**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', 'mean']

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 = []

    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 into 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():

    '''

    Make a scatterplot of the %GC of 100bp non-overlapping windows

    across the genome: x-axis = genome location, y-axis = (#G + #C) / 100.

    STDIN:

    ----------------------------------------

    the fasta file

    run as 'python3 ques2.py yeast.fa'

    ----------------------------------------

    '''

    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)

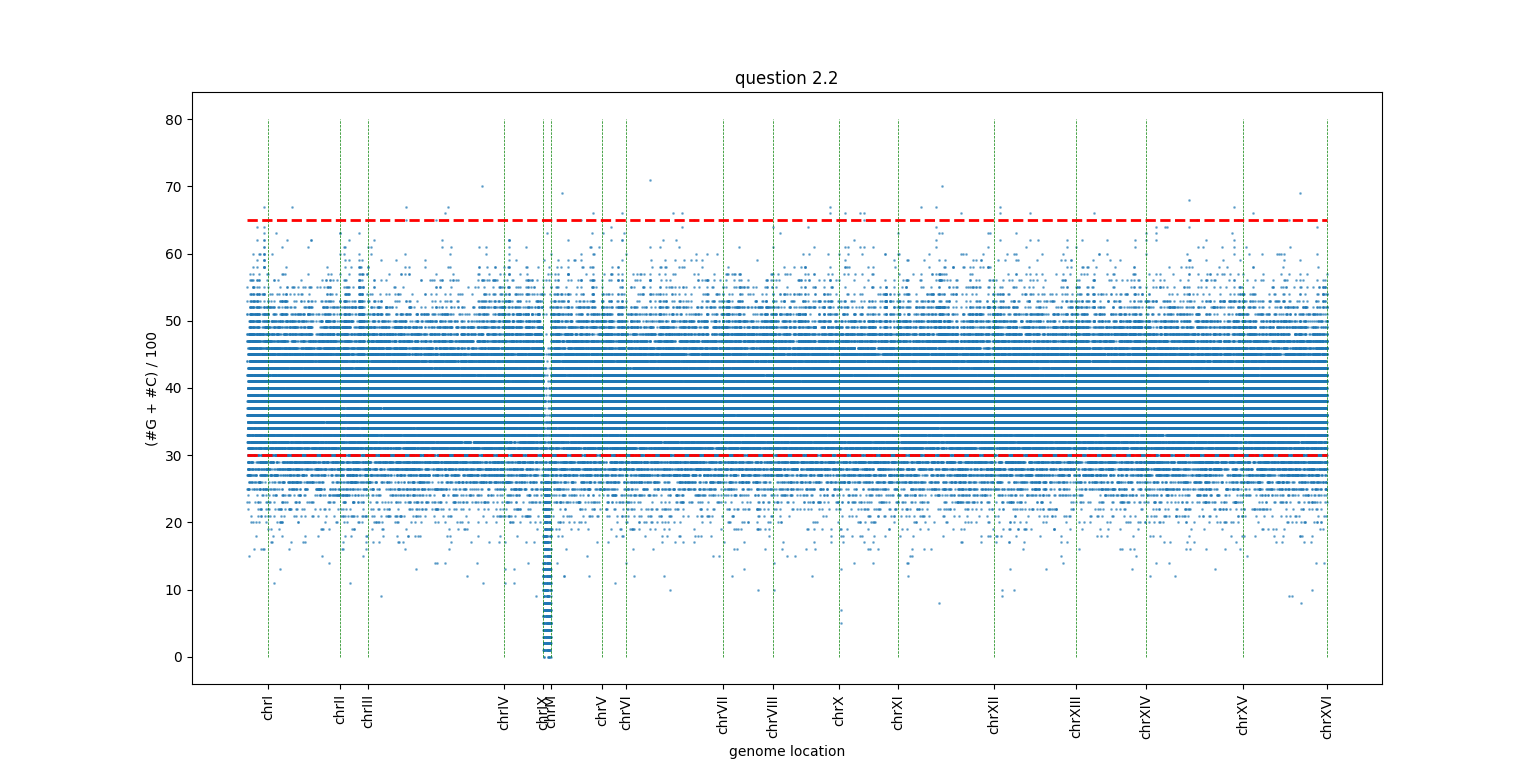
    plt.xticks(num, chrom\_name, rotation=90)

    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():

    '''

    Make a histogram of the number of genomic bins of a given %GC

    STDIN:

    ----------------------------------------

    the fasta file

    run as 'python3 ques3.py < yeast.fa'

    ----------------------------------------

    '''

    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()

    count\_list = []

    for bp in seq.values():

        bp\_list = list(bp)

        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")

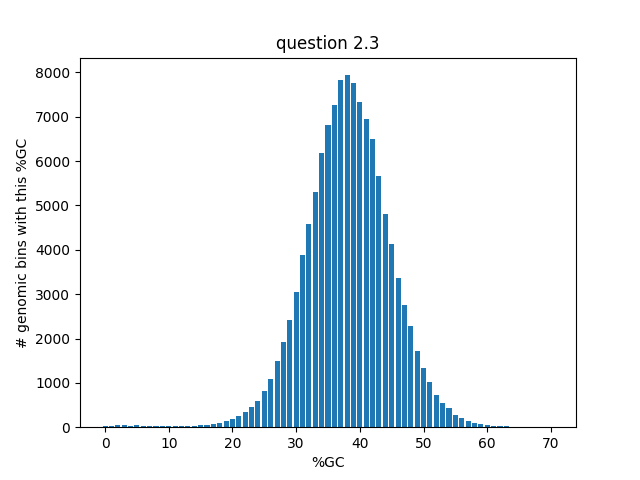
    plt.title('question 2.3')

    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 is lots of GC <= 30%.