A Co-Evolution Simulator (aCES) 2.0 *BETA VERSION*, User Manual

From "aCES: A Co-Evolution Simulator generates co-varying proteins and nucleic acids"

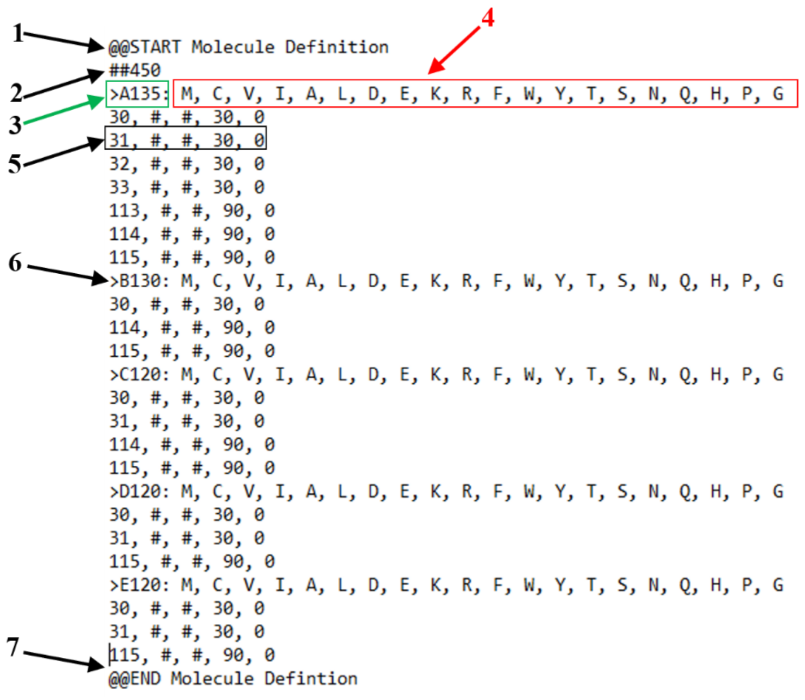
Authored by Devin Camenares
Assistant Professor, Department of Biochemistry
Alma College
614 W. Superior St, Alma MI 48801
camenaresd@alma.edu

Purpose:

This tool will take necessary parameters and generate up to five FASTA files that contain a simulated multiple sequence alignment. This alignment can be generated with a user-determined number of sequences, sequence and residue type, and the conservation of similarity and identity can (and must) be specified by the user. Most importantly, co-evolution constraints of both intra- and inter-molecule fashion can be applied to the generation of sequences.

Input:

This tool requires a text file with the parameters or constraints for the simulation. The input file provided, both the blank and the example, are annotated to help guide the user. **These annotations are provided before or after a block of code that is bracketed by "@@START .. " and "… @@END". Any text outside those brackets is ignored by the program; any text inside must conform to the expected input format.** For clarity, consider the beginning of the following example input file, which defines the nature of each molecule:



```
@@START Molecule Definition
##450
>A135: M, C, V, I, A, L, D, E, K, R, F, W, Y, T, S, N, Q, H, P, G
30, #, #, 30, 0
31, #, #, 30, 0
32, #, #, 30, 0
33, #, #, 30, 0
113, #, #, 90, 0
114, #, #, 90, 0
115, #, #, 90, 0
>B130: M, C, V, I, A, L, D, E, K, R, F, W, Y, T, S, N, Q, H, P, G
30, #, #, 30, 0
114, #, #, 90, 0
115, #, #, 90, 0
>C120: M, C, V, I, A, L, D, E, K, R, F, W, Y, T, S, N, Q, H, P, G
30, #, #, 30, 0
31, #, #, 30, 0
114, #, #, 90, 0
115, #, #, 90, 0
>D120: M, C, V, I, A, L, D, E, K, R, F, W, Y, T, S, N, Q, H, P, G
30, #, #, 30, 0
31, #, #, 30, 0
115, #, #, 90, 0
>E120: M, C, V, I, A, L, D, E, K, R, F, W, Y, T, S, N, Q, H, P, G
30, #, #, 30, 0
31, #, #, 30, 0
115, #, #, 90, 0
@@END Molecule Defintion
```

1. This line defines the start of the block defining all molecule characteristics. Anything before this line is ignored by the program.

2. After the ## the number of sequences for each molecule is specified. In this example, the program will 'evolve' 450 sequences each for molecules A, B, C, D, and E.

3. This begins the block defining molecule A – the number indicates the total length of the sequence/molecule.

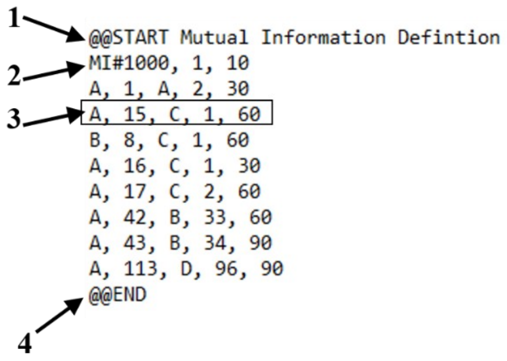
4. Following the colon in the molecule definition line is the grouping of residues that can be chosen from when creating the sequence. In this example, the simulation will create a protein, and treat each of the twenty amino acids (initially) as equal choices. For DNA or RNA, the input on this line should read as **"A, G, T, C"** or "**A, G, U, C"** respectively, in any order. It is also possible to group the amino acids or any other characters together. For example, the line "**MC, VIAL, DE, KR, FWY, TSNQ, H, P, G"** will create a two-dimensional array of choices. The first dimension of choice, in which there are 9 possibilities, relates to the chemical nature of the amino acid. The second dimension of choice governs the selection of the exact amino acid identity. For the purposes of simulating co-evolution, the program will only consider the first dimension – in this respect, D (Asp) and E (Glu) are treated as identical with that particular grouping.

5. This line indicates, separated by commas 1) the number of a given residue 2) the default identity choice, in the first dimension 3) the default identity choice in the second dimension 4) the percentage conservation, 0-100 in the first dimension, and 5) the percentage conservation, 0-100, in the second dimension. If a # symbol is present, the program will select a value at random from the available space.

6. The beginning of a block that defines molecule B – this follows the same format as was outlined for molecule A in items 3-5 above. This same pattern also applies for molecules C, D, and E.

7. This line defines the end of the entire block for defining each molecule. Any code after this line will be ignored by the program.

This is followed by a second block that defines the co-evolution constraints within and between the molecules simulated. Again, for the example input this appears as follows:



```
@@START Mutual Information Defintion
MI#1000, 1, 10
A, 1, A, 2, 30
A, 15, C, 1, 60
B, 8, C, 1, 60
A, 16, C, 1, 30
A, 17, C, 2, 60
A, 42, B, 33, 60
A, 43, B, 34, 90
A, 113, D, 96, 90
@@END
```

1. This line defines the start of the block for defining all co-evolution. Anything before this line is ignored by the program.

2. This line defines the overall parameters for how co-evolution will be simulated. The numbers after the MI# indicate the following, separated by commas: 1) The maximum number of iterations of adjustments to the distribution of a pair of residue identities, 2) the % threshold for a MI score to deviate from the target and be selected, and 3) The multiplier that is used to the random adjustment variable in each iteration. A larger number creates more volatile swings possible in each iteration (i.e. the multiplier for how many standard deviations from the mean value of a normal distribution will be sampled).

3. This line defines each specific co-evolution constraint. It indicates the following, separated by commas: 1) the first molecule in the co-evolving pair and 2) the residue in this molecule, 3) the second molecule in the pair and 4) the co-evolving residue in this molecule, and 5) the strength of the constraint, as a percentage of possible co-evolution from 0-100. Note that this is modulated by the overall conservation of the residues in question – a constraint of 100% co-evolution will actually produce very little co-evolution if each residue is already invariant (100% conservation).

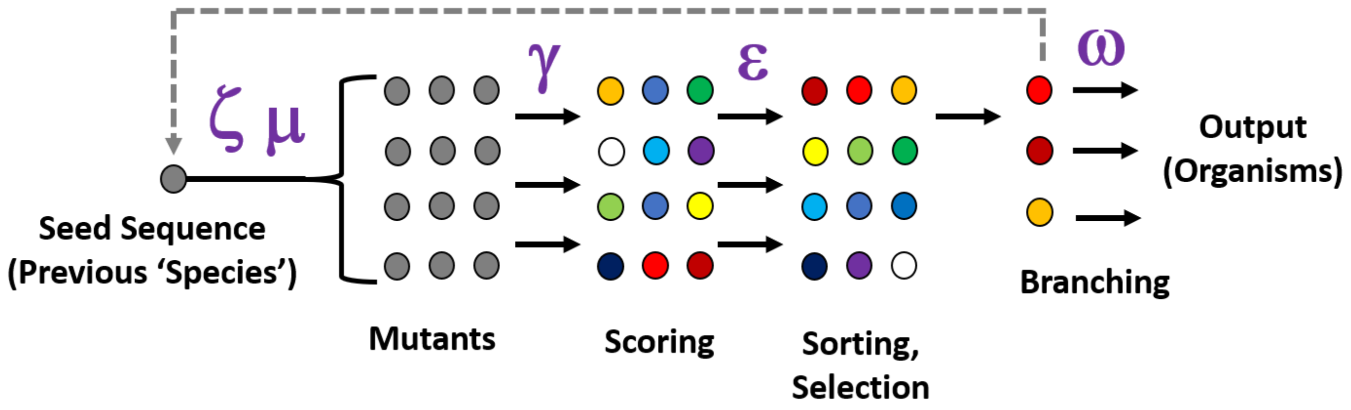
4. This line defines the end of the entire block for defining co-evolution. Any code after this line will be ignored by the program.

** NOTICE: The program has not been exhaustively tested with incorrect input formats. It is likely to freeze or abort if incorrect inputs are provided. You are encouraged to use the debugging option to help identify the problem – please email your input, debugging logs and results of failed runs to camenaresd@alma.edu for additional assistance.

Once the input file contains your desired constraints in the correct format, you can proceed to load the program. The interface for this program is described in the following section.

Generation of Sequences

Compared to the first iteration of this program, aCES2.0 creates sequences for each 'organism' by first generating a pool of mutants, scoring each mutant in accordance with the constraints and distributions generated, followed by sorting and selection. This process is carried out for each sequence (for all 5 molecules for each organism) and then repeated for each organism, and is depicted in the following figure:



Starting with a seed sequence, or a previous 'species', a number of mutant sequences will be generated. This number is specified by **Zeta**, and is by default 1000. For each mutant created in this pool, the chance that a particular residue is mutated is specified by **Mu**, by default 1. This is a constant mutation rate, in accordance with the Jukes Cantor model for evolution (future versions of this program may allow for implementation of different models). Each mutant is then scored in how well each of its residues satisfies both the conservation constraints and the co-evolution constraints; the degree to which the latter is favored over the former is the **Gamma** constant (default is 1, which is no special weighting). By increasing this constant, selection of a sequences which display co-evolution can be prioritized. These mutant sequences are then sorted by score, and a selection is made from a subset of this pool – a top fraction that is specified by **Epsilon.** For example, the default value of 90 means that only the top 10% of mutants with respect to score may proceed. From this subset, one or more are chosen at random to be included as 'organsims' in the final set. The number chosen is specified by **Omega**, which defaults to 1. All of those selected are written as final results – only one will be used as a branch to create a new sequence, repeating the cycle for the number of organsims specified in the input file. s

Interface:

- **Job ID:** This textfield, with a timestamp generated number, will be used to append each file name with a unique ID. This may also be used to name the directory (see below). Note that a new ID is automatically generated each time the program is loaded anew.

- **Generate New ID:** As the name implies, this button will generate a new ID (so that previous work is not overwritten). The new Job ID is displayed in the textfield (item #3)

- **Output Folder Name/Location:** This selection option will allow you to use either the unique Job ID as the output folder directory (which is created anew if it does not exist) or a common folder simply named 'output'. The latter option may be useful if you want all the results from multiple runs to be in the same folder; the files themselves will still be appended with the unique ID, to allow them to be distinguished from another.

- **Debugging Mode:** Checking this box will lead the program to output a file that details the steps being taken – this will report information in several stages:

a. **New Molecule Created:** All the inputs will be read and values that are captured are reported line by line. This includes parameters that may not have been specified, in which case a random number will be chosen and displayed. [Category #1 is the first dimension of choice, Category #2 is the 2nd dimension. Thus, if you are using grouped amino acids, first dimension is the chemical nature of the amino acid and the 2nd dimension is the actual identity. The co-evolution algorithm ignores the 2nd dimension.]
b. **Fill in unspecified positions:** similar to the preceding lines, but for any positions that were not explicitly defined.
c. **MI pair:** for any specified co-evolution constraint, the target level of MI, as well as the final MI score and the number of iterations used to reach this score is displayed.
d. **MI Needed!** Every time the program encounters a residue that participates in a direct co-evolution interaction, it will call upon the already established distribution to determine the probability of a particular residue being selected. It reports on the first line of this block the residues in question and the residue choice for the partner residue. The next three lines report the counts of residues it has available to choose from, with the total listed after the asterisk. The PbS-T line reports a probability cutoff array – the program then selects a random number from 0-100 and compares this against the array to select the residue, which is then displayed on the following line.

- **Enter Constants (Zeta, Omega, Epsilon, Mu):** Here, users can specify constants that are unique to aCES2.0, and described in more detail in the preceding section. If an out-of-bounds value is selected (for example, 110 for Epsilon, which ranges from 0-100), then the maximum or minimum value will be used. If a number is not entered (for example, if the default text is left there), then hard-coded default values will be used.

- **Open MSA Parameters File:** Selecting this button will allow you to choose the parameters file for generating the simulated MSA. After selecting the file, the parameters will be immediately processed and the result delivered.

Beta Version Note

This version will produce sequences that follow the conservation constraints faithfully, and creates a bifurcating phylogeny. However, these tend to obscure the co-evolution signals, and generation of these signals in the resulting data is so far not robust. I invite the reader, if you have programming experience, to try and solve this problem! I'd be happy to collaborate to do so.