**CS423 Lab 10: Finding Motifs**

**Authors: Sara Perkins, Caleb Piekstra**

**Write-up (65 points):**

Create a .doc or .docx file for the write-up and be sure your name(s) is/are in the file. Answer the questions below.

E-M Program:

1. (4 pts) What is the *best motif* you found when running findMotif 10 times?

**Profile:**

**1 2 3 4**

**A 0.0 0.0 1.0 0.0**

**C 0.0 0.5 0.0 0.0**

**G 0.0 0.25 0.0 1.0**

**T 1.0 0.25 0.0 0.0**

1. (4 pts) What are the best *instances* of the motif in each sequence?

**GTATACGATGTCTAGTATCAGCGGCATTAG TCAG**

**TAGCTGTACGTAGCGGCTTTAGCTGCAT TTAG**

**GACAGTCAGCGTTAGCTATATGCT TCAG**

**GCAGCAGTTGAGCAGCGATGATTTATCG TGAG**

**^ THESE GUYS**

1. (4 pts) When running findBestMotif (finds a motif 10,000 times), what is the best motif you found and what is its information content?

**1 2 3 4**

**A 1.0 0.001 0.001 0.001**

**C 0.001 0.001 1.0 0.001**

**G 0.001 1.0 0.001 1.0**

**T 0.001 0.001 0.001 0.001**

**Information content: 7.904410588584055 (8.0 technically since 0.001’s are 0.0)**

1. (4 pts) What are the best *instances* in each sequence of this best motif found in 10,000 runs?

**GTATACGATGTCTAGTATCAGCGGCATTAG AGCG**

**TAGCTGTACGTAGCGGCTTTAGCTGCAT AGCG**

**GACAGTCAGCGTTAGCTATATGCT AGCG**

**GCAGCAGTTGAGCAGCGATGATTTATCG AGCG**

**^ THESE GUYS**

Finding Motifs Upstream of the ModA Gene and its Orthologs:

1. (3 pts) Is the best instance of the motif *identical* for all nine bacteria organisms? If not, list the different instances found.

**No, all motif instances were unique. Note that the results excluded sequence 5: Shewanella putrefaciens. Our research indicates that it must have scored lower than some threshold that dictates whether the start site is good or bad.**

**TCCTTAGGGCTATGGA**

**TCCATTGGCTTATAGA**

**TCCATAGGCCTTTGGA**

**TCCATAGGCCTATAGG**

**TCCATAGGCCTATCGT**

**TTCATTGGCCTATGGA**

**TACATAGGGCTATGGA**

**TCCATTGGCCTATGGA**

1. (3 pts) What is the consensus sequence (the highest scoring sequence of the motif profile)?

**TCCATAGGCCTATGGA**

1. (3 pts) Do any of the organisms have substrings that match the consensus exactly? If so, which organisms?

**No.**

1. (4 pts) Does the consensus sequence exhibit a special property? (Hint: think palindromes for DNA)

**Yes, the complementary strand, AGGTATCCGGATACCT, is the same from right to left as the original from left to right. In other words, reading 5’ to 3’ on one strand matches reading from 5’ to 3’ on the complementary strand.**

1. (5 pts) Why might the motif have this property for bacteria?

**Bacteria only have one chromosome. Consequently, if this chromosome is damaged, the bacteria would normally have no way to repair it. However, if only one side of a chromosome is damaged, a palindromic DNA structure allows a chromosome to be self-repairing by bending over at the middle.**

**Source: https://en.wikipedia.org/wiki/Palindromic\_sequence**

1. (2 pts) What is the background distribution for A’s, T’s, C’s, and G’s used in this model?

**A: 0.242**

**T: 0.285**

**C: 0.240**

**G: 0.234**

1. (2 pts) What does this motif-finding program use for frequency of 0 (pseudo-counts)?

**0.001**

1. (3 pts) Include the text file of the gibbs sampler program in your zip file (name it gibbs\_modA\_results.txt).

Finding Motifs For Co-Expressed Yeast Genes:

1. (4 pts) For the 21 upstream regions of yeast genes, what is the consensus sequence of the length 10 motif that the gibbs motif sample found?

**AAAA-AAA-GCA**

1. (3 pts) Do any of the sequences match the length 10 motif exactly? If so, which genes?

**Yes (we considered them an exact match if all actual nucleotides in the motif (non ‘-‘) appeard in their respective places).**

**YAL040C**

**YBR202W**

**YER139C\_YER140W**

**YGR229C\_YGR230W**

**YGR229C\_YGR230W**

**YGR229C\_YGR230W**

1. (2 pts) Does the length 10 motif found have gaps? If so, which positions of the substrings are **not** included?

**Yes, at positions 5 and 9.**

1. (4 pts) What are the consensus sequences for two of the other motifs you found with motif width other than 10?

**Width 8: TT-TTTTT**

**Width 13: TTTCCTAAAAAGG**

1. (6 pts) Include the text files of the gibbs sample program in your zip file (name them gibbs\_yeast\_length10\_results.txt, gibbs\_yeast\_lengthXX\_results.txt, gibbs\_yeast\_lengthYY\_results.txt where XX and YY are the lengths you inputted to the system).

Comparing your yeast gene to the 21 yeast genes:

1. (5 pts) Does the upstream region of your yeast gene, chosen in lab 2, contain a substring similar to the length 10 motif? If so, what is the sequence?

**No.**

**Appendix A: Authorship (please include statement in your write-up)**

The write-up and code submitted for this lab was authored by the named person(s) on this lab report. All external sources to BIO/CS423 are cited properly.

**Appendix B: Code (20 points, based on correctness and style):**

Paste your python code EM.py here using Courier Font size 9 (and include it as a separate file in the .zip file you submit).

#!/usr/bin/python

# Lab10 CS423 fall 2015

# Expectation-Maximization Algorithm for Finding Motifs

# Sara Perkins, Caleb Piekstra

import random, math

# Suppose \*sequence\* is a DNA sequence, and suppose

# A1,C1,G1,T1,A2,C2,G2,T2,A3,C3,G3,T3,A4,C4,G4,T4

# represent a model for a motif of 4 nucleotides:

#

# 1 2 3 4

# A A1 A2 A3 A4

# C C1 C2 C3 C4

# G G1 G2 G3 G4

# T T1 T2 T3 T4

#

# This function calculates and returns the 4-mer

# in \*sequence\* that best matches the motif model.

#

# Tammy apologizes for the long list of parameters, but

# this is the clearest way for you to know which frequency

# is which, instead of putting the data in a 2D list

# complete this function

def getBestMatchInSequence(sequence, A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4):

A = [A1, A2, A3, A4]

C = [C1, C2, C3, C4]

G = [G1, G2, G3, G4]

T = [T1, T2, T3, T4]

# holds the best sequence

bestseq = ""

# holds the score of the best sequence

bestscore = 0.0

# loop through the sequence and determine which 4-mer has the best score

# by multiplying the probabilties of each nuc ocurring in it's position

for i in range(0, len(sequence)-3):

subseq = sequence[i:i+4]

score = 1.0

for idx, nuc in enumerate(subseq):

if nuc == 'A': score \*= A[idx]

elif nuc == 'C': score \*= C[idx]

elif nuc == 'G': score \*= G[idx]

else: score \*= T[idx]

# keep track of the highest scoring sequence

if score > bestscore:

bestseq = subseq

bestscore = score

# return the best found sequence or emptystring if none (shouldn't happen)

return bestseq

# Calculates and returns the frequency of the given

# nucleotide, nt, at given position in four sequences.

# If frequency is 0, return .01 (so info content calculation is

# well-defined)

# complete this function

def getFrequencyOfNucleotideAtPosition(nt, position, seq1, seq2, seq3, seq4):

sequences = [seq1, seq2, seq3, seq4]

count = 0

# loop through and count the occurrences of the nuc

for seq in sequences:

if seq[position] == nt:

count += 1

frequency = count/4.0

# Default to 0.001 so that math.log works if the

# sequence has a frequency of 0

if frequency == 0.0:

return 0.001

return frequency

# Calculates and returns the information content of

# a 4-mer motif profile, given the 16 values in the profile

# Assumes background frequency of 25% for each nucleotide

# complete this function

def calcInfoContent(A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4):

prof = [[A1, C1, G1, T1], [A2, C2, G2, T2], [A3, C3, G3, T3], [A4, C4, G4, T4]]

# calculates I for each position (1-4) and then sums all of those I's

return sum([sum([weight \* math.log(weight/.25, 2) for weight in col]) for col in prof])

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

# Prints the specified motif model to the screen.

# 1 2 3 4

# A A1 A2 A3 A4

# C C1 C2 C3 C4

# G G1 G2 G3 G4

# T T1 T2 T3 T4

#

# (function completed for you)

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

def printMotif(A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4):

print ("\t 1 \t 2 \t 3 \t 4")

print ("A\t" + str(A1) + "\t" + str(A2) + "\t" + str(A3) + "\t" + str(A4))

print ("C\t" + str(C1) + "\t" + str(C2) + "\t" + str(C3) + "\t" + str(C4))

print ("G\t" + str(G1) + "\t" + str(G2) + "\t" + str(G3) + "\t" + str(G4))

print ("T\t" + str(T1) + "\t" + str(T2) + "\t" + str(T3) + "\t" + str(T4) + "\n")

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

# Uses expectation-maximization algorithm for finding

# the best motif in the four sequences

#

# (function completed for you)

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

def findMotif(seq1, seq2, seq3, seq4):

# remember the motif instances from the previous iteration so we know

# when algorithm converges

old\_instance1 = ""

old\_instance2 = ""

old\_instance3 = ""

old\_instance4 = ""

# randomly choose starts of 4-mers from each sequence

randomStart1 = random.randint(0, len(seq1)-4)

randomStart2 = random.randint(0, len(seq2)-4)

randomStart3 = random.randint(0, len(seq3)-4)

randomStart4 = random.randint(0, len(seq4)-4)

# create the random 4-mers from each sequence

instance1 = seq1[randomStart1:randomStart1+4]

instance2 = seq2[randomStart2:randomStart2+4]

instance3 = seq3[randomStart3:randomStart3+4]

instance4 = seq4[randomStart4:randomStart4+4]

# repeat two steps of EM until convergence.

while(old\_instance1 != instance1 or old\_instance2 != instance2 or old\_instance3 != instance3 or old\_instance4 != instance4):

# calculate a motif model for the 4 motif instances (step 1 of EM)

A1 = getFrequencyOfNucleotideAtPosition("A", 0, instance1, instance2, instance3, instance4)

C1 = getFrequencyOfNucleotideAtPosition("C", 0, instance1, instance2, instance3, instance4)

G1 = getFrequencyOfNucleotideAtPosition("G", 0, instance1, instance2, instance3, instance4)

T1 = getFrequencyOfNucleotideAtPosition("T", 0, instance1, instance2, instance3, instance4)

A2 = getFrequencyOfNucleotideAtPosition("A", 1, instance1, instance2, instance3, instance4)

C2 = getFrequencyOfNucleotideAtPosition("C", 1, instance1, instance2, instance3, instance4)

G2 = getFrequencyOfNucleotideAtPosition("G", 1, instance1, instance2, instance3, instance4)

T2 = getFrequencyOfNucleotideAtPosition("T", 1, instance1, instance2, instance3, instance4)

A3 = getFrequencyOfNucleotideAtPosition("A", 2, instance1, instance2, instance3, instance4)

C3 = getFrequencyOfNucleotideAtPosition("C", 2, instance1, instance2, instance3, instance4)

G3 = getFrequencyOfNucleotideAtPosition("G", 2, instance1, instance2, instance3, instance4)

T3 = getFrequencyOfNucleotideAtPosition("T", 2, instance1, instance2, instance3, instance4)

A4 = getFrequencyOfNucleotideAtPosition("A", 3, instance1, instance2, instance3, instance4)

C4 = getFrequencyOfNucleotideAtPosition("C", 3, instance1, instance2, instance3, instance4)

G4 = getFrequencyOfNucleotideAtPosition("G", 3, instance1, instance2, instance3, instance4)

T4 = getFrequencyOfNucleotideAtPosition("T", 3, instance1, instance2, instance3, instance4)

# print the motif model to the screen (comment out when running findBestMotif)

#printMotif(A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4)

# re-assign old instances as the current instances to determine convergence

old\_instance1 = instance1

old\_instance2 = instance2

old\_instance3 = instance3

old\_instance4 = instance4

# find best match in each sequence (step 2 of EM)

instance1 = getBestMatchInSequence(seq1, A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4)

instance2 = getBestMatchInSequence(seq2, A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4)

instance3 = getBestMatchInSequence(seq3, A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4)

instance4 = getBestMatchInSequence(seq4, A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4)

# print out each of the 4 sequences and instances (comment out when running findBestMotif)

## print (seq1 + "\t" + instance1)

## print (seq2 + "\t" + instance2)

## print (seq3 + "\t" + instance3)

## print (seq4 + "\t" + instance4 + "\n")

return [A1, C1, G1, T1, A2, C2, G2, T2, A3, C3, G3, T3, A4, C4, G4, T4]

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

# Runs findMotif 10000 times and returns the motif

# with the highest information content

# When calling this function, be sure to comment out

# printing in the findMotif function

# (function completed for you)

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

def findBestMotif(seq1, seq2, seq3, seq4):

bestInfoContent = 0.0

bestModel = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] #stored 2D table as a 1-dimensional list

for i in range(10000):

m = findMotif(seq1, seq2, seq3, seq4)

infoContent = calcInfoContent(m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

# keep best motif found so far

if infoContent > bestInfoContent:

bestModel = m

bestInfoContent = infoContent

# display information about best motif

m = bestModel

# print best motif

printMotif(m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

# find best match in each sequence

instance1 = getBestMatchInSequence(seq1, m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

instance2 = getBestMatchInSequence(seq2, m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

instance3 = getBestMatchInSequence(seq3, m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

instance4 = getBestMatchInSequence(seq4, m[0], m[1], m[2], m[3], m[4], m[5], m[6], m[7], m[8], m[9], m[10], m[11], m[12], m[13], m[14], m[15])

# print out each of the 4 sequences and instances

print (seq1 + "\t" + instance1)

print (seq2 + "\t" + instance2)

print (seq3 + "\t" + instance3)

print (seq4 + "\t" + instance4 + "\n")

# print info content

print ("Information content: " + str(bestInfoContent))

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

# Run functions here #

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

# search for 4-mer (length 4) motif in the following 4 sequences

seq1 = "GTATACGATGTCTAGTATCAGCGGCATTAG"

seq2 = "TAGCTGTACGTAGCGGCTTTAGCTGCAT"

seq3 = "GACAGTCAGCGTTAGCTATATGCT"

seq4 = "GCAGCAGTTGAGCAGCGATGATTTATCG"

findBestMotif(seq1, seq2, seq3, seq4)

# findBestMotif(seq1, seq2, seq3, seq4) # be sure to comment out printing when running findBestMotif