------------------------------
    the fasta file
    run as 'python3 ques1.py yeast.fa'
    ----------------------------------------

    Return:
    ----------------------------------------
    base_a:int
    the number of As
    base_c:int
    the number of Cs
    base_g:int
    the number of Gs
    base_t:int
    the number of Ts
    ----------------------------------------
    '''
    line = sys.stdin.readline().replace('\n','')
    seq = {}
    while line != '':
        if line[0] == '>':
            name = line.replace('\n','')
            seq[name] = ''
        else:
            seq[name] += line.replace('\n','').strip()
        line = sys.stdin.readline()
    base_a = 0
    base_t = 0
    base_c = 0
    base_g = 0
    for bp in seq.values():
        bp_list = list(bp)
        for bp_sort in bp_list:
            if bp_sort == 'A':
                base_a += 1
            elif bp_sort == 'T':
                base_t += 1
            elif bp_sort == 'C':
                base_c += 1
            else:
                base_g += 1

    print('A:',base_a,'T:',base_t,'C:',base_c,'G:',base_g)
    return base_a, base_t, base_c, base_g

if __name__ == "__main__":
    fafile2dict()
```

the result of this code is:

A: 3766349 T: 3753080 C: 2320576 G: 2317100

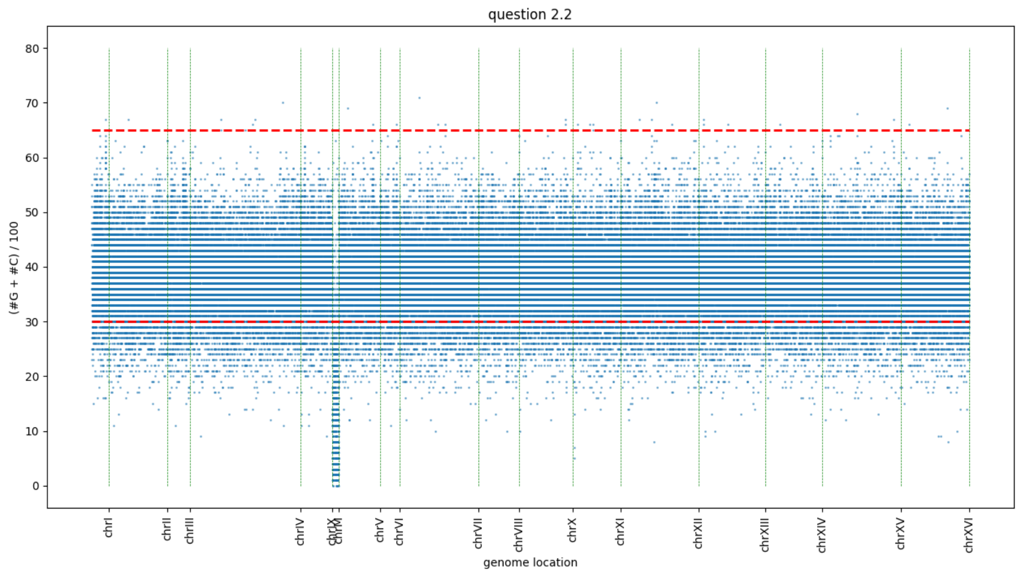
Question 2.2

```
import sys
import matplotlib.pyplot as plt

def fafile2dict():
    '''
    this function can be used to calculate the As, Cs, Gs, Ts in entire genome

    STDIN:
    ----------------------------------------
    the fasta file
    run as 'python3 ques1.py yeast.fa'
    ----------------------------------------
```

```python
Return:
----------------------------------------
base_a:int
the number of As
base_c:int
the number of Cs
base_g:int
the number of Gs
base_t:int
the number of Ts
----------------------------------------
'''
line = sys.stdin.readline().replace('\n','')
seq = {}
num = []
bp_list = []
n = 0
chrom_name = []
while line != '':
    if line[0] == '>':
        name = line.replace('\n','')
        seq[name] = ''
    else:
        seq[name] += line.replace('\n','').strip()
    line = sys.stdin.readline()

for chrom, bp in seq.items():
    bp_list += bp
    n = n + len(bp)/100
    num.append(n)
    chrom_name.append(chrom.replace('>',''))

count_list = []
bp_list = list(bp_list)
for i in range(int(len(bp_list)/100)):
    bp_frag = bp_list[100*i:(100*i+100)]
    base_gc = 0
    for bp_sort in bp_frag:
        if bp_sort == 'G' or bp_sort == 'C':
            base_gc += 1
    count_list.append(base_gc)

# draw the figure
plt.xlabel('genome location')
```

```
    plt.ylabel('(#G + #C) / 100')
    plt.title('question 2.2')
    plt.vlines(num, 0, 80, 'g', 'dashed', linewidths=0.5)
    plt.hlines([30,65], 0, len(count_list),'r', 'dashed', linewidths=2)
    plt.scatter(range(len(count_list)), count_list, s=1, alpha=0.5)
    plt.xticks(num, chrom_name, rotation=90)
    plt.show()


if __name__ == "__main__":
    fafile2dict()
```

the result of the question is:



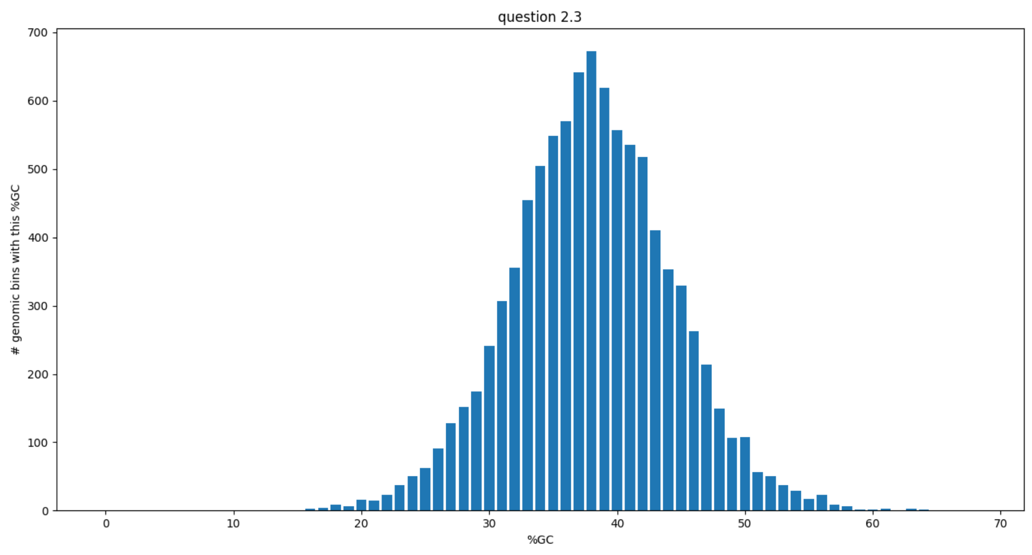
Question 2.3:

```
import sys
import matplotlib.pyplot as plt
import numpy as np

def fafile2dict():
    '''
    this function can be used to calculate the As, Cs, Gs, Ts in entire genome

    STDIN:
    ----------------------------------------
    the fasta file
```

```python
run as 'python3 ques1.py < yeast.fa'
----------------------------------------

Return:
----------------------------------------
base_a:int
the number of As
base_c:int
the number of Cs
base_g:int
the number of Gs
base_t:int
the number of Ts
----------------------------------------
'''
line = sys.stdin.readline().replace('\n','')
seq = {}
while line != '':
    if line[0] == '>':
        name = line.replace('\n','')
        seq[name] = ''
    else:
        seq[name] += line.replace('\n','').strip()
    line = sys.stdin.readline()
for bp in seq.values():
    bp_list = list(bp)
    count_list = []
    for i in range(int(len(bp_list)/100)):
        bp_frag = bp_list[(100*i-100):100*i]
        base_gc = 0
        for bp_sort in bp_frag:
            if bp_sort == 'G' or bp_sort == 'C':
                base_gc += 1
        count_list.append(base_gc)

count_set = set(count_list)

percentage = []
num_gc = []

for item in count_set:
    percentage.append(item)
    num_gc.append(count_list.count(item))
num_gc = list(map(int, num_gc))
percentage = list(map(int, percentage))
```

```
    plt.bar(percentage, num_gc)
    plt.xlabel("%GC")
    plt.ylabel("# genomic bins with this %GC")
    plt.title('question 2.3')
    plt.show()


if __name__ == "__main__":
    fafile2dict()
```

the result is:



Question2.4:

From the result of question2.2, we can know that chrM will sequence poorly, because there if lots of %GC is <= 30% or >= 65%.