Bioinformatics Algorithms:

HW II

**Important Notes**

1. What to submit:
   1. To Canvas: This document in PDF or Word format with answers typed. Only one group member should upload.
   2. **For groups**: Submit group member and self-evaluation form: **Each group member must submit this form** to give score for themselves and their fellow group members about their contribution to the project. The form is available at https://forms.office.com/r/kKTUx62gm7
2. Answer questions by typing your answers below each question. Please do not delete the questions.
3. Remember to write your name(s) to the header above,
4. Note academic integrity policy reminder below.

**URL to Code:** <https://colab.research.google.com/drive/1qGUL_R2lbAZJEBUPou9mH4VMQ29CtO6e?usp=sharing>

1. **Give sequences where (4,1)-motif GTAG is planted to each sequence (shown in bold).**

AC**GTAA**CACCGT

TG**GTAC**GTCCCA

C**GTCG**CTCGTGA

CC**GAAG**GGTACG

CGAAC**GTAT**ACG

1. **(5 pts.) Compute the profile using the first k-mers in each sequence (utilize pseudocounts).**

k-mers:  
GTAA  
GTAC  
GTCG  
GAAG  
GTAT

Counts:  
A: 0 1 4 1  
C: 0 0 1 1  
G: 5 0 0 2  
T: 0 4 0 1

Counts + Pseudocounts:  
A: 1 2 5 2  
C: 1 1 2 2  
G: 6 1 1 3  
T: 1 5 1 2

Column Sum(s): 9

Profile:  
A: 0.111 0.222 0.556 0.222  
C: 0.111 0.111 0.222 0.222  
G: 0.667 0.111 0.111 0.333  
T: 0.111 0.556 0.111 0.222

1. **(5 pts.) Compute the entropy of each column in the profile computed in (a).**

*H*(*p*1, …, *pN*) = -∑i=1*Npi*\*log2(*pi*)

Log2(0.111) = -3.171  
Log2(0.222) = -2.171  
Log2(0.333) = -1.586  
Log2(0.556) = -0.847  
Log2(0.667) = -0.584

Column 1:

-1 \* (0.111 \* Log2(0.111) + 0.111 \* Log2(0.111) + 0.667 \* Log2(0.667) + 0.111 \* Log2(0.111)) =

-1 \* (0.111 \* -3.171 + 0.111 \* -3.171 + 0.667 \* -0.584 + 0.111 \* -3.171) =

1.445

Column 2:

-1 \* (0.222 \* Log2(0.222) + 0.111 \* Log2(0.111) + 0.111 \* Log2(0.111) + 0.556 \* Log2(0.556)) =

-1 \* (0.222 \* -2.171 + 0.111 \* -3.171 + 0.111 \* -3.171 + 0.556 \* -0.584) =

1.511

Column 3:

-1 \* (0.556 \* Log2(0.556) + 0.222 \* Log2(0.222) + 0.111 \* Log2(0.111) + 0.111 \* Log2(0.111)) =

-1 \* (0.556 \* -0.847 + 0.222 \* -2.171 + 0.111 \* -3.171 + 0.111 \* -3.171) =

1.657

Column 4:

-1 \* (0.222 \* Log2(0.222) + 0.222 \* Log2(0.222) + 0.333 \* Log2(0.333) + 0.222 \* Log2(0.222)) =

-1 \* (0.222 \* -2.171 + 0.222 \* -2.171 + 0.333 \* -1.586 + 0.222 \* -2.171) =

1.974

1. **(10 pts.) Using the profile computed in (a), find the most probable k-mer in each sequence. Show your calculations.**

ACGTAACACCGT:

ACGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGTA

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

GTAA

0.667 \* 0.556 \* 0.556 \* 0.222 = 0.045775

TAAC

0.111 \* 0.222 \* 0.556 \* 0.222 = 0.003042

AACA

0.111 \* 0.222 \* 0.222 \* 0.222 = 0.001215

ACAC

0.111 \* 0.111 \* 0.556 \* 0.222 = 0.001521

CACC

0.111 \* 0.222 \* 0.222 \* 0.222 = 0.001215

ACCG

0.111 \* 0.111 \* 0.222 \* 0.333 = 0.000911

CCGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

Best k-mer: ACGT

TGGTACGTCCCA:

TGGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

GGTA

0.667 \* 0.111 \* 0.111 \* 0.222 = 0.001824

GTAC

0.667 \* 0.556 \* 0.556 \* 0.222 = 0.045775

TACG

0.111 \* 0.222 \* 0.222 \* 0.333 = 0.001822

ACGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGTC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

GTCC

0.667 \* 0.556 \* 0.222 \* 0.222 = 0.018277

TCCC

0.111 \* 0.111 \* 0.222 \* 0.222 = 0.000607

CCCA

0.111 \* 0.111 \* 0.222 \* 0.222 = 0.000607

Best k-mer: TGGT

CGTCGCTCGTGA:

CGTC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

GTCG

0.667 \* 0.556 \* 0.222 \* 0.333 = 0.027416

TCGC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGCT

0.111 \* 0.111 \* 0.222 \* 0.222 = 0.000607

GCTC

0.667 \* 0.111 \* 0.111 \* 0.222 = 0.001824

CTCG

0.111 \* 0.556 \* 0.222 \* 0.333 = 0.004562

TCGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGTG

0.111 \* 0.111 \* 0.111 \* 0.333 = 0.000456

GTGA

0.667 \* 0.556 \* 0.111 \* 0.222 = 0.009139

Best k-mer: CGTC

CCGAAGGGTACG:

CCGA

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGAA

0.111 \* 0.111 \* 0.556 \* 0.222 = 0.001521

GAAG

0.667 \* 0.222 \* 0.556 \* 0.333 = 0.027416

AAGG

0.111 \* 0.222 \* 0.111 \* 0.333 = 0.000911

AGGG

0.111 \* 0.111 \* 0.111 \* 0.333 = 0.000456

GGGT

0.667 \* 0.111 \* 0.111 \* 0.222 = 0.001824

GGTA

0.667 \* 0.111 \* 0.111 \* 0.222 = 0.001824

GTAC

0.667 \* 0.556 \* 0.556 \* 0.222 = 0.045775

TACG

0.111 \* 0.222 \* 0.222 \* 0.333 = 0.001822

Best k-mer: CCGA

CGAACGTATACG:

CGAA

0.111 \* 0.111 \* 0.556 \* 0.222 = 0.001521

GAAC

0.667 \* 0.222 \* 0.556 \* 0.222 = 0.018277

AACG

0.111 \* 0.222 \* 0.222 \* 0.333 = 0.001822

ACGT

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

CGTA

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

GTAT

0.667 \* 0.556 \* 0.556 \* 0.222 = 0.045775

TATA

0.111 \* 0.222 \* 0.111 \* 0.222 = 0.000607

ATAC

0.111 \* 0.556 \* 0.556 \* 0.222 = 0.007618

TACG

0.111 \* 0.222 \* 0.222 \* 0.333 = 0.001822

Best k-mer: ACGT

1. **(5 pts.) Compute the score of the selected motifs in (c).**

k-mers:

ACGT

TGGT

CGTC

CCGA

ACGT

Consensus: AGGT

Score: 3 + 2 + 1 + 2 = 8

1. **(10 pts.) Repeat a-d for one more iteration and report the final list of motifs you got in each sequence. Show your work.**

k-mers:

ACGT

TGGT

CGTC

CCGA

ACGT

Counts:

A: 2 0 0 1

C: 2 3 0 1

G: 0 2 4 0

T: 1 0 1 3

Counts + Pseudocounts:

A: 3 1 1 2

C: 3 4 1 2

G: 1 3 5 1

T: 2 1 2 4

Column Sum(s): 9

Profile:

A: 0.333 0.111 0.111 0.222

C: 0.333 0.444 0.111 0.222

G: 0.111 0.333 0.556 0.111

T: 0.222 0.111 0.222 0.444

*H*(*p*1, …, *pN*) = -∑i=1*Npi*\*log2(*pi*)

Log2(0.111) = -3.171  
Log2(0.222) = -2.171  
Log2(0.333) = -1.586

Log2(0.444) = -1.171  
Log2(0.556) = -0.847  
Log2(0.667) = -0.584

Column 1:

-1 \* (0.333 \* Log2(0333.) + 0.333 \* Log2(0.333) + 0.111 \* Log2(0.111) + 0.222 \* Log2(0.222)) =

-1 \* (0.333 \* -1.586 + 0.333 \* -1.586 + 0.111 \* -3.171 + 0.222 \* -2.171) =

1.890

Column 2:

-1 \* (0.111 \* Log2(0.111) + 0.444 \* Log2(0.444) + 0.333 \* Log2(0.333) + 0.111 \* Log2(0.111)) =

-1 \* (0.111 \* -3.171 + 0.444 \* -1.171 + 0.333 \* -1.586 + 0.111 \* -3.171) =

1.752

Column 3:

-1 \* (0.111 \* Log2(0.111) + 0.111 \* Log2(0.111) + 0.556 \* Log2(0.556) + 0.222 \* Log2(0.222)) =

-1 \* (0.111 \* -3.171 + 0.111 \* -3.171 + 0.556 \* -0.847 + 0.222 \* -2.171) =

1.656

Column 4:

-1 \* (0.222 \* Log2(0.222) + 0.222 \* Log2(0.222) + 0.111 \* Log2(0.111) + 0.444 \* Log2(0.444)) =

-1 \* (0.222 \* -2.171 + 0.222 \* -2.171 + 0.111 \* -3.171 + 0.444 \* -1.171) =

1.836

ACGTAACACCGT:

ACGT

0.333 \* 0.444 \* 0.556 \* 0.444 = 0.036499

CGTA

0.333 \* 0.333 \* 0.222 \* 0.222 = 0.005465

GTAA

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

TAAC

0.222 \* 0.111 \* 0.111 \* 0.222 = 0.000607

AACA

0.333 \* 0.111 \* 0.111 \* 0.222 = 0.000911

ACAC

0.333 \* 0.444 \* 0.111 \* 0.222 = 0.003643

CACC

0.333 \* 0.111 \* 0.111 \* 0.222 = 0.000911

ACCG

0.333 \* 0.444 \* 0.111 \* 0.111 = 0.001822

CCGT

0.333 \* 0.444 \* 0.556 \* 0.444 = 0.036499

Best k-mer: GTAA

TGGTACGTCCCA:

TGGT

0.222 \* 0.333 \* 0.556 \* 0.444 = 0.01825

GGTA

0.111 \* 0.333 \* 0.222 \* 0.222 = 0.001822

GTAC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

TACG

0.222 \* 0.111 \* 0.111 \* 0.111 = 0.000304

ACGT

0.333 \* 0.444 \* 0.556 \* 0.444 = 0.036499

CGTC

0.333 \* 0.333 \* 0.222 \* 0.222 = 0.005465

GTCC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

TCCC

0.222 \* 0.444 \* 0.111 \* 0.222 = 0.002429

CCCA

0.333 \* 0.444 \* 0.111 \* 0.222 = 0.003643

Best k-mer: GTAC

CGTCGCTCGTGA:

CGTC

0.333 \* 0.333 \* 0.222 \* 0.222 = 0.005465

GTCG

0.111 \* 0.111 \* 0.111 \* 0.111 = 0.000152

TCGC

0.222 \* 0.444 \* 0.556 \* 0.222 = 0.012166

CGCT

0.333 \* 0.333 \* 0.111 \* 0.444 = 0.005465

GCTC

0.111 \* 0.444 \* 0.222 \* 0.222 = 0.002429

CTCG

0.333 \* 0.111 \* 0.111 \* 0.111 = 0.000455

TCGT

0.222 \* 0.444 \* 0.556 \* 0.444 = 0.024333

CGTG

0.333 \* 0.333 \* 0.222 \* 0.111 = 0.002732

GTGA

0.111 \* 0.111 \* 0.556 \* 0.222 = 0.001521

Best k-mer: GTCG

CCGAAGGGTACG:

CCGA

0.333 \* 0.444 \* 0.556 \* 0.222 = 0.01825

CGAA

0.333 \* 0.333 \* 0.111 \* 0.222 = 0.002733

GAAG

0.111 \* 0.111 \* 0.111 \* 0.111 = 0.000152

AAGG

0.333 \* 0.111 \* 0.556 \* 0.111 = 0.002281

AGGG

0.333 \* 0.333 \* 0.556 \* 0.111 = 0.006844

GGGT

0.111 \* 0.333 \* 0.556 \* 0.444 = 0.009125

GGTA

0.111 \* 0.333 \* 0.222 \* 0.222 = 0.001822

GTAC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

TACG

0.222 \* 0.111 \* 0.111 \* 0.111 = 0.000304

Best k-mer: GAAG

CGAACGTATACG:

CGAA

0.333 \* 0.333 \* 0.111 \* 0.222 = 0.002733

GAAC

0.111 \* 0.111 \* 0.111 \* 0.222 = 0.000304

AACG

0.333 \* 0.111 \* 0.111 \* 0.111 = 0.000455

ACGT

0.333 \* 0.444 \* 0.556 \* 0.444 = 0.036499

CGTA

0.333 \* 0.333 \* 0.222 \* 0.222 = 0.005465

GTAT

0.111 \* 0.111 \* 0.111 \* 0.444 = 0.000607

TATA

0.222 \* 0.111 \* 0.222 \* 0.222 = 0.001215

ATAC

0.333 \* 0.111 \* 0.111 \* 0.222 = 0.000911

TACG

0.222 \* 0.111 \* 0.111 \* 0.111 = 0.000304

Best k-mer: GAAC

1. pl

k-mers:

GTAA

GTAC

GTCG

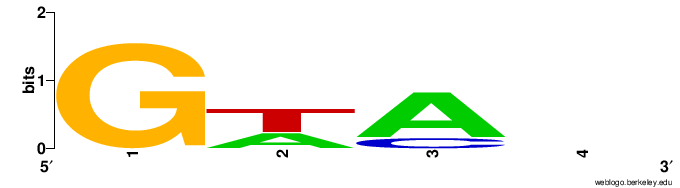
GAAG

GAAC

Consensus: GTAC

Score: 0 + 2 + 1 + 3 = 6

1. **(5 pts.) Compute the motif logo of the motifs found in (e) using the webtool at** [**https://weblogo.berkeley.edu/logo.cgi**](https://weblogo.berkeley.edu/logo.cgi)



1. **A count matrix is given below**

**A [ 10 12 4 1 2 2 0 0 0 8 13 ]**

**C [ 2 2 7 1 0 8 0 0 1 2 2 ]**

**G [ 3 1 1 0 23 0 26 26 0 0 4 ]**

**T [ 11 11 14 24 1 16 0 0 25 16 7 ]**

1. **(5 pts.) Compute the profile using pseudocounts.**

Matrix + Pseudocounts:

A [ 11 13 5 2 3 3 1 1 1 9 14 ]

C [ 3 3 8 2 1 9 1 1 2 3 3 ]

G [ 4 2 2 1 24 1 27 27 1 1 5 ]

T [ 12 12 15 25 2 17 1 1 26 17 8 ]

Column Sum(s): 30

Profile:

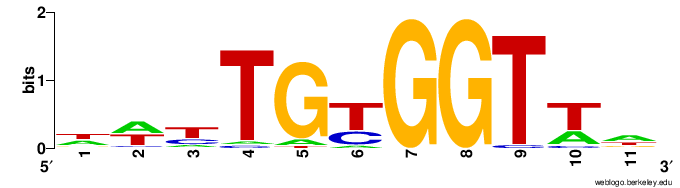
A [0.367, 0.433, 0.167, 0.067, 0.1, 0.1, 0.033, 0.033, 0.033, 0.3, 0.46]

C [0.1, 0.1, 0.267, 0.067, 0.033, 0.3, 0.033, 0.033, 0.067, 0.1, 0.]

G [0.133, 0.067, 0.067, 0.033, 0.8, 0.033, 0.9, 0.9, 0.033, 0.033, 0.16]

T [0.4, 0.4, 0.5, 0.833, 0.067, 0.567, 0.033, 0.033, 0.867, 0.567, 0.26]

1. **(5 pts.) Visualize the motif logo using the webtool at** [**https://weblogo.berkeley.edu/logo.cgi**](https://weblogo.berkeley.edu/logo.cgi)



1. **(25 pts.) Implement RandomizedMotifSearch on Rosalind and run it on the sequences in the attached file to compute motifs for k=11.**

Code:

import random

# Get profile given motifs

def get\_profile(motifs):

profile = []

# Get counts

for i in range(len(motifs[0])):

s = 0

profile.append({"A": 0, "C": 0, "G": 0, "T": 0})

for m in motifs:

profile[-1][m[i]] += 1

s += 1

# Add pseudocounts

for b in profile[-1]:

profile[-1][b] += 1

s += 1

# Convert to probabilities

for b in profile[-1]:

profile[-1][b] /= s

return profile

# Get the profile most probable motif

def get\_profile\_most\_probable\_motif(seq, profile):

max\_p = -1

max\_m = None

k = len(profile)

# Iterate over all possible motifs

for i in range(len(seq) - k + 1):

motif = seq[i:i+k]

# Calculate probability of current motif

prob = 1

for j, b in enumerate(motif):

prob \*= profile[j][b]

# Keep best probability

if prob > max\_p:

max\_p = prob

max\_m = motif

return max\_m

def get\_motifs(profile, dna):

motifs = []

for seq in dna:

motifs.append(get\_profile\_most\_probable\_motif(seq, profile))

return motifs

def score\_motifs(motifs):

score = 0

# Iterate over positions

for i in range(len(motifs[0])):

counter = {"A": 0, "C": 0, "G": 0, "T": 0}

max\_c = -1

max\_b = None

# Iterate over motifs

for j, m in enumerate(motifs):

# Get count of base at position

counter[m[i]] += 1

# Keep highest count base

if counter[m[i]] > max\_c:

max\_c = counter[m[i]]

max\_b = m[i]

# Sum non-max counts

for b in counter:

if b != max\_b:

score += counter[b]

return score

def randomized\_motif\_search(dna, k, t):

# Get inital motifs

motifs = []

for seq in dna:

i = random.randint(0, len(seq) - k)

motifs.append(seq[i:i+k])

best\_motifs = motifs

# Iterate until score stops changing

while True:

# Get new profile and motifs

profile = get\_profile(motifs)

motifs = get\_motifs(profile, dna)

# Check score

if score\_motifs(best\_motifs) > score\_motifs(motifs):

best\_motifs = motifs

else:

return best\_motifs

def multi\_randomized\_motif\_search(dna, k, t, n):

# Iterate n times

best\_motifs = None

for i in range(n):

# Get motifs

motifs = randomized\_motif\_search(dna, k, t)

# Check for new best motifs

if best\_motifs is not None:

if score\_motifs(best\_motifs) > score\_motifs(motifs):

best\_motifs = motifs

else:

best\_motifs = motifs

return best\_motifs

motifs = []

with open("sequences.txt", "r") as file:

content = file.readlines()

line = content[0].strip().split(" ")

k = 11

t = len(content)

dna = [l.strip() for l in content]

motifs = multi\_randomized\_motif\_search(dna, k, t, 1000)

for m in motifs:

print(m)

Output:

ATCTGGGGATG

CTTCGCCATTA

AACAGCCGTTA

ATCAGAGGATA

ATCGAAGATGA

AACTACGCTTA

ATGTGGGATTC

ATGTGCGGACA

ATAGGAGCTTA

ATATGTCGTTA

ATCTAGGGTTA

ATCAGTCATTA

ATGTGCGCTTT

CTCTGCGCGTA

ATTAGACCATA

ATCTGCCCATA

ATCTACGCTCA

ATCCGCCGTTG

ATATGCTGTTG

CCCTGTGGTTA

1. **(20 pts.) Implement MedianString algorithm on Rosalind and run it on the same sequences in (c) to compute motifs for k=11.**

Code:

import itertools

# Get the hamming distance of 2 sequences

def hamming(txt1, txt2):

dis = 0

for i in range(len(txt1)):

if txt1[i] != txt2[i]:

dis += 1

return dis

# Get all possible k-mers of size i

def gen\_combos(k, k\_mers=None):

return ["".join(p) for p in itertools.product(["A", "C", "G", "T"], repeat=k)]

# Get minimum distance from pattern of a k-mer in a sequence

def d\_text(seq, pat):

min\_d = 2\*len(seq)

for i in range(len(seq) - len(pat) + 1):

txt = seq[i:i+len(pat)]

d = hamming(pat, txt)

if d < min\_d:

min\_d = d

return min\_d

# Sum all minimum distances in dna

def d\_dna(dna, pat):

sum\_d = 0

for seq in dna:

d = d\_text(seq, pat)

sum\_d += d

return sum\_d

def min\_d\_dna(dna, k):

min\_d = 1000000\*len(dna[0])

min\_p = None

# Iterate over every possible k-mer

for pat in gen\_combos(k):

# Get score of pattern in dna

d = d\_dna(dna, pat)

# Check for new best

if d < min\_d:

min\_d = d

min\_p = pat

return min\_p

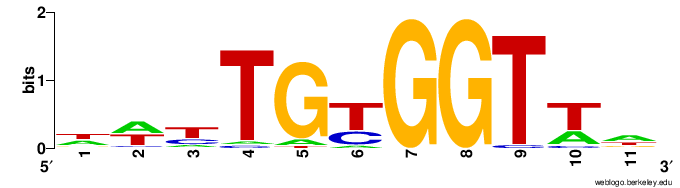
k = 11

print(min\_d\_dna(motifs, k))

Output:

ATCTGCGGTTA

1. **(5 pts.) Compute the motif logo of the motifs found in (c) and (d). Compare both motif logo to the motif logo found in (b). Interpret the results. Which motif logo is more similar?**





1. ATCTGCGGTTA



Comparison: The motif logo produced in C has none of the high bit level nucleotides in B, and at those locations the C logo does not have a dominating score by any nucleotide.The mid-level scoring nucleotides in B largely have the same nucleotide as the highest score in C, though only in one case is the score equal or better. For most of the low-scoring locations in B’s motif logo the scores in C’s logo are both higher and more decisive with one exception. Comparing the median string obtained in D shows that for indices of high deciciveness the nucleotides in the median string match the best scoring nucleotide at that location. For the other locations one of the best scoring nucleotides was selected in the median string.

Interpretation: The motif logo produced in B has a general agreement with the motif logo produced in C, though not a strong one. While on average the scores are about the same B’s logo has about half of its locations that are strongly enough predicted that this motif is probably a lot better for downstream tasks. I would say that the median string is a good interpretation of the motif logo produced in B.

Similarity: I would say that the C result looks the most similar to B,as the D result produced a single motif, resulting in an empty graph. However, I would say that based on matching selected nucleotides, the results of D are better.

**Important Reminder on Academic Integrity Policy**

**This assignment must be done in a group up to 2 people. All solutions/writing must be in your own words. Make sure to cite your resources for questions that require any research.**

Academic honesty policy is strictly enforced in this class. Violators of this policy will be reported to the Office of the Provost and the Office for Student Success. There are no exceptions. Please refer to the course syllabus for the academic honesty policy for more details. If you have any questions about this policy, please contact the instructor. **All submissions will be scanned by Turnitin.**