**Supplementary File 2. Python code for sequence extraction for motif analysis.**

"""

Sequence Extraction for Motif Analysis

By Xiaoji Sun

This script extracts sequences from peak files to put into a fasta file and also generates a background fasta file containing ramdonly selected sequences from the whole genome.

This script accepts four inputs from commandlines and outputs a fasta file of

extracted sequences:

-g: sk1 genome sequence filename

-p: peaks.xls filename (from MACS)

-r: range of the region

-o: output fasta filename

-b: output filename of background sequences in fasta format

This scripts runs by the commandline:

e.g. python Sequence\_extraction.py -g SK1\_MvO\_V1\_\_\_GENOME/sk1\_MvO\_V1.fasta -p AH6407B\_P15\_peaks.xls -r 50 -o motif.fasta -b bg.fasta

"""

##################################################################################

# Modules

from Bio import SeqIO

import random

import optparse

##################################################################################

# Functions

def sequence\_extraction(sk1\_filename, peaks\_filename, range\_input, output\_filename, bg\_filename):

# reads the sk1 fasta file

sk1 = list(SeqIO.parse(sk1\_filename, 'fasta'))

# reads the peaks file

f = open(peaks\_filename, 'r')

# xls file

peaks = f.readlines()[24:]

f.close()

# puts the peaks data into a list of strings

for i in range(len(peaks)):

peaks[i] = peaks[i].strip().split('\t')

# calculates the summits positions and extract corresponding sequences

# summits in peaks xls file is off by two comparing to summits bed file

for i in range(len(peaks)):

for j in range(len(sk1)):

if peaks[i][0]==sk1[j].id:

peaks[i].append(str(sk1[j].seq[int(peaks[i][1]) + int(peaks[i][4])\

-2-range\_input-1 : int(peaks[i][1]) + int(peaks[i][4]) -2 + range\_input]))

# removes telomere regions (10kb)

peaks\_notelo = []

for i in range(len(peaks)):

for j in range(len(sk1)):

if peaks[i][0]==sk1[j].name and (int(peaks[i][1])+int(peaks[i][4]))> 10000 and (int(peaks[i][1])+int(peaks[i][4])) < (len(sk1[j].seq)-10000):

peaks\_notelo.append(peaks[i])

# sort the peaks by significance

#index = range(len(peaks\_notelo))

#pvalues = [float(i[6]) for i in peaks\_notelo]

#z = zip(pvalues,index)

#z.sort(reverse=True)

#index\_sig = [i[1] for i in z]

# sort the peaks by fold-enrichment

index = range(len(peaks\_notelo))

fe = [float(i[7]) for i in peaks\_notelo]

z = zip(fe, index)

z.sort(reverse=True)

index\_sig = [i[1] for i in z]

# outputs the sequences in a fasta format

output\_lines = []

for i in index\_sig:

output\_lines.append('>'+'peak'+str(i+1)+'-'+peaks\_notelo[i][0]+':'+\

str(int(peaks\_notelo[i][1])+int(peaks\_notelo[i][4])-2)+'-'+peaks\_notelo[i][7]+'\n'+peaks\_notelo[i][9])

f=open(output\_filename,'w')

f.write('\n'.join(i for i in output\_lines))

f.close()

# generates random sequences from the genome

bg\_sequences = []

for i in range(5\*len(peaks\_notelo)):

chr\_number = random.randint(1,16)

start\_number = random.randint(1,len(sk1[chr\_number-1].seq)-(range\_input\*2))

bg\_seq = sk1[chr\_number-1].seq[start\_number : start\_number+(range\_input\*2)]

bg\_sequences.append('>'+'bgseq'+str(i+1)+'\n'+str(bg\_seq))

# writes out background sequences

f=open(bg\_filename,'w')

f.write('\n'.join(i for i in bg\_sequences))

f.close()

###############################################################################

# Main

# parse object for managing input options.

parser = optparse.OptionParser()

# essential data, defines commanline options

parser.add\_option('-g', dest = 'sk1\_filename', default = '', help = 'This input\

is the fasta file of the sk1 genome sequences')

parser.add\_option('-p', dest = 'peaks\_filename', default = '', help = 'This input\

is the peaks.xls filename')

parser.add\_option('-r', dest = 'range\_input', default = '50', help = 'This input \

is the range on either side of summit position, default is 50(bp)')

parser.add\_option('-o', dest = 'output\_filename', default = '', help = 'This input \

is the output fasta filename')

parser.add\_option('-b', dest = 'bg\_filename', default = '', help = 'This input is the output background filename')

# loads the inputs

(options, args) = parser.parse\_args()

# reads the inputs from command lines

sk1\_filename = options.sk1\_filename

peaks\_filename = options.peaks\_filename

range\_input = int(options.range\_input)

output\_filename = options.output\_filename

bg\_filename = options.bg\_filename

# runs the function

sequence\_extraction(sk1\_filename, peaks\_filename, range\_input, output\_filename, bg\_filename)