CES–Codon Evolution Simulation

Version 1.2

Preston Hewgley, Micheal A. Gilchrist

November 22, 2013

1 Overview

CES (Codon Evolution Simulation) is a program that takes genetic sequences observed in the wild, generates random synonymous[1] sequences, and simulates the evolution of the random sequence under selective pressures against nonsense errors (NSE). CES simulates evolution by calculating the probability of mutant fixation[2] for all one-step neighbors[3] and choosing a substitution based on these probabilities and a randomly generated number. By default, evolution steps are carried out until an average of 5 substitutions per codon are carried out. The basis of the calculations used in this code are described in Sella and Hirsh (2005) and Gilchrist, Shah, and Zaretzki (2009). The code was used in Gilchrist, Shah, and Zaretzki (2009) to generate simulated sequences and compare them to observed sequences using several metrics.

2 Repository Information

Description	Location	Notes
Base Repository	file:///home/semppr/svnrepos/CES	The base repository for this project was made by Preston Hewgley in June 2012.
Source Files	file:///home/semppr/svnrepos/CES/source	This folder holds current and past versions.
Data	file:///home/semppr/svnrepos/CES/data	This folder holds FASTA files, elongation rates, expression levels, and mutation files.
Results	file:///home/semppr/svnrepos/CES/results	This folder holds data from results and graphics produces with the data.
Benchmarks	file:///home/semppr/svnrepos/CES/benchmark	This folder holds benchmark data produced using the -ben option.

3 Commandline Arguments and Options

3.1 Commandline Arguments

Here is a sample commandline template:

./CES -F <*FASTA input file*> -T <*tRNA/elongation rate file*>[*Options*]

[1] A synonym to a codon is a different codon that codes for the same amino acid. Since there are 61 translated codons and only 20 translated amino acids, there is redundancy among codon-amino acid pairs. Thus, a synonymous sequence is a sequence that may have a different codon sequence, but has the same amino acid sequence.

[2] The probability of a mutant fixation is the probability that a mutation will occur and the new genotype will take over the entire population.

[3] One-step neighbors are synonymous sequences that differ by only one codon.

3.2 Options

Flag	Argument	Notes
-F	<FILE>	Location of the fasta file to be used for simulation. Note the program will likely crash if genes with internal stop codons are used. phi values must be included on the same line as the ORF name OR included in a phi value file, e.g. >YAL001C phi= 0.0088862 Note the use of TABS between the ORF name and "phi=" and the phi value
-T	<STRING>	Specify the location of the tRNA abundance/codon translation file.
-P	<STRING>	Specify the location of the phi value file.
-C	<NONE>	Outputs codon counts for low expression genes. Does not produce output files for evolution simulation.
-U1	<STRING>	Specify the location of incoming sum mutation rate file (for more info see section **??**).
-U2	<STRING>	Specify the location of individual mutation rate file (for more info see section **??**).
-A1	<REAL>	Specify ribosome initation cost in ATPs [DEFAULT] a1=4
-A2	<REAL>	Specify ribosome elongation cost in ATPs, a2 in SEMPPR [DEFAULT] a2=4
-AT	<REAL>	Specify genome AT bias (0.0-1.0). [DEFAULT] at=0.5
-B	<REAL>	Specify the B parameter value [DEFAULT] B=0.0025
-D	<BOOLEAN>	Indicate whether to print out delta_eta files for each evolutionary step. Print outs can get quite large [DEFAULT] D=0.
-E	<BOOLEAN>	Indicate whether to print out eta and mu trace file [DEFAULT] E = 0.
-I	<INT>	Specify whether CES should relax selection on the last <INT> amino acids of each sequence. Used because it is thought that most genes can lose a few aa at the end, but still be functional.
-M	<INT>	Indicates simulation time. Simulation time is scaled by the mutation rate mu, where mu = 1E-9. If the argument X is > 0, then we expect to see X * mu substitutions/codon. If X < 0, then we run for -X/mu time or, in other words, we expect to see -X substitutions if the genes were evolving neutrally. [DEFAULT] = -20
-Ne	<REAL>	Effective population size. [DEFAULT] = 1.36E7
-O	<STRING>	Specifies the folder for output files as well as prefix for specific output and log files. [DEFAULT] "output/out"
-V	<REAL>	Ratio of transition to transversion mutation rates. [DEFAULT] 1.0
-ben	<NONE>	Replaces randomly generated numbers with predetermined ones. Useful when creating benchmarks or testing.

4 Input File Information

All acquired Input Files can be found in the data directory of CES.

4.1 FASTA file

- File may start with header
- Each gene must start with ATG
- Must either (a) include phi values in header or (b) include phi value file
- Must be in proper FASTA file format, e.g.
  >(Gene ID) phi= (Value)
  ATGNNNNNNNNNNN...

4.2 Phi value file

- File may start with header, but the header must start with a double quote (")
- Gene ID and Phi value must be separated with "\t", ",", or " " (tab, comma, or space).
- E.g.
  "Optional Header"
  YAL001C,0.008886154568855773

4.3 tRNA/Elongation file

- File may start with header, but the header must start with a double quote(").
- Must separate AA index, Codon, and Elongation rate with "\t" (tab).
- Must group synonymous codons together.
- E.g.
  "Optional Header"
  A GCA 7.82

4.4 Mutation file

There are two types of mutation files. The first type consists of the sum of incoming mutation rates for each codon. The second includes all individual mutation rates for all codons.

4.4.1 Incoming Sum Mutation file

- File may start with header, but the header must start with a double quote (").
- Must separate AA index, Codon, Mutation rate with "\t" (tab).
- Must group synonymous codons together
- E.g.
  "Optional Header"
  A GCA 1
  A GCC 0.5872495

4.4.2 Individual Rate Mutation file

- File may start with header, but the header must start with a double quote(").
- Must conform to this format:
  "Optional Header"

>A	GCA	GCC	GCG	GCT
GCA	0	.22	.11	.32
GCC	.34	0	.13	.30
GCG	.30	.25	0	.29
GCT	.32	.27	.09	0
>C	TGC	TCT		
TGC	0	2.09		
TGT	1	0		

  ETC...
- A ">" must preeced each Amino Acid.
- Entries may be separated with "\t" or " " (tab or space).
- Index convention is as follows: Entry in row i and column j is mutation rate from codon i to codon j (μ_{ij}).
- Mutation rate of any codon to itself is 0.
- Rows and columns must be labeled symmetrically. Check if diagonal is composed of zero's.

5 Pseudocode

5.1 Quick and Dirty

1. Read Input Files (Fasta sequence file, elogation rate file, and (optional) mutation rate file)
2. Generate random synonymous sequence
3. Calculate η_{obs}
4. Calculate $\Delta\eta$ for all one-step neighbors
5. Calculate Replacement Probability for all one-step neighbors
6. Choose random number to determine substitution based on Replacement Probability
7. Repeat steps 3-6 for desired evolution time
8. Print output

6 Overview of Equations

$$\sigma_i(\vec{c}) = \prod_{j=1}^{i} \frac{c_j}{(c_j+b)} \qquad \boxed{\text{Translation Probability}}$$

Where: c_j=Elongation rate of codon j
b=Background nonsense error rate

This is the probability that a ribosome will translate up to and including codon i. Since it can be shown that the probability of translating a single codon j equals $\frac{c_j}{c_j+b}$, it follows that the cumulative translation probability of codon i equals the product of the elongation probability for every codon up to and including codon i. **NOTE:** σ_n denotes the probability that a protein is fully translated.

$$\xi(\vec{c}) = \frac{1}{1-\sigma_n(\vec{c})} \sum_{i=1}^{n} (a_1 + a_2 i)\sigma_{i-1}(\vec{c})\frac{b}{c_i+b}$$

Expected energetic cost of a NSE

Where: a_1=Energetic cost of ribosome recharge
a_2=Energetic cost of a peptide bond

Note that the first part of the summation, $(a_1 + a_2 i)$, is the cost of translating up to codon i and the second part, $\sigma_{i-1}(\vec{c})\frac{b}{c_i+b}$, is the probability that a nonsense error will occur on codon i. $\frac{1}{1-\sigma_n(\vec{c})}$ is a factor to scale this calculation.

$$\eta(\vec{c}) = \left(\frac{1}{\sigma_n} - 1\right)\xi(\vec{c}) + (a_1 + a_2 i)$$

Expected cost of producing one protein

Essentially, this is a composition of the expected cost of all nonsense errors plus the cost of a completed protein, $(a_1 + a_2 i)$.

$$w(\vec{c}) \propto e^{-q\phi\eta}$$

Fitness function

Where: q=Scaling factor
ϕ=Protein production rate

We assume that fitness is proportional to a negative exponential function of the protein cost rate.

$$\pi(i \to j) = \frac{1-\frac{w_i}{w_j}}{1-\left(\frac{w_i}{w_j}\right)^{2Ne}}$$

Fixation probability

Where: Ne=Effective population size

This is the probability of fixation given a mutation occurs.

$$P_j = \mu_{ij} Ne \pi(i \to j)$$

Replacement probability

Where: μ_{ij}=Mutation rate from codon i to codon j

This is the probability that a mutation occurs and it fixes. Basically, it is the probability of fixation given a mutation occurs, $\pi(i \to j)$, weighted by the rate at which mutations occur, $\mu_{ij} Ne$.

7 Other Info

7.1 Compilers

The code can be recompiled from source and optimized for the hardware of the machine it will be run on. The source code has been successfully compiled with the following compilers.
Mac: gcc, g++
Linux: g++, gcc -lm
In addition, the development libraries of the GNU Scientific Libraries must be installed.

7.2 Build

Builds on Ubuntu 12.04 machine with GSL libraries using provided makefile.
Makefile provides several different sets of options. For use with multiple CPUs that share memory, choose a set of options that includes the flag -fopenmp when compiling

7.3 Bugs

In case of any bugs or trouble with the code, send an email to whewgley@utk.edu or mikeg@utk.edu