DNA Motif Alignment and Recognition Bioinformatics Pipeline Manual

Introduction

This pipeline aligns together sequences that have a motif and creates a nucleotide probability matrix, also called a position weight matrix. This matrix is then entered into a DNA energy normalized logo generator (an example utilization will be posted below in the final step).

The execution of this pipeline represent the obtainment or creation of the following information:

I. Encoding all *.fasta* files of a data set directory into *wyk* format.
II. Determining sliding scores between a reference *wyk* file and the other *wyk* files and then shifting the other *wyk* files to align with the reference.
III. Summation of the reference *wyk* file with the other aligned *wyk* files in the data set.
IV. Encoding or converting of all ultimate *wyk* sum of the data set into a nucleotide probability matrix (also known as a position weight matrix).
V. Uploading the matrix to a DNA logo generator to obtain a final visualization. This step is not necessarily required but can be instrumental in understanding the success of the motif recognition.

Data Sets

Input data can be any data set of *.fasta* files. It is best recommended, however, that there is a motif or regions to explore that are conserved in the data set as this pipeline will work to align any regions of similarities but if they are not uniquely conserved in the data set, the final output will not be of significant interest.

An example, which is the example in which this pipeline has been utilized, is a collection of *E.coli* genes that have binding regions for the cyclic AMP receptor protein (CRP), an important transcription factor. The specific sequence information and gene names for this collection can be found via navigation in this pipeline's github directory.

Pipeline

I. encoder.wyk

A. Usage
   a. ./encoderWYK.sh /home/a_kesanapally/pipeline-project-evokes/data/ecoli output/ecoli_wyk
B. Behavior
   a. The script encodes all the *.fasta* files in a data folder into *wyk* format. Each file will be placed in a specified directory.
   b. The format for the command and its arguments are:
      i. ./encoder WYK.sh (input directory of *.fasta* files) (the output directory where the *wyk* files will be stored)

c. The script calls a *.py* file, encodeWYK.py, that does the encoding of each individual *.fasta* file. The usage for this *.py* file is as follows:
    i. python3 encodeWYK.py --infile (infilename) --outfile (outfile name)

C. Corresponding Files
    a. A corresponding manualencoderWYK.py, not used in the pipeline, that has the same contents but is meant for troubleshooting and visualization of the data has similar usage as encodeWYK.py:
        i. python3 manualencoderWYK.py --infile (infilename) --outfile (outfile name)

II. shifter.sh

A. Usage
    a. ./shifter.sh /home/a_kesanapally/pipeline-project-evokes/output/ecoli_wyk output/ecoli_shift_wyk /home/a_kesanapally/pipeline-project-evokes/output/ecoli_wyk/wyk_ecoblgr1.txt

B. Behavior
    a. The script will take in a directory of *wyk* files and with a user-input of a reference *wyk* file, will shift the remaining *wyk* files in the directory to that reference.
    b. The format for the command and its arguments are as follows:
        i. ./shifter.sh (*wyk* input directory) (output directory) (reference file path)
    c. This script utilizes one *.py* file, corrshift.py, that determines the shift value from the peak of all the sliding alignment score values and aligns a non-reference *wyk* file to the reference *wyk* file. The usage for this is as follows:
        i. python3 corrshift.py --ref (path of reference *wyk* file) --infile (input file path of a non-reference *wyk* file) --outfile (the path/name of the outfile)

C. Corresponding Files
    a. There are two *.py* files that can be used for further analysis, troubleshooting, and visualization but are not used in the pipeline. manualscorer.py is a file that determines all the possible sliding alignment scores to be assessed visually or in the python environment and manualshifter.py is a file that allows a sequence to be shifted given input values. The usage for both is as follows:
        i. python3 manualscorer.py --ref (path of reference *wyk* file) --infile (input file path of a non-reference *wyk* file)
        ii. python3 manualshifter.py --infile (path of a *wyk* file) --shiftright (a boolean of whether or not the shift is to the right (if the peak shift value from manualscorer.py is positive) or the the left (if the value is negative)) --number (the number to shift by or the absolute value of the peak shift value from manualscorer.py) --outfile (output file path of the shifted *wyk* file)

III. summer.sh

A. Usage

a. ./summer.sh /home/a_kesanapally/pipeline-project-evokes/output/ecoli_shift_wyk output/ecoli_sums /home/a_kesanapally/pipeline-project-evokes/output/ecoli_wyk/wyk_ecoblgr1.txt /home/a_kesanapally/pipeline-project-evokes/output/ecoli_shift_wyk/shifted_wyk_ecocya.txt

B. Behavior
   a. This script will take in a directory of shifted *wyk* files, the name of an output directory, the path of the reference *wyk* file, and the path of one of the shifted *wyk* files in the directory to initialize the script (such that the first output, labeled sum1.txt, is summed from two known files). It outputs an iterative amount of sum files labeled sum1.txt to sumN.txt, where N is the number of data files in the shifted directory and N+1 represents the total amount of files used to determine that sumN.txt file.
   b. The command arguments are as follows:
      i. ./summer.sh (input directory path) (output directory path) (reference *wyk* file path) (path of one shifted *wyk* file that is in the input directory)
   c. This script calls one *.py* file labeled sumcreator.py that takes in two *wyk* files and determines the summed *wyk* files of both. The usage and arguments for this python file is as follows:
      i. python3 sumcreator.py --infile1 (path of the first *wyk* file) --infile2 (path of a second *wyk* file) --outfile (the path/name of the outfile which will be the sum of both *wyk* files)

C. Corresponding Files
   a. A corresponding *.py* file, once again not used in the pipeline but used for troubleshooting and visualization, is titled manualsummer.py. It's arguments and usage are similar to sumcreator.py and are as follows:
      i. python3 manualsummer.py –infile1 (path of the first *wyk* file) –infile2 (path of the second *wyk* file) –outfile (the path/name of the outfile which will be the sum of both *wyk* files)

IV. encoderACGT.sh

A. Usage
   a. ./encoderACGT.sh /home/a_kesanapally/pipeline-project-evokes/output/ecoli_sums/sum6.txt output/ecoli_acgt 7

B. Behavior
   a. This script takes in a file that is supposed to represent the final, ultimate summation of a reference *wyk* file and its aligned *wyk* files (the output of summer.sh, for example), the directory for the output file, and the number of total files used to create that file (N+1 from summer.sh, for example). The script results

in a *.csv* file that represents a nucleotide probability matrix or a position weight matrix that gives a probability of each of the four nucleotides at each index.

b. The command and argument structure is as follows:
    i. ./encoderACGT.sh (path of the input sum *wyk* file) (output directory) (number of files used to determine the sum)

c. The script calls a *.py* file named encodeACGT.py that does the sum *wyk* to probability matrix conversion within its code through the following arguments and usage:
    i. python3 encodeACGT.py –infile (path of the input sum *wyk* file) –number (number of files used to determine the sum) –outfile (output file path)

C. Corresponding Files

a. A corresponding file, manualencoderACGT.py, is not used in the pipeline but used to test, troubleshoot, and visualize this step in the process has similar usage as encodeACGT.py:
    i. python3 manualencoderACGT.py –infile (path of the input sum *wyk* file) –number (number of files used to determine the sum) –outfile (output file path)

V. DNA Logo Generator

A. Usage

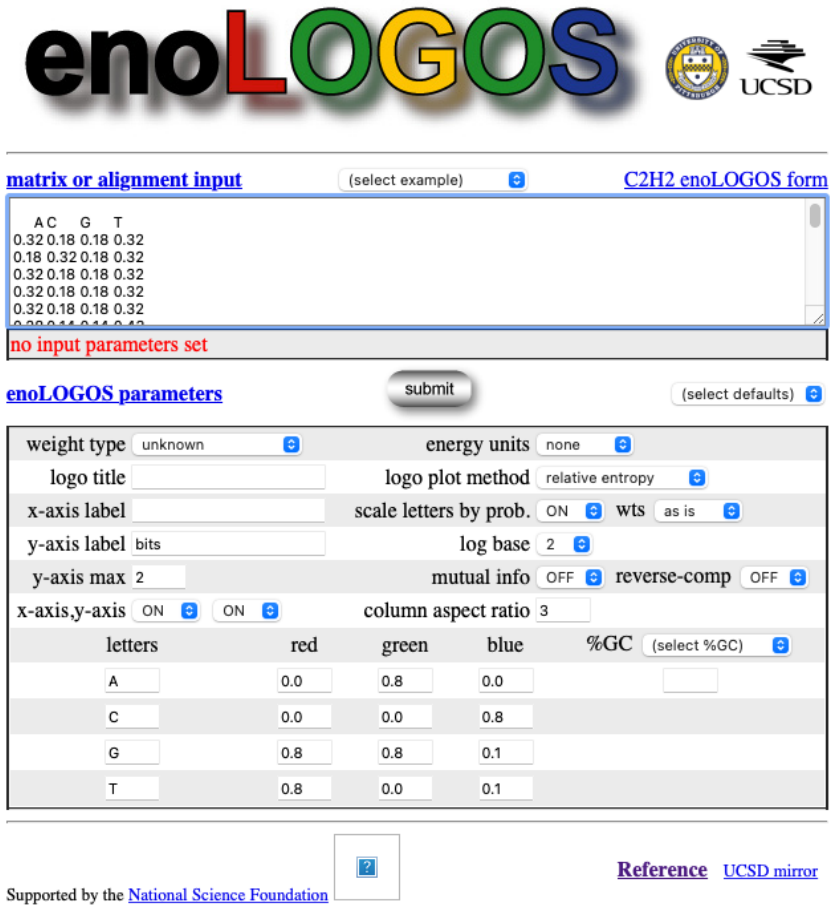
a. This step is not a script but rather usage of a DNA logo generator to visualize the probability matrix into something more accessible. The recommendation, and example of this step, for this pipeline is used via [Panayiotis (Takis) Benos Lab's DNA Energy Normalized Logo Generator (enoLOGOS)](#).

b. Select and copy all the contents of encoderACGT.sh's output (a nucleotide probability matrix):
    i.

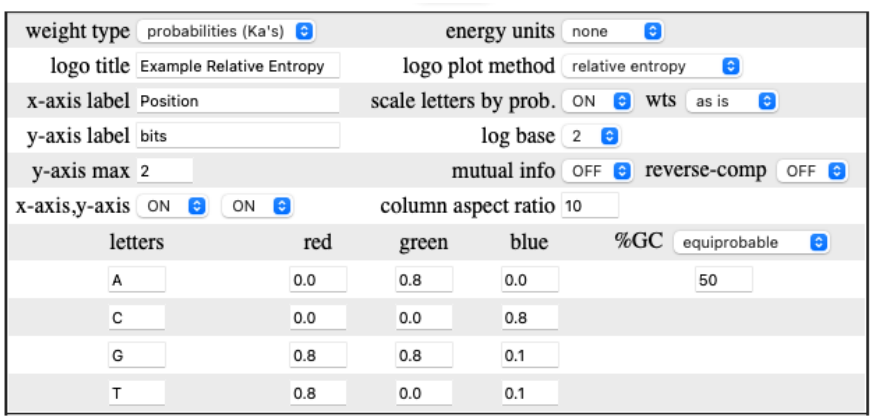
example_matrix

	A	C	G	T
1	A	C	G	T
2	0.32	0.18	0.18	0.32
3	0.18	0.32	0.18	0.32
4	0.32	0.18	0.18	0.32
5	0.32	0.18	0.18	0.32
6	0.32	0.18	0.18	0.32
7	0.28	0.14	0.14	0.43
8	0.14	0.28	0.14	0.43
9	0.28	0.28	0.14	0.28
10	0.14	0.43	0.14	0.28
11	0.28	0.14	0.14	0.43
12	0.28	0.14	0.14	0.43
13	0.28	0.14	0.14	0.43
14	0.57	0.14	0.14	0.14
15	0.57	0.14	0.14	0.14
16	0.14	0.57	0.14	0.14
17	0.28	0.14	0.14	0.43
18	0.14	0.14	0.14	0.57
19	0.28	0.14	0.28	0.28
20	0.43	0.14	0.14	0.28
21	0.28	0.14	0.14	0.43
22	0.28	0.14	0.28	0.28
23	0.57	0.14	0.14	0.14
24	0.14	0.14	0.14	0.57
25	0.28	0.14	0.14	0.43
26	0.14	0.28	0.28	0.28
27	0.14	0.14	0.28	0.43
28	0.25	0.11	0.25	0.39
29	0.54	0.25	0.11	0.11
30	0.22	0.22	0.36	0.22
31	0.22	0.22	0.36	0.22
32	0.07	0.07	0.22	0.64
33	0.22	0.07	0.64	0.07

c. Paste the contents into the generator's input text box:

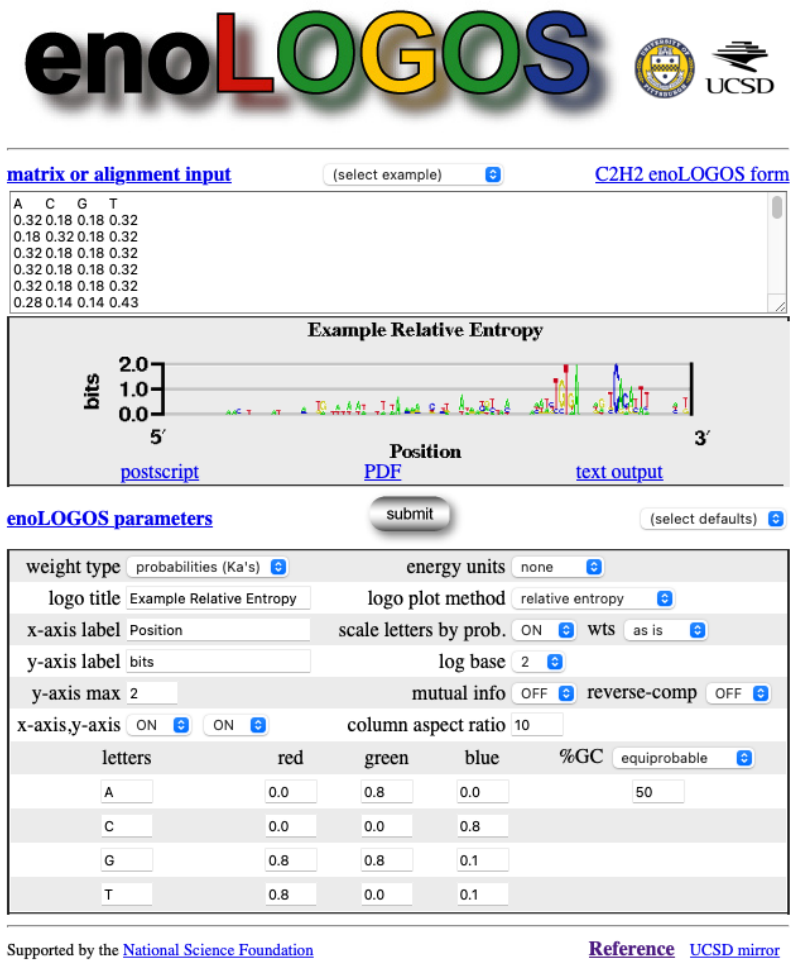
i. 

d. Select for the following parameters for a relative entropy logo:

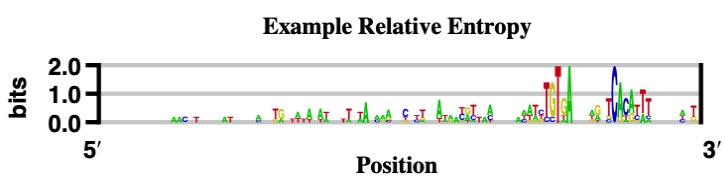
i. 

ii. If you would like to create a frequency logo, change the logo plot method to “frequency” from “relative entropy”. It is recommended to manipulate sections such as logo title, x-axis label, y-axis label, y-axis max, x-axis, y-axis, and column aspect ratio but leave the parameters as shown above.

e. Press submit for the settings to run and a preliminary output like this will appear:

i. 

f. Now you can assess the output via postscript, PDF, or text output. To capture the logo, it is best recommended to select PDF and save the PDF or screenshot the logo for a final output such as this:

i. 

Authorship

This pipeline was created by Anirudh Kesanapally, simultaneously for the Gary D. Stormo, Ph.D Lab at the Washington University in St. Louis School of Medicine and for Michael Landis', Ph.D Fall 2022 course *Practical Bioinformatics* at Washington University in St. Louis College of Arts & Sciences.