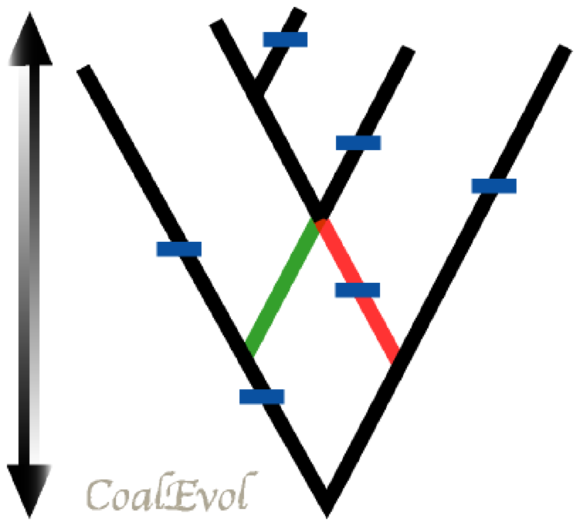
Documentation for **CoalEvol**

Coalescent simulation of nucleotide, codon and amino acid data under complex recombination, demographics, multispecies histories and migration

Current version is 7.4.0

© 2006-2024 Miguel Arenas (marenas@uvigo.es) and David Posada (dposada@uvigo.es).

December 8, 2024



Contents

Disclaimer

Credits

This program was developed at the Department of Biochemistry, Genetics and Immunology of the University of Vigo (Spain) and the Centre for Molecular Biology "Severo Ochoa", Consejo Superior de Investigaciones Científicas (CSIC), Madrid (Spain).

1. Purpose

CoalEvol generates samples of nucleotide, codon and amino acid sequences, collected at same or different times, from populations evolved under complex recombination (including recombination hotspots and intracodon recombination), species/population trees, migration and demographic changes (that can be different for each population) by using the coalescent approach [1-3]. Phylogenetic tree/s can also be introduced as an alternative to the coalescent simulation. Then, a large set of substitution models (at nucleotide, codon and amino acid levels) allows the simulation of molecular evolution. Currently, this is the only one simulator that jointly implements all these capabilities. The program is freely available from http://code.google.com/p/coalevol/.

What is exactly implemented in CoalEvol?

CoalEvol can simulate coalescent histories with:

- **Recombination** (including hotspots [4] and intracodon recombination [5]).
- **Populations structure** and several **migration** models (island [6], stepping-stone [7] and continent-island [8]). Indeed, migration rates may vary with time.
- **Population/species tree** can be introduced by the user [see, 9]. Here, each population can be evolved under a particular population size that can be constant or variable with time [e.g., 9].
- **Demographics**. Population size may change according to a growth rate or by temporal periods user-defined.
- Samples can be simulated at same time or at different times (temporal **longitudinal sampling or tip dates**) [see, 10].

Alternatively to the coalescent simulation, **user-specified trees** can be directly introduced in *CoalEvol*.

Then, *CoalEvol* can evolve **nucleotide**, **codon** and **amino acid** data along the simulated coalescent tree or the user-introduced phylogenetic tree:

- Nucleotide data. **All nucleotide substitution models** are implemented, such as JC [11], HKY [12] and GTR (or REV) [13]. Moreover, the non-reversible GTR (GTnR) is also implemented.
- Codon data. *CoalEvol* implements a large variety of codon models:
    - **GY94** [14] × heterogeneous synonymous/nonsynonymous ratio rate (dN/dS or ω) **M1-M10** models [15, 16] × any nucleotide model. Moreover, **dN/dS** may also **change per branch** [e.g., 17]. **Physicochemical contributions** of the derived amino acids can be also considered [see, 18].
    - **MG94** [19] × any nucleotide model [see, 20].
    - **HB codon models** [21].
    - **Empirical codon models** [22, 23].
- Amino acid data. *CoalEvol* evolves amino acid data under multiple substitution models, namely ***Blosum62*** [24], ***CpRev*** [25], ***Dayhoff*** [26], ***DayhoffDCMUT*** [27], ***HIVb*** [28], ***HIVw*** [28], ***JTT*** [29], ***JonesDCMUT*** [27], ***LG*** [30], ***Mtart*** [31], ***Mtmam*** [32], ***Mtrev24*** [33], ***RtRev*** [34], ***VT*** [35], ***WAG*** [36] and **any user-defined model**.

In addition, for all these models:

- Nucleotide, codon or amino acid frequencies may vary across sites by user-specified categories (**CAT models**) [37].
- **Haploid and diploid** data can be simulated.
- **Heterogeneity (+G)** can be applied across sites (see *SGWE [http://code.google.com/p/sgwe-project/]* for variable +G across genomic regions).
- **Proportion of invariable sites (+I)**.

CoalEvol can ouput:

- The simulated **coalescent trees** (newick format), including the tree for each recombinant fragment.
- The simulated **ancestral recombination graph** (ARG) [38] (branches format [39]).
- **Recombination breakpoints**.
- Simulated **alignments** can be printed in ***phylip***, ***fasta*** and ***nexus*** formats.
- Codon models with variable dN/dS across sites and/or branches allow **print the simulated dN/dS per site and/or branch**.
- **Ancestral sequences** (GMRCA and sumMRCA [see, 5, 40]).
- **Other information.** Statistics for substitution events (synonymous and nonsynonymous for codon models), recombination events (including intracodon recombination for codon models) and migration events. Times to the MRCA and GMRCA. Comparisons with the expected values.

Overall, *CoalEvol* allows the simulation of molecular data and multiple complex evolutionary histories and substitution models (see the folder “example_input_files” with a variety of examples simulated by *CoalEvol*). It can be also combined with SGWE to simulate genomic and proteomic sequences.

2. Executables and compilation

Executable files are provided for Linux Debian and MacOS X (Intel and G4 processors), and one Makefile is provided for compilation in any OS with a C compiler. This makefile can be optimized for different users, for example using the optimization option –fast instead of –O3, for G5 Mac processors. To compile the program type (it may take some minutes): *make all*

A second makefile "Makefile_MPI" is provided to compile a MPI version (which could be convenient for diverse simulation experiments [e.g., 41, 42]). This makefile might need some modifications for particular OS. To compile the program type: *make -f Makefile_MPI*

MPI libraries have to be installed in the host for running *CoalEvol* in parallel. The minimum number of processors is two. An example of execution for 3 processors is the next: *mpirun -np 3 CoalEvol7.3.5*

3. CoalEvol Usage

The input of the program consists of a series of arguments and parameter values (Table 1) that can be entered in the command line or, more conveniently, specified in a text file called "parameters" that should be located in the same directory as the executable. These arguments include the parameter values used in the simulations and several printing options that control the amount of information that is sent to the console or output files.

Additionally, some scenarios also require additional input files (see section 3.4):

- User-specified tree/s.
- Settings for recombination hotspots.
- A user-specified sequence for the MRCA/GMRCA.
- A user- specified empirical codon substitution model.
- A user- specified empirical amino acid substitution model.

3.1. Command line

In Mac systems, the command line is provided by the Terminal, while in Windows, this is provided by the windows console or command prompt. If the user specifies any argument in the command line, *CoalEvol* will apply such arguments with default values not included in the command line.

It is recommended check the information shown in the screen to be sure that simulations are parameterized as intended. Some example command lines are the following:

./CoalEvol7.3.5 -n2 -s8 150 -e200 1 -@JTT

*./CoalEvol7.3.5 -n2 -s8 150 -e1000 2 -=4 1995 1 1 2003 4 6 1997 2 3 2001 7 8 -/1200 -g1 3 1000 1250 1000 1300 1550 2000 1560 1000 3000 -q1 4 2 2 3 1 -t3 100 800 0.002 0.001 0.003 -%1 1 2 10000 -r2.3e-6 -hUserHetRec -o0.1 -u4.1e-5 -f20 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06 -@UserEAM -a1 0.7 -i0.52 -xseqGMRCA -bsequences -c1 1 0 -jtrees -ktimes -dbreakpoints -*NetworkFile -y2 -#245*

./CoalEvol7.3.5 -n2 -ptreefile -v1 0.5 -a1 0.7 -i0.5

We strongly recommend the use of the "parameters" file (instead of the command line, see section 3.2) in order to avoid text errors and to better understand the settings of the target scenario to be simulated.

3.2. Parameters file

If no argument is specified, the program will only use the text file "*parameters", in the* same directory as the executable. If no arguments are specified in the command line, and there is not "*parameters*" file, the program will stop and throw an error. In the "*parameters*" file anything within brackets will be ignored. Several examples of "*parameters*" file are included in the distribution. Using file "*parameters",* just type:

./CoalEvol7.3.5

3.3. Arguments

Possible arguments for *CoalEvol* are (# means number, *NAME* means a word):

General settings
-n# : Number of replicates
The number of samples to be generated. Each sample is an independent realization of the evolutionary process. This specification is mandatory. Example: *-n10*

Coalescent settings
-s# #: Sample size; Number of sites
The number of sequences to be simulated for each sample and the total sequence length in nucleotides, codons or amino acids, of the sequences. Example: -s6 102

-e# #: Effective population size; Haploid / Diploid
The effective size (*N*) of the population from which the sample was theoretically drawn. If there are several demes, this argument is the effective population size for each deme. The second number indicates if the data can be simulated as haploid (1) or diploid (2). Example: *-e100 1*

-=# : Tip dates
Tip nodes (sample) can have different times. For example, 4 samples: 1995:sequence 1; 2003: sequences 4 and 6; 1997: sequences 2 and 3; 2001: sequences 7 and 8. This option does not work if there is any convergence of demes at younger times. Example (of above): - *=4 1995 1 1 2003 4 6 1997 2 3 2001 7 8*

-/# : *Generation time*
The time per generation. Example: *-/300*

-g# (# # #) : Demographics. Exponential growth rate or Demographic periods
The first number specifies the model, exponential growth rate (0), demographic periods for a single and global population (1), constant population sizes for each population of a

species/populations tree (2), variable population sizes for each population of a species/populations tree (3). These parameters are looking back in time, so it is not a good idea to specify a negative growth rate for the last period, as the coalescent time could become infinite in the past.

Exponential growth rate per individual per generation, after "0" the growth rate must be specified. Example: *-g0 1e-5* (= *-g0 0.00001*)

Demographic periods for a single population, after "1" the user has to specify the number of periods (from present to past) and *N* during those periods. The first number here specifies the number of periods. For each period should be three consecutive numbers on the size *N* at the beginning and at the end of the period, and the duration of the period in generations. Example: *-g1 3 1000 1250 1000 1300 1550 2000 1560 1000 3000*

The exponential growth rate during the period (positive or negative) will be deduced from the specified *N* at the beginning and at the end of the period. The growth rate derived for the last period will continue into the indefinite past. This implementation is borrowed from [3]. This option is incompatible with the exponential growth rate option (*-g0*). Again, these parameters are looking back in time, so it is not a good idea to specify a negative growth rate for the last period because the coalescent time could become infinite in the past.

Constant population size for each population of a populations/species tree, after "2" the user has to specify the number of populations and then, the population size for each population following a consecutive numerical order (e.g., the first population size is for the population "1"). See the parameters "*Migration model and population structure*" and "*Species/population tree*", such parameters are mandatory for this option. In the example, 7 populations, the first one with a population size = 1000 and the last one with a population size = 700. Example: *-g2 7 1000 500 900 800 1300 1000 700*

Variable population size for each population of a populations/species tree, after "3" the user has to specify the number of populations and then, the initial and final population size for each population following a consecutive numerical order (e.g., the first population size is for the population "1"). That specification must be introduced for all populations excepting the last one (the most ancestral one), this population will consider a constant size, so this population requires only one number. See the parameters "*Migration model and population structure*" and "*Species/population tree*", such parameters are mandatory for this option. In the example, 7 populations, the first one with an initial population size = 1000 and a final population size = 1200, the second one with an initial population size = 500 and a final population size = 600, the last one with a constant population size = 900. Example: *-g3 7 1000 1200 500 600 900 1000 100 200 800 900 2000 1900 900*

-q# # (#): Migration model and population structure

The first number specifies the migration model (island model=1, stepping-stone model=2, continent-island model=3). The second number specifies the total number of populations sampled. The next *n* numbers specify the number of individuals (or sequences) per deme (note that the sample size (-*s*) must be equal to the sum of the sequences here specified). For the island-continent model, deme #1 will be the continent while the other demes will be the islands (further details in the section on migration). Example: -q2 2 3 3 (a stepping-stone model, two demes with three samples each.

-t# (#)(#): Migration rate

This parameter introduces the migration rate, which can be constant or variable with time according to temporal periods. The first number specifies the number of temporal periods, then:

For only 1 period, the second number is the migration rate (constant).
Example: -t1 0.001 (only 1 period with migration rate = 0.001).

For more than 1 period, for each period the second number is the time for the beginning of the new migration rate and the third number is the corresponding migration rate.
Example: -t2 100 0.001 0.005 (2 periods, the first period occurs from t = 0 to t = 100 with a migration rate = 0.001, the second period occurs from t = 100 to the end of the simulation with a migration rate = 0.005).
Example: t3 100 800 0.002 0.001 0.003 (3 periods: period from t = 0 to t = 100 with migration rate = 0.002, period from t = 100 to t = 800 with migration rate = 0.001, period from t = 800 to the end of the simulation with migration rate = 0.003).

-%# (# # #) : Species/population tree
The first number specifies the total number of convergent events. For each convergence event should be three consecutive numbers. The first and second numbers are the numbers of the species/populations to converge. The third number is the convergence time. With this option the user can build the evolutionary tree of the species/populations and it is only available when a migration model is specified (though the migration rate can specified as zero). See theoretical section for further details. Examples: -%1 1 2 2000 (A single event, for 2 initial populations (-q2), convergence of pop 1 with pop 2 at time 2000 to create a new pop 3). -%3 1 2 400 3 4 1900 5 6 2000 (3 events, for 4 initial demes (-q4) convergence of pop 1 with pop 2 at time 400 to create a new pop 5, convergence of pop 3 with pop 4 to create a new pop 6 at time 1900, convergence of pop 5 with pop 6 at time 2000 to create a final ancestral pop 7).

-r# : Homogeneous Recombination rate
The homogeneous recombination rate per site (nucleotides, codons or amino acids) per generation. All sites will share this same rate. This is also the background of recombination when simulating recombination hotspots. Example: *-r2e-6* (= *-r0.000002*)

-hNAME : Recombination hotspots
This option activates recombination hotspots. Parameters for recombination hotspots must be introduced in the file here indicated (see section 3.4). Example: *-hUserHetRec*

-w# : Fixed number of recombination events
This option fixes the number of events of recombination per replicate, so every sample will have the same number. What it does is to filter out replicates with a different number of recombination events, so it will take more time. We recommend to use this option with careful, with sense according to the specified recombination rate (e.g., do not fix the number of events of recombination = 0 when the specified recombination rate is very high, or do not fix this number to a high value when the recombination rate is low or zero), otherwise the execution may never ends until the program crashes. Example: *-w2*

-u# : Mutation rate
Mutation rate per site per generation. Example: *-u0.0015*

-o# : outgroup branch length
If this option is specified the program simulates an outgroup sequence that evolves independently from the sample, along a branch of the specified length. By default it is not simulated. Example: *-o0.1*

User-specified tree/s settings
-pNAME : User-specified tree/s
This option activates user-specified tree/s. A tree for each fragment can be user-specified. Then, this/these tree/s will be applied to evolve sequences. Note that when this option is activated the coalescent is not simulated and all coalescent input settings are ignored. Parameters for user-specified tree/s must be introduced in the file here indicated (see section 3.4). Example: *-ptreefile*

Substitution model settings
*-f4 # # # #, -f12 # # # # # # # # # # # #, -f20 # or -f0 # # (# #): Nucleotide frequencies, Nucleotide frequencies for each codon position (*3×4*), amino acid frequencies and, frequencies by categories (CAT models [37]).*
The nucleotide frequencies A C G T are specified in this order.
The amino acid frequencies are specified in the order: A R N D C Q E G H I L K M F P S T W Y V.

It is possible introduce 4, 12 or 20 frequencies. When 4 frequencies are introduced in codon models such values are applied to the three positions in the codon. Only for codon models, the frequencies of A_p C_p G_p T_p can be specified in turn for each of the three positions (3×4). 20 frequencies only works with amino acid models. By default all frequencies are equal.
Four frequencies example: *-f4 0.4 0.2 0.1 0.3*
Twelve frequencies example: *-f12 0.3 0.2 0.4 0.1 0.2 0.2 0.3 0.3 0.2 0.2 0.1 0.5*
Twenty frequencies example: *-f20 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.06*

Frequencies can change across sites according to user-defined categories (CAT) by the argument -*f*0. After "*f*", a 0 means that a CAT model will be specified. The second number indicates how many frequencies will be introduced (4 for 1x4 -DNA-, 12 for 3x4 -codon-, 20 for 1x20 -amino acid-). The third number introduces the number of categories. Then, for each category the user has to specify the proportion and the frequencies. For example:
Nucleotide data, 3 categories, the first one with a proportion of 0.5, the second one 0.25, the last one 0.25: *-f0 4 3* ***0.5*** *0.35 0.15 0.2 0.3* ***0.25*** *0.25 0.25 0.27 0.23* ***0.25*** *0.5 0.15 0.15 0.20*
Codon data, 2 categories, the first one with a proportion of 0.6, the second one 0.4: *-f0 12 2* ***0.6*** *0.1 0.4 0.3 0.2 0.4 0.4 0.1 0.1 0.3 0.3 0.3 0.1* ***0.4*** *0.1 0.3 0.3 0.2 0.4 0.4 0.1 0.1 0.3 0.2 0.4 0.1*
Amino acid data, 2 categories, the first one with a proportion of 0.7, the second one with 0.3: *-f0 20 2* ***0.7*** *0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06* ***0.3*** *0.05 0.03 0.07 0.04 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.07*

-v1 #, -v6 # # # # # #, -v12 # # # # # # # # # # # #: relative substitution rates

The first number specifies the total number of relative substitution rates, which can be 1 (for transition/transversion rate ratio), 6 (for relative symmetric substitution rates) or 12 (relative asymmetric substitution rates).

If the first number = 1, the second number is the transition/transversion rate ratio. When set to 0.5, the probability of transitions and transversions are the same, like in "*JC*" [11] or "*F81*" [43] models. Example: *-v1 2.1*

If the first number = 6, the following 6 numbers are the relative symmetric substitution rates A↔C, A↔G, A↔T, C↔G, C↔T and G↔T. Example: *-v6 1.0 5.0 1.0 1.0 5.0 1.0.*

If the first number = 12, the following 12 numbers are the relative asymmetric substitution rates AC, CA, AG, GA, AT, TA, CG, GC, CT, TC, GT and TG. Note that some calculations under this option can be problematic (complex eigenvalues/vectors) if rates are too asymmetric, especially dealing with codon models. Example: *-v12 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 1.0 1.0*

-m# (#): GY94 [14-16], *MG94* [19] and *HB* [21] *codon models*
This option controls the implementation of a variety of parametric codon models:

- MG94 (-m0 # #): After a 0, the following two numbers are the synonymous and nonsynonymous rates. Example –m0 0.63 0.23

- GY94 M0 (-m1 # #): ω (nonsynonymous/synonymous rate ratio ($\omega = d_N/d_S$)) is constant across sites. The first number specifies the model (1), and the second number is the value of ω. Example -m1 1.63

- GY94 user-given categories (-m2 # (# #)): ω varies across sites according to some user-defined categories. The first number specifies the model (2) and the second number indicates the number of categories. The next numbers are, for every category, the ω value and its corresponding probability. Example –m2 3 1.2 0.2 0.4 0.3 1.3 0.5

- GY94 M1 (neutral) (-m3 # #): Two categories ($\omega 0$ *P0,* $\omega 1$ *P1*), two values of ω with a proportion each category. The first number specifies the model (3), the second number indicates the proportion of the $\omega 0$ *(P0*), and the third number indicates the value of $\omega 0$. Remember that for M1, $\omega 1$ = 1.0 and *P1=1-P0.* Example –m3 0.8 0.9

- GY94 M5 (discrete gamma) (-m4 # # #): ω varies across sites according to a discrete gamma distribution [44]. The first number indicates the model (4), and the second number indicates the number of equally probable categories (no more than 10) for the gamma distribution. The third number is the alpha shape of the gamma distribution and the last number is the average ω. Example –m4 3 0.78 1.4

- GY94 M5 (continuous gamma) (-m5 # #): ω varies across sites according to a continuous gamma distribution [44]. The first number indicates the model (5). The second number indicates the alpha shape of the gamma distribution and the third number is the average ω. Note that this model may run slow because the "continuous" distribution. Example –m5 0.91 1.42

- GY94 M6 (continuous two gammas) (-m6 # # # #): ω varies across sites according to two continuous gamma distributions. The first number indicates the model (6). The second number indicates the proportion of the first distribution, the third and fourth numbers indicate the alpha shape and average ω for the first gamma distribution, the last number indicates the alpha shape of the second gamma distribution. Note that this model can run slow. Example –m6 0.5 5 3 0.54

- GY94 M7 (continuous beta) (-m7 # #): ω varies across sites according to a beta distribution. The first number specifies the model (7), the second number is "*p*" and the last number indicates "*q*" of the beta distribution. Example -m7 1.4 0.5

- GY94 M8 (continuous beta&omega) (-m8 # # # #): *ω* varies across sites according to two categories, one is a beta distribution and the other is a *ω* fixed for the user. The first number specifies the model (8), the three next numbers indicate the category of the beta distribution, the *P0* (proportion the sites to evolve by the beta distribution), "*p*" and "*q*" for the beta distribution. Finally, the last number indicates the *ω1* fixed for the user for the second category. Remember that *P1=1-P0*. Example -m8 0.05 5.01 3.05 2.00

- GY94 M9 (continuous beta&gamma) (-m9 # # # # #): *ω* varies across sites according to two categories, the first one indicates a beta distribution and the other one indicates a gamma distribution. The first number specifies the model (9), the following three numbers indicate the category of the beta distribution, *P0* (proportion the sites to evolve by the beta distribution), "*p*" and "*q*" for the beta distribution. Finally, the last two numbers indicate the alpha shape and the average *ω* of the gamma distribution. Remember that *P1=1-P0*. Example -m9 0.80 5.5 3.3 1.9 3.8

- GY94 M9 (continuous beta&gamma+1) (-m10 # # # # #): *ω* varies across sites according to two categories, a beta distribution and a gamma distribution + 1. The first number specifies the model (10), the following three numbers indicate the category of the beta distribution, *P0* (proportion the sites to evolve by the beta distribution), "*p*" and "*q*" for the beta distribution. Finally, the last two numbers indicate the alpha shape and the average *ω* of the gamma distribution. Remember that *P1=1-P0*. Example -m10 0.70 5.5 3.3 1.8 2.7

- HB (-m20 (#) #): Using this argument the HB codon model is specified. The first number specifies the HB model (-m20). The following number can be "20" (for constant amino acid frequencies across sites) or "0" (for variable frequencies across sites, CAT). The amino acid frequencies are specified by the following order: A R N D C Q E G H I L K M F P S T W Y V.

For constant amino acid frequencies across sites, just follow the amino acid frequencies. Example -m20 20 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06

For variable frequencies across sites, CAT, follow the number of categories, and the proportion and amino acid frequencies for each category. Note that in this case CAT should be also specified in the nucleotide or codon frequencies. Example (**3** categories) -m20 0 **3** **0.4** 0.05 0.05 0.07 0.03 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06 **0.2** 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.02 0.08 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06 **0.4** 0.03 0.07 0.05 0.05 0.05 0.05 0.05 0.05 0.03 0.07 0.04 0.06 0.05 0.05 0.05 0.05 0.05 0.05 0.04 0.06

-_#NAME : GY94 (M0-M10) codon models with amino acid constrains

With this option a GY94 codon model (which must be previously specified by using the argument "-m") can be modified with amino acid constraints based on any empirical amino acid matrix, according to the algorithm developed in [18]. Amino acid models available are: *Blosum62* [24], *CpRev* [25], *Dayhoff* [26], *DayhoffDCMUT* [27], *HIVb* [28], *HIVw* [28], *JTT* [29], *JonesDCMUT* [27], *LG* [30], *Mtart* [31], *Mtmam* [32], *Mtrev24* [33], *RtRev* [34], *VT* [35], *WAG* [36] and UserEAM (this last one is the name of the input file with a user-defined model). Example -_WAG. Example -_ UserEAM

-l# (#) : Variable ω per branch

With this option a GY94 codon model can simulate variable *ω* per branch. There are two main options based on the codon model selected before using the option "-m":

For only the M0 codon model, where *ω* is constant among sites (-m1 should be introduced before). Here *ω* can vary across branches according to user-defined categories (1), a continuous gamma distribution (2) or a continuous beta distribution (3). (1) For user-defined categories the first number must be 1, the following number is the number of categories, and then, for each category just introduce the *ω* and the proportion to occur. Example -l1 3 0.1 0.2 3.4 0.3 1.0 0.5 (3 categories: *ω*=0.1 and P=0.2, *ω*=3.4 and P=0.3, *ω*=1.0 and P=0.5). (2) For the variation of *ω* according to the gamma distribution the first number should be 2 and the following numbers are the alpha shape and the average *ω* for the gamma distribution. Example -l2 0.91 1.00. (3) For the variation of *ω* according to the beta distribution the first number should be 3, the second number is "*p*" and the last number is "*q*" of the beta distribution. Example -l3 1.4 0.7

For the other codon models (M1-M10) models, where *ω* varies among sites (-m2 to –m10 must be selected before). Here *ω* can vary across branches according to the same model applied for the variation among sites. Just type 4. Example -l4

-zNAME : empirical codon model

Empirical codon model: three empirical codon models are implemented in *CoalEvol*: ECMrest [22], ECMunrest [22] and ECMSchn2005 [23]; in this case just specify one of these names to apply the corresponding codon model. But in addition, it is possible to specify a user-defined empirical codon model by an input file (see section 3.4); here specify the name of the input file with the empirical codon model. Example: -*zECMSchn2005*. Example: *-zUserECM*.

-@NAME : empirical amino acid model

This option allows for the simulation of amino acid data. The empirical amino acid models implemented in *CoalEvol* are the following: *Blosum62* [24], *CpRev* [25], *Dayhoff* [26], *DayhoffDCMUT* [27], *HIVb* [28], *HIVw* [28], *JTT* [29], *JonesDCMUT* [27], *LG* [30], *Mtart* [31], *Mtmam* [32], *Mtrev24* [33], *RtRev* [34], *VT* [35], *WAG* [36]; in this case just specify the name of the empirical amino acid model. But in addition, it is possible to specify a user-defined empirical amino acid model by an input file (see section 3.4); here specify the name of the input file with the empirical amino acid model. Example: -@JTT. Example: -@UserEAM.

-a# #: alpha shape of the gamma distribution

A gamma distribution (+*G*) is used to simulate substitution rate variation among sites [44]. Alpha is the shape of this distribution. Note that smaller alphas imply stronger rate variation. Specify 1 followed by alpha. Example: -a1 0.7.

-i# : proportion of invariable sites

A proportion of sites can be set to be invariable (+ *I*). Example: *-i0.3*

-xNAME : GMRCA input file

The user can specify a GMRCA sequence in a text file (see section 3.4). This file must contain only a DNA or amino acid sequence (depending on the substitution model applied), and its length has to be equal to the number of sites specified before. By default, the G*MRCA* is simulated from the nucleotide or amino acid frequencies (see above, -f argument). Example: *-xseqGMRCA*

Output settings

-bNAME : print sequences
When this argument is invoked, aligned sequences are printed to the specified text file in the *Results* folder. Example: *-bdatafile*

-c# # #: format of print sequences
This option specifies the format of the output alignments. The first argument indicates *Phylip* sequential, *Fasta* and *Nexus* format (1-3, respectively). The second argument specifies if alignments for each replicate should be printed in a single file (0) or in different files (1). The third argument prints the ancestral sequence of all internal nodes (1) or not (0). Example: - c1 1 0 (that means *Phylip* format, a file for each replicate, internal nodes are not printed).

-$: print catMRCA/GMRCA
This option will print the catMRCA/GMRCA sequences in files called *catMRCA* (MRCA fragments concatenated*)* and GMRCA, respectively. Files will be included in the *Results* folder. Example: *-$*

-jNAME : print genealogies
When this option is specified, genealogies (also for each recombinant fragment) are printed in *Newick* format. File will be included in the *Results* folder. Example: *-jtrees*

-kNAME : print times
When this argument is specified the coalescent times for each genealogy will be printed to the specified text file in the *Results* folder. This option will slow down the simulations. Example: *-ktimes*

-**NAME : print ARGs*
When this argument is specified the ancestral recombination graph (ARG) will be printed to the specified text file in the *Results* folder. Then, this file can be directly introduced to the *NetTest* web server in order to visualize the ARG (see section 4.2.2).
The version 7.4.0 includes printing the sequences of the tip nodes with information about the past recombination events that occurred in those sequences. This info is printed with the option of network file (*NetworkFile) in fasta format (c2 1 0).
Example: -**NetworkFile*

-dNAME : print breakpoints
When this argument is specified the position of recombination breakpoints are printed to the specified text file in the *Results* folder. Example: *-dbreakpoints*

-+ *: print Simulate omegas per site/branch*
When this argument is specified the omegas simulated per codon and/or branch are printed to the specified output file in the *Results* folder. Example: -+

Other settings
-y# : noisy level
This option controls the level of information that will be printed to the screen.

0 : does not print run information, just the simulation progress.
1 : run settings and run information summarizing the simulations.
2 : calculation status, initial populations and event information, run settings for each sample

3 : print ancestral status for each sequence at each event + *MRCA* status, tip dates insertion, demes.

4 : potential recombining locations (*g* and *G* vectors) and information about recombinant fragments evolution.

Note that higher levels of noisy will slow down the simulations. The default level is 1 Example: *-y1*

-## : Seed

Seed for the random number generator. If no seed is specified, the computer clock will be used. When a seed is fixed the process can be always reproduced using same settings. Example: *-#386658297*

3.4. Additional input files

The following input files are optional and can only be introduced as an input file in the same directory as the executable (cannot be specified in the command line by arguments). The name of these files must be introduced in the main settings (parameters file or command line).

3.4.1. User-specified tree/s. No coalescent simulation

Instead of the coalescent simulation, the user can specify a tree for each fragment or partition in order to evolve sequences over the corresponding specified tree/s. This possibility is convenient when the user already has tree/s (e.g., inferred from real data) or to evade coalescent assumptions such as the molecular clock.

The tree/s must be incorporated into an input file which should be specified in the parameters file, by the argument –p (e.g., *-ptreefile*). Then the input file is constructed by the following structure. Each line must contain a range of sites (nucleotides, codon or amino acids) and a tree. Such a range specifies the first and last site where the following tree is going to act. Necessarily, any site should be covered by a tree. The file does not allow empty lines. Importantly, the number and name of taxa should be the same for all trees. Trees should be rooted. An example is shown below. Sites from position 1 to position 1000 will be evolved under the first tree. For sites from position 1001 to 2000, and 2001 to 3000, the second and third trees will be applied, respectively.

```
1 1000 ((taxonA:0.1,taxonB:0.1):0.4,(taxonC:0.1,taxonD:0.1):0.5,taxonE:1.0);
1001 2000 ((taxonA:0.1,taxonC:0.1):0.4,(taxonB:0.1,taxonD:0.1):0.5,taxonE:1.0);
2001 3000 ((taxonA:0.1,taxonD:0.1):0.4,(taxonC:0.1,taxonB:0.1):0.5,taxonE:1.0);
```

Note that by this procedure the total number of sites, number of taxa and name of taxa are here introduced. In the example, a total of 3000 sites with 5 taxas (taxonA, taxonB, taxonC, taxonD, taxonE). When this option is applied all coalescent input settings are ignored (see section 3.3).

3.4.2. Recombination hotspots

The user can optionally simulate recombination hotspots following the algorithm implemented in SNPsim [4]. This file must be specified in the main settings (*-hNAME*)

and has to be located in the same directory as the program. An example file (UserHetRec) is included in the folder with examples.
Possible arguments for the *recombination hotspots file* are (# means number):

-k# : Hotspot recombination rate
Expected recombination rate at the hotspots sites. If the hotspots are homogeneous (option *-t#* is not invoked) all the hotspots have the same rate. Example: *-k1e-4* (= -*k0.0001*)

-h# : Expected number of hotspots
This is the expected number of hotspots for a given sample in the absence of interference. This parameter corresponds to the intensity parameter for a Poisson distribution from which the actual number of hotspots is drawn. For a given sample, the actual number of hotspots will change around this value. It does not have to be an integer. NOTE: When interference is specified we need to divide this number by the interference interval (*-z#*) to obtain the expected number of hotspots. It does not have to be an integer. Example: *-h1.1*

-q# : Fixed number of hotspots
This option fixes the number of hotspots inside the region of interest, so every sample will have the same number. In this case the hotspot locations are chosen from a uniform distribution. If the hotspots overlap, they will be displaces to the closest available location. Note that in this case no recombination events will originate from a hotspot located outside the region of interest. Example: *-q3*

-v# : Hotspot imprecision
The hotspot imprecision corresponds to the variance of a Normal distribution for the specific site to recombine around the hotspot center (chosen by a Poisson process). The bigger the imprecision, the wider is the hotspot. If the imprecision is 0, all the recombination events happen exactly at the hotspot center. See figures 2 and 3 (theoretical section). Example: *-v0*

-m# : Hotspot width
This option specifies the width of the hotspots. In this case any site in the hotspot has the same probability of recombination. If the width is 1 all the recombination events happen exactly at the hotspot center. This parameter has to be bigger than 0. See figures 2 and 3. Example: *-m1*

-t# : Hotspot heterogeneity
This parameter indicates that there is hotspot heterogeneity, that is, hotspots may have different recombination rates. This heterogeneity is accomplished through the use if the continuous gamma distribution. The shape parameter of this distribution (%) will control the strength of this heterogeneity. The smaller the shape the strongest the heterogeneity. This is similar to the application of Yang [44]. Example: *-t0.5*

-z# : Hotspot interference
This parameter indicates whether the location of the hotspots is not independent of each other. If this parameter is 1 there is no interference, if it is between 0 and 1 hotspots tend to cluster, and if it is bigger than 1 hotspots will tend to be pushed away from each other. Example: *-z1*

3.4.3. GMRCA sequence user defined

By default, the grand most recent common ancestor (GMRCA) sequence is simulated according to the nucleotide frequencies. However, the user can optionally specify a GMRCA sequence from a text file (the name of this file must be specified in the main settings (*-xNAME*), which has to be located in the same directory as the program. The file just contains a single sequence. An example file (seqGMRCA) is included in the examples.

3.4.4. Empirical codon matrix user defined

The user can optionally specify an empirical codon matrix from a text file, which has to be located in the same directory as the program. An example file (userECM) is included in the folder with examples. This file consists of two sets of parameter values:
First, a 61 × 61 matrix with the substitution rates (values must start after a "c"). Second, 61 codon frequencies (values must start after a "z"). Importantly, the order of codons must be the following: AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA CAC CAG CAT CCA CCC CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG GCT GGA GGC GGG GGT GTA GTC GTG GTT TAC TAT TCA TCC TCG TCT TGC TGG TGT TTA TTC TTG TTT.

3.4.5. Empirical amino acid matrix user defined

The user can optionally specify an empirical amino acid matrix from a text file, which has to be located in the same directory of the executable. An example file (userEAM) is included in the folder with examples. This file consists of two sets of parameter values. First it contains the substitution rates for each amino acid (values must start after the letter of the corresponding amino acid). Second, 20 amino acid frequencies can be introduced (values must start after a "z"), but importantly the order of amino acids must be the following: A R N D C Q E G H I L K M F P S T W Y V.

3.5. Default settings

By default *CoalEvol* will simulate 10 samples of 6 individuals with 201 sites, a constant effective size of 1000, constant size, no recombination, no migration and a mutation rate of 1e-7 under the *JC* nucleotide substitution model with each nucleotide frequency value equal to 0.25. Noisy level is 1. The sequences will be printed to the *sequences* output file in the *Results* folder. To run the program with the equivalent arguments we would type:

```
./CoalEvol7.3.5 -n10 -s6 201 -e1000 2 -u1.0e-07 -f4 0.25 0.25 0.25 0.25 -bsequences -y1
```

Table 1. Key arguments for CoalEvol. The user can specify several parameters to define different simulation scenarios. These arguments can be entered in the command line or in the parameters text file.

Parameter	Arg	Example value	Application
Number of replicates	n	200	All
Sample size	s	8	All
Number of sites (bp, codons or amino acids)	s	102	All
Effective population size	e	1000	All
Haploid / Diploid	e	1	All
Tip dates	=	2 1995 1 3 2003 4 8	All[1]
Generation Time	/	400	All
Exponential growth rate	g	$2.1x10^{-5}$	Demography
Demographic periods[2]	g	1000 5000 200	Demography
Constant population size per population (species/populations tree)	g	1000	Demography /Populations/Migration
Variable population size per population (species/populations tree)	g	500 600	Demography /Populations/Migration
Migration model	q	1	Migration
Number of populations	q	"4" 2 2 3 1	Populations/Migration
Population structure	q	4 "2 2 3 1"	Populations/Migration
Migration rate (constant or variable with time)	t	1.2×10^{-4} or 100 "0.001 0.005"	Migration
Species/populations tree[3]	%	1 2 5000	Populations/Migration
Homogeneous recombination rate	r	5×10^{-6}	Homogeneous recombination/Recombination hotspots
Fixed number of recombination events	w	3	Homogeneous recombination/Recombination hotspots
Recombination hotspots	h	UserHetRec	Recombination hotspots
Mutation rate	u	5.1×10^{-4}	All
Outgroup branch length	o	0.1	All
User-specified tree/s	p	Treefile	Genetic data simulation
Nucleotide frequencies CAT model	f	0.4 0.3 0.1 0.2	Nuc/codon models
Transition / transversion ratio	v	2.1	Nuc/codon models
Relative symmetrical	v	1.0 2.3 2.1 3.0 4.2	Nuc/codon models

substitution rates		1.0	
Relative asymmetrical substitution rates	v	0.1 0.2 0.3 0.4 0.9 0.6 0.9 0.8 0.9 0.1 0.2 0.3	Nuc/codon models
Nucleotide frequencies for each codon position CAT model	f	0.1 0.4 0.3 0.2 0.4 0.4 0.1 0.1 0.3 0.3 0.3 0.1	Codon models
Nonsynonymous / synonymous rate ratio[4] (GY94, constant and variable across sites and/or branches)	m, l	1 1.8, 1 3 0.1 0.2 3.4 0.3 1.0 0.5	Codon models
Synonymous and Nonsynonymous rates (MG94)	m	1.8 0.6	Codon models
Codon models with amino acid contributions	_	WAG	Codon models
HB codon model	m	-	Codon models
Empirical codon models	z	ECMrest	Codon models
Amino acid frequencies CAT model	f	0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.06	Amino acid models
Empirical amino acid model	@	JTT	Amino acid models
Rate variation among sites[5]	a	2.4	All
Proportion of invariable sites	i	0.2	All
User-defined sequence for GMRCA	x	seqGMRCA	All
Print sequences using diverse options	b, c	sequences, 1 1 0	All
Print GMRCA sequence	$		All
Print simulated omega per site and/or per branch	+		Codon models with variable omega per site and/or branch
Print simulated trees	j	trees	All
Print simulated ARG (optionally also the sequences with their past recombination history)	*	NetworkFile	All
Print times for genealogies	k	times	All
Print breakpoints	d	breakpoints	All
Apply a seed	#	3444556	All
Level of output information printed in the screen	y	2	All

[1]In presence of species/population tree, the time of the tip dates must be younger than the time of the species/population tree.
[2]from 1000 to 5000 effective size during 200 generations.
[3]pop 1 is converging with pop 2 at time 5000 to make a new ancestral pop 3.
[4]*dN/dS*. There are multiple options to implement this parameter (see section 3.3).
[5]shape of the gamma distribution.

3.6. Output Files

All output files produced by the program are written to the folder *Results*. There are four optional output files:

- *Sequences file*: Aligned sequences in *Phylip* sequential, *Fasta* or *Nexus* formats.
- *Breakpoints file*: Recombination breakpoints ordered by event time. Disabled when input tree/s are user-specified.
- *Trees file*: Simulated Tree/s in newick format (also for each recombinant fragment).
- *Network file*: Simulated ARG by a branches format (each line contains two connected nodes) [39]. Disabled when input tree/s are user-specified. See section 4.2.2 for additional details. It can include information about the type of recombination nodes (i.e., [1]: tip node, [4]: coalescent node, [3]: recombination node, [2]: outgroup node (optional), [5] root node (optional)).

In addition, under some settings a sequence file can be printed with the history of recombination events: breakpoint (order of event from the present to the past).

- *Times file*: Information about time and branch length for each tree.
- *CatMRCA/GMRCA*: Corresponding sequences catMRCA (concatenated MRCA fragments) and GMRCA (sequence of the GMRCA node) in the ARG.
- *Simulate omegas per site/branch*: Simulated omega value per codon and/or branch. This option can be only used for codon models with variable omega across sites and/or branches.
- *Frequencies per site*: Simulated frequencies assigned to each site when a CAT model is specified. The file will be printed automatically when using CAT models.
- *Settings file*: A summary of applied settings showed in the screen at the end of the simulation. This option must be activated in the source code.

3.7. Solving message errors

Here we show common message errors of *CoalEvol*. However, if you find any other unexpected error, or there is any doubt, do not hesitate to contact us (miguelmmmab@gmail.com) with questions, bugs, comments, suggestions, etc. Your questions are welcome!

- There is a common message of error due to lack of RAM memory in scenarios with very high recombination rates. Note that recombination increases the number of nodes of the ARG and the number of recombinant fragments within nodes (see section 4.2). Then, when ρ (= 4Nrl) is very high, the ARG must save in memory a high amount of information, and with time such amount exponentially increases. Thus, at a given point the machine could not have enough available memory and the program will stop with an error message like the following:

```
malloc: *** mmap(size=1584291840) failed (error code=12)
*** error: can't allocate region
Could not reallocate segments (1584144000 bytes)
```

In this situation we recommend to reduce the value of the following parameters: recombination rate, population size, sequences length and sample size. Of course, the other option is just the use of a machine with more memory.

- Using a fixed number of recombination events keep in mind that the recombination rate introduced should generate a proper number of recombination events. Otherwise the simulation could never finish (because the fixed number of recombination events never occurs) leading to crash.

- Using growth rates or demographic periods you could find an error like:

```
ERROR: Coalescent time (nan) is infinite
	This might suggest that the growth rate is too negative
	and the coalescent time is therefore infinite.
	Try a smaller value
```

This is because the population increases too much going back in time and thus, the coalescent time gets infinite. The best option to solve this error is to reduce the growth rate. In the case of demographic periods this could be done for example by using longer times for the period.

- Asymmetric substitution rates could be problematic in some cases (especially dealing with codon models) due to complex eigenvalues/eigenvectors. In these cases the program writes an output message like the following:

```
complex roots in EigenREV_Codon (k = -1)
```

- Input trees from user. The file with input trees must consider the following aspects: any site should be covered by a tree, no empty lines, the number and name of taxa should be the same for all trees, trees should be rooted.
Very long trees (longer than 6,000 characters) may require a modification of the macro "MAX_LINE" in the source code. Note that these modifications can slow down the program.

4. CoalEvol model

The model implemented in *CoalEvol* is an extension of the coalescent with recombination, demographics and migration based on the neutral Wright-Fisher model [2, 3, 6] and the simulation of nucleotide, codon and amino acid data under Markov models of evolution. Given the specified recombination and migration rates (and other parameters like the effective population size (N)) random genealogies are produced. For any evolutionary scenario intracodon breakpoints in recombination are accepted. Several migration models (island [8], stepping-stone [7] and continent-island [8]) are allowed and migration rate can change with time. The evolution of species or populations can be fixed. Complex demographic histories can be implemented by defining demographic periods in which population sizes augment, reduce, or remain constant. Samples can be collected at same or different times [see, 10] and simulated data can be haploid or diploid. Optionally tree/s can be specified by the user and therefore do not require coalescent simulations (e.g., when a phylogeny is known).
Mutations can be placed upon the previously simulated genealogies, using a large variety of nucleotide (JC, …, HKY, … GTR), codon (GY94 [14] × M1-M10 [15, 16] × any nucleotide model, GY94 with consideration of physicochemical contributions of the derived amino acids [18], MG94 [19] × any nucleotide model [see, 20], HB × any nucleotide model [21] and empirical models [22, 23], variable synonymous/non synonymous ratio rate (dN/dS or ω) per site and/or branch) and amino acid (*Blosum62* [24], *CpRev* [25], *Dayhoff* [26], *DayhoffDCMUT* [27], *HIVb* [28], *HIVw* [28], *JTT* [29], *JonesDCMUT* [27], *LG* [30], *Mtart* [31], *Mtmam* [32], *Mtrev24* [33], *RtRev* [34], *VT* [35], *WAG* [36] and any user defined model) substitution models. Frequencies can also vary per site using CAT models [37]. The result is a random sample of aligned nucleotide, codon or amino acid sequences.

The coalescent proceeds backwards starting from the sample of *s* gametes. Time is scaled in units of 2*N* generations, where *N* is the effective population size. For a given site, without recombination or migration, and under constant population size, the time to the most recent common ancestor (TMRCA) is:

$$E(TMRCA) = 2\left(1 - \frac{1}{s}\right)$$

$$Var(TMRCA) = \sum_{i=2}^{s} \frac{4}{i^2(i-1)^2}$$

The times to a coalescence (CA), recombination (RE) or migration (MI) event are exponentially distributed, with intensity equal to their respective rates (see below). The next event will be the one that would occur before according to these expectations.

$$Time\ to\ CA \propto Exp\left[rateCA\right] \cdot 2N$$

$$Time\ to\ RE \propto Exp\left[rateRE\right] \cdot 2N$$

$$Time\ to\ MI \propto Exp\left[rateMI\right] \cdot 2N$$

4.1. Coalescence

The rate of coalescence depends only on the number of lineages (*k*):

$$rateCA = \frac{k(k-1)}{2}$$

4.2. Recombination

The rate of recombination depends on the population size (*N*) and on the total recombination rate at all valid recombination sites (G). A *valid* recombination site has to have at both sides ancestral material that has not found its MRCA yet. **In codon models, recombination can also occur within codons.**

$$G = \sum_{j=1}^{k} \sum_{i=1}^{L} r_{Gi}$$

$$\rho = rateRE = 2NG$$

Given that a recombination event occurs, a gamete is chosen according to the total rate at potential recombining sites in that gamete. Breakpoint sites are chosen according to the recombination probabilities per site (r_{Gi}). The expected number of recombination events in a panmictic population with constant size is:

$$E(number\ of\ recombination\ events) = \rho \sum_{i=1}^{s-1} \frac{1}{i}$$

Importantly, in the presence of recombination, different regions of the alignment might evolve under different genealogies (Figure 1), which together conform to the ancestral recombination graph. The number of genealogies will be the number of breakpoints + 1.

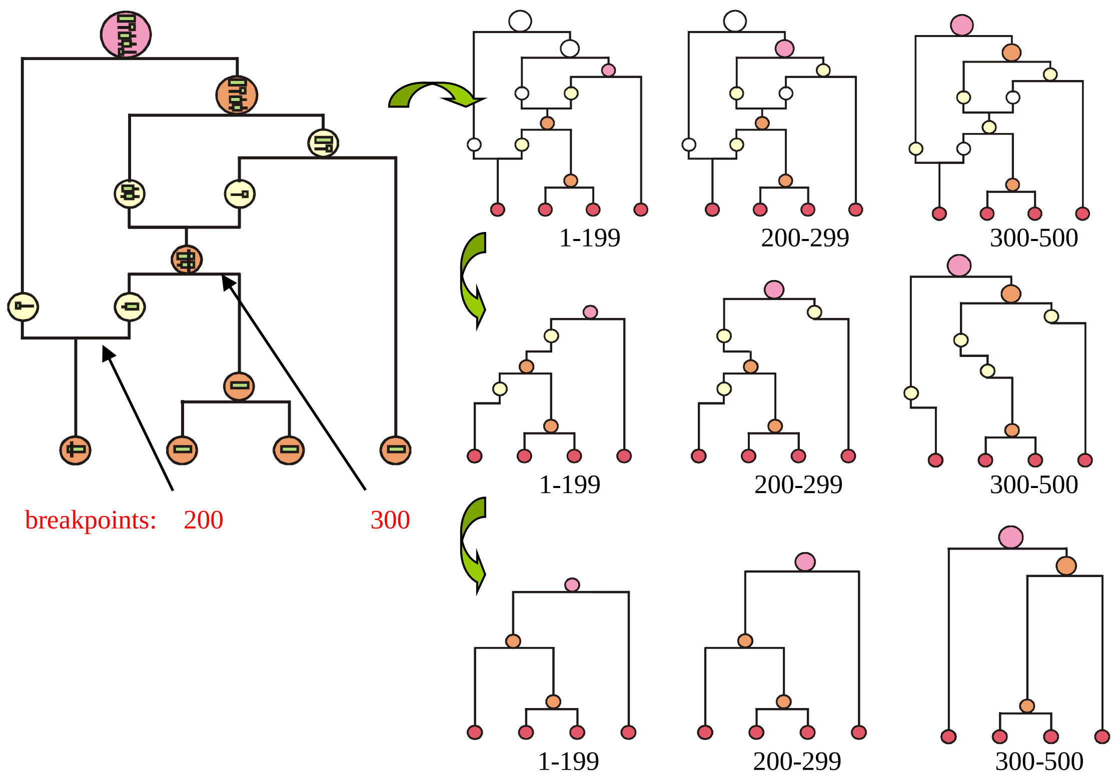


Figure 1. Representation of the ancestral recombination graph and the binary tree embedded.

4.2.1 Recombination hotspots

It is based on *SNPsim* [4], the population genetic model of recombination hotspots developed by Wiuf and Posada, to result in heterogeneous recombination rates along the chromosomes simulated.
Given an expected number of recombination hotspots, a background homogenous recombination rate and a hotspot recombination rate, the algorithm starts by choosing the position and number of recombination hotspots for a particular sample. Adding recombination events around the hotspot center results in the specification of a probability distribution for the recombination rate along the region of interest. Other recombination events coming from recombination hotspots centered outside the region of interest are also considered. This fast simulation results in different recombination rates for different sites along the region (hotspots and coldspots). Next model for hotspot recombination is described following the algorithm [4].

Hotspot recombination model
The hotspots recombination model aims to represent the idea that some sites in a chromosome are more likely to recombine than others ("recombination hotspots"). This general model is composed of two basic recombination rates and a set of hotspots sites. The background recombination rate (R_B) is the per generation recombination rate at any site in the chromosome of length L, while the hotspot recombination rate (R_H) is the

additional per generation recombination rate at the recombination hotspot. X is the number of hotspots.

$$\textit{Background recombination rate}\left(R_B\right) = 4Nr_BL$$

$$\textit{Hotspot recombination rate}\left(R_H\right) = 4Nr_HX$$

$$\textit{Global recombination rate}\left(R_G\right) = R_B + R_H$$

In real life we do not expect the recombination hotspot to be always restricted to the same single site, to have always the same intensity, or to occur independently from other hotspots. The model described above can be generalized to include these relevant biological features. When the hotspot is not restricted to a single site, and under constant population size, the expected number of recombination events is

$$E(\textit{number of recombination events}) = R_G \sum_{i=1}^{s-1} \frac{1}{i}$$

Hotspot location: interference
We will assume that the distance between hotspots is Gamma distributed, $\Gamma(m, \lambda)$, $m > 0$. If $m=1$, we have a Poisson process with intensity λ. Allowing $m \neq 1$ introduces interference. If $m>1$ hotspots are pushed away from each other; if $0<m<1$ they tend to be clustered (although the algorithm will only accept integer values for m). The average number of hotspots in the gene is λ/m (see Figure 2).

Hotspot imprecision
We will consider that the hotspot location actually represents the center of the hotspot, and where recombination is more likely, but that there are also some sites around where recombination occurs with some frequency that decreases with increasing distance from the hotspot center. This can be represented by a Normal or a Uniform distribution for the location of the recombination. When the Normal distribution is used, this has a mean equal to the location of the hotspot center and variance called hotspot imprecision (σ^2) (see Figure 2). When the variance is small the hotspot tends to be narrow. If the variance is 0, the hotspot is 1 bp wide. When the Uniform distribution is used, recombination events occur with the same probability along a given width for the hotspot. In addition there is the possibility of recombination events coming from hotspots located outside the region of interest of length L. To implement this idea we can extend the region of interest by a number of sites K at each end. In the Normal distribution an arbitrary, but seemingly reasonable value for K that assures that wide hotspots outside L are taking into account is

$$K = 10\sqrt{s^2}$$

If the uniform distribution is used, K equals half the width of the hotspot.

Hotspot heterogeneity
In addition, not all hotspots have to be equally "hot". We can model this heterogeneity of recombination rates using a gamma distribution with a mean of 1. The shape of this

distribution (α) will determine the strength of this hotspot heterogeneity. The smaller α, the bigger the heterogeneity of recombination rates at the hotspots.

Implementation

A nice feature of the hotspot model is that it allows for the construction of a distribution of recombination rate along the chromosome ($\Re$)(Figure 2) for every sample.

The algorithm starts by building $\Re$. The first step is to set up the number of hotspot centers (X) along the extended region L + 2K. The average number of hotspots in the gene is λ/m, and when m>1 we use a thinning algorithm to locate these hotspots (Figure 2). If m=1 we distributed the hotspots according to a Poisson distribution with intensity λ.

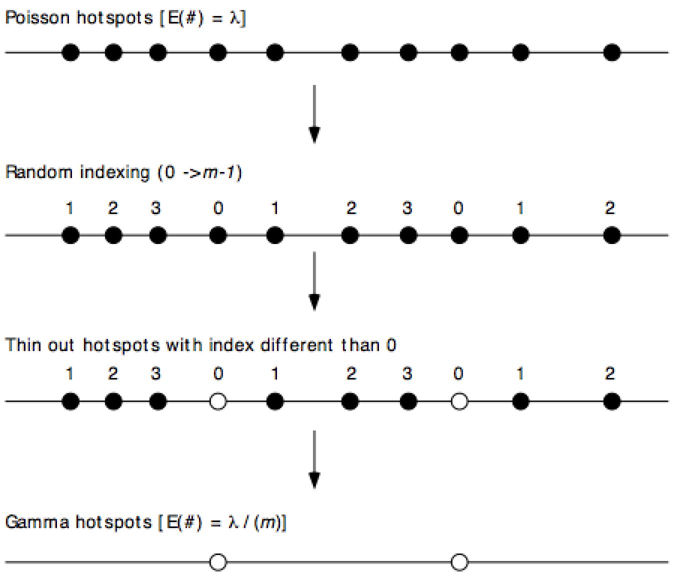

Poisson hotspots [E(#) = λ]
Random indexing (0 ->m-1)
1 2 3 0 1 2 3 0 1 2
Thin out hotspots with index different than 0
1 2 3 0 1 2 3 0 1 2
Gamma hotspots [E(#) = λ / (m)]

Figure 2. Thinning algorithm for the location of hotspots with interference (m=4). The index for the first hotspot site is chosen uniformly between 0 and m-1. This plot was taken from SNPsim manual [4].

Alternatively the user can specify a fixed number of hotspots, that will be located uniformly (see option -q in recombination hotspots file).

The distribution $\Re$ is can be constructed according to a Normal (x_i, σ^2), or a Uniform (hotspot width) distribution. If there is hotspot recombination, a random gamma variable will scale the recombination events at each hotspot. Figures 3 and 4 represent a realization of this process.

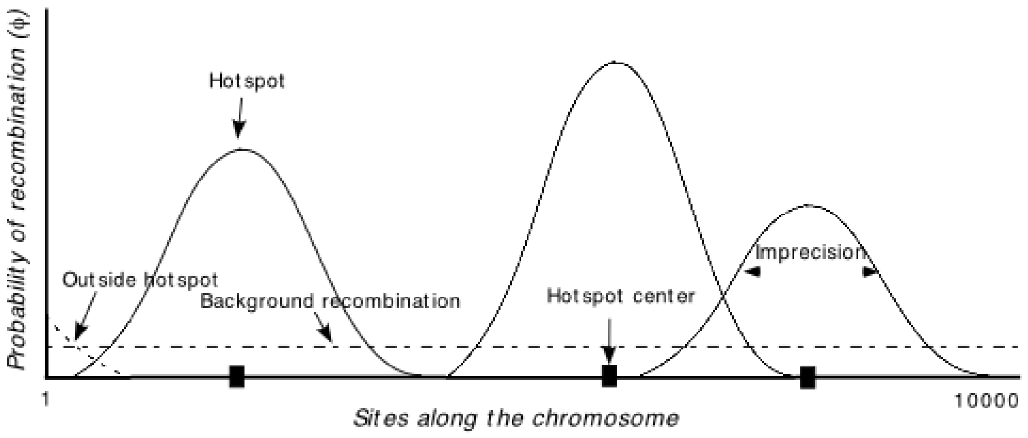


Figure 3. Schematic representation of ℜ in the case of three Normal hotspots for the region of interest (L=10000). In this case the hotspot imprecision is quite big. Note that some recombination probability is contributed by a hotspot outside the region of interest. The background recombination is the same for all sites. In this case there is hotspot heterogeneity. This plot was taken from SNPsim manual [4].

We can use now ℜ now to set the global recombination rate per each site i,

$$r_{Gi} = r_{Bi} + r_{Hi}\varphi_i$$

where φi is the probability of block recombination at site i.

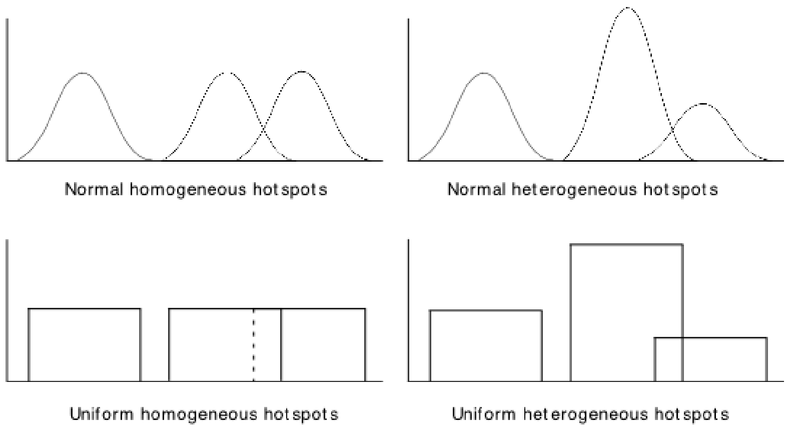

Normal homogeneous hotspots
Normal heterogeneous hotspots
Uniform homogeneous hotspots
Uniform heterogeneous hotspots

Figure 4. Schematic representation of different ℜ. This plot was taken from SNPsim manual [4].

Finally, the total recombination rate at all valid recombination sites is considered to compute times for recombination events (as described before). Breakpoint sites are chosen according to the recombination probabilities per site (r_{Gi}).

4.2.2 Recombination at inter and intra codon positions

Recombination can happen at inter and intra codons [5]. When a recombination breakpoint breaks a codon, that codon is completed using "unsampled ancestral material", this material can affect the evolution of the sampled ancestral material (Figure 5).

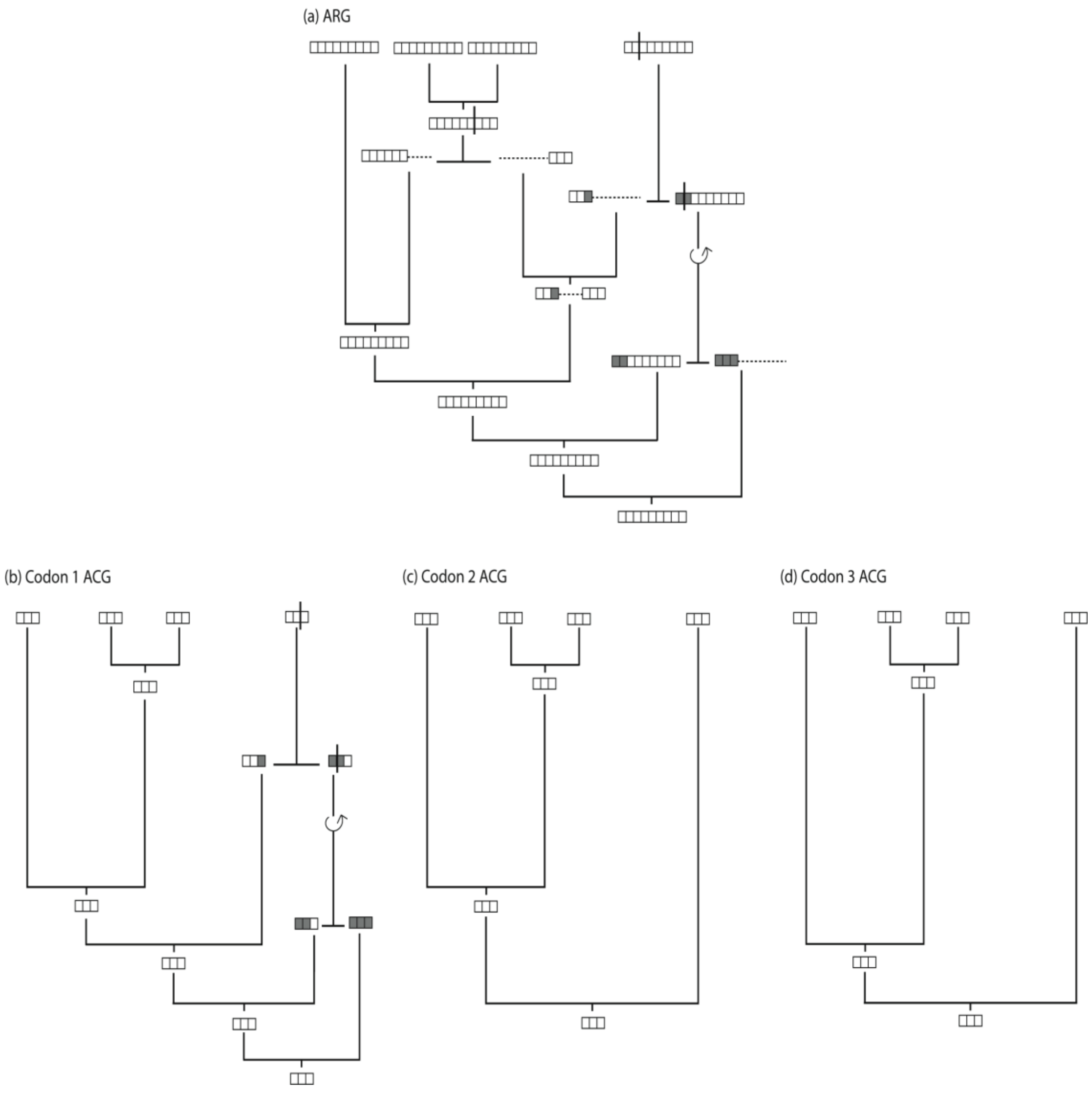


Figure 5. Genealogy of intracodon recombination. ARG is showed in a) and the corresponding ancestral codon graphs (ACGs) are b), c) and d). White squares correspond to sampled ancestral material while grey squares indicates unsampled ancestral material, pointed line means non-ancestral material.

After build the genealogy, the codon evolution works over it. An example of a breakpoint breaking a codon and its evolution is showed in the figure 6.

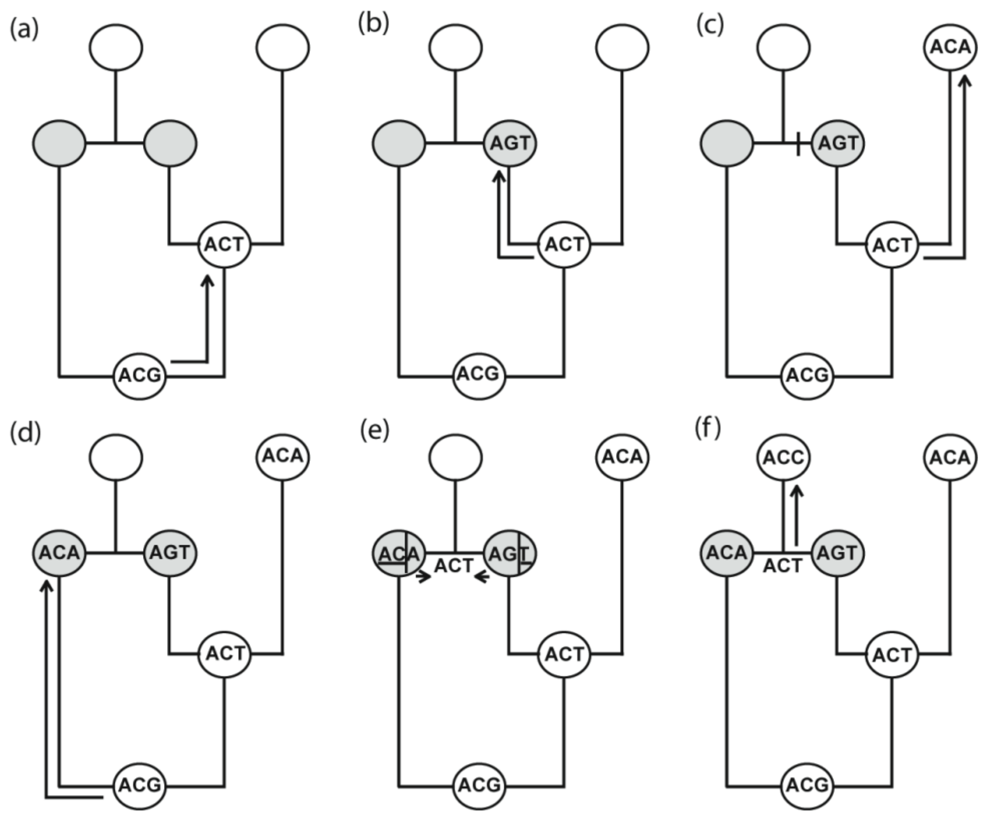


Figure 6. Codon evolution of a codon that suffered an intracodon recombination.

4.2.3 Visualization and analysis of the simulated ARG

The simulated ARG can be directly exported to the *NetTest* web server (http://darwin.uvigo.es/software/nettest/) [39] in order to visualize and analyze the ARG [39, 45]. Figure 7 shows an example of the ARG in branches format (output of *CoalEvol*) and plotted by *NetTest*.

```
6 4_seq00005
6 3_seq00004
7 1_seq00002
8 0_seq00001
9 6
9 7
10 8
10 7
11 8
12 11
13 2_seq00003
13 9
14 12
15 10
16 11
17 5_seq00006
17 13
18 15
19 14
19 16
20 18
21 19
21 20
22 16
22 17
23 21
24 23
24 22
25 23
25 20
26 24
26 25
```

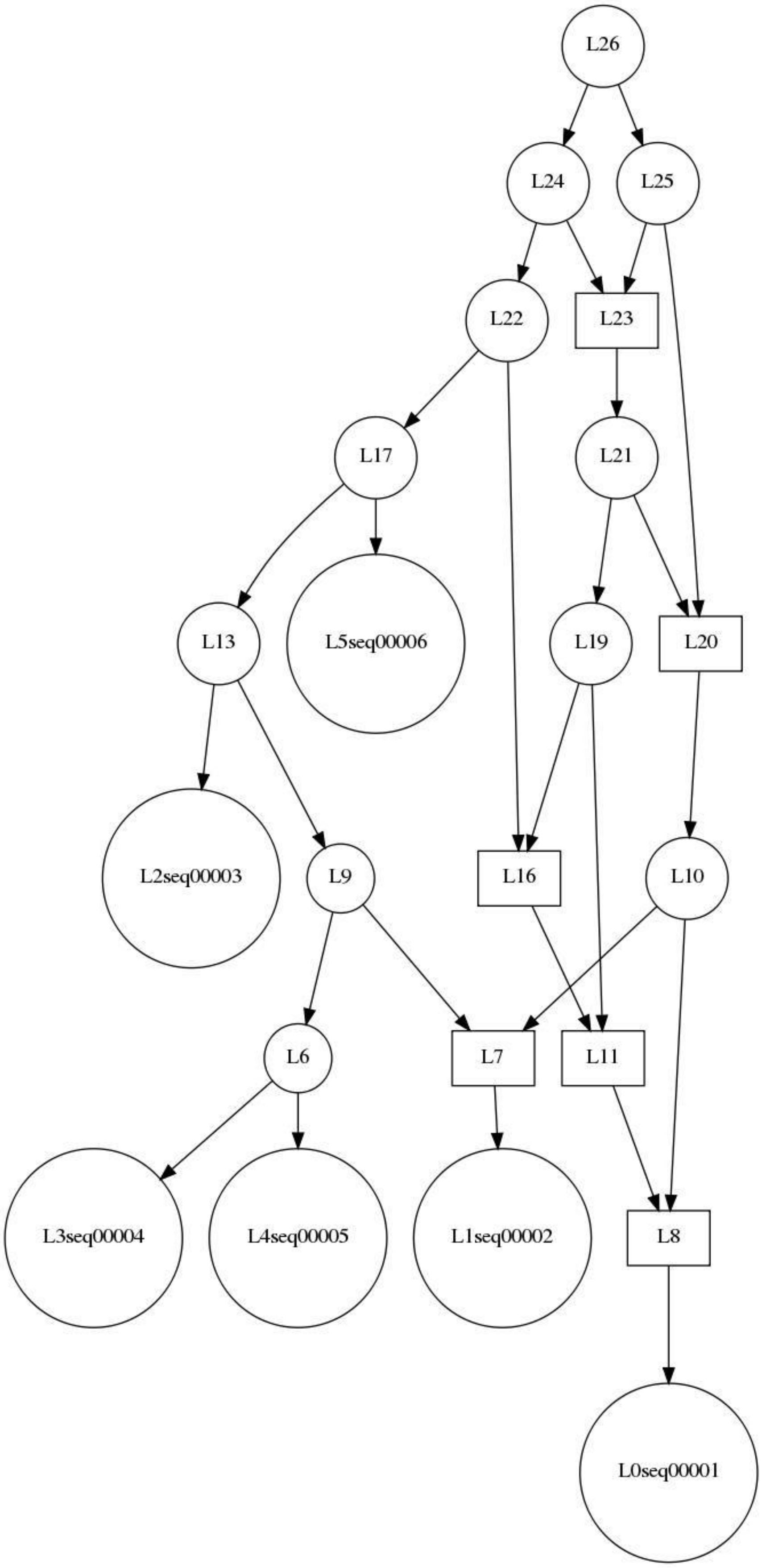


*Figure 7. An ARG simulated by CoalEvol (*branches format, *left) and directly plotted by NetTest (right).*

4.3. Migration

The simplest and most widely used model of population structure is the " finite island model" [6, 8, 46-48] however the stepping-stone [7] and an approximation to the continent-island [8] are also implemented. The rate of migration depends on the population size (*N*), the migration rate per population (*m*) and the number of available populations (*q*).

$$rateMI = 2Nmq$$

Importantly, in the coalescent with migration, lineages can only coalesce with other lineages in the same population. When a migration event occurs, a given lineage changes from one population to another (Figure 8).

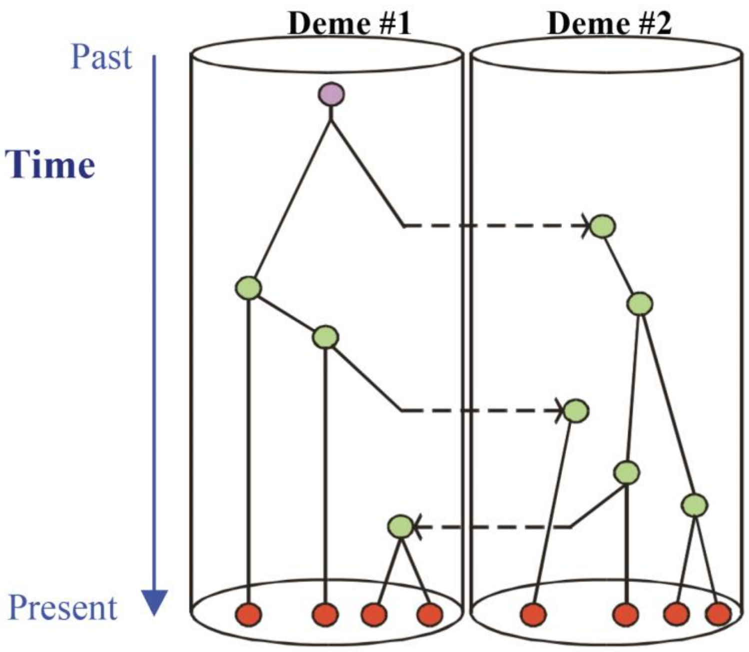


Figure 8. Coalescent with migration for two populations.

4.3.1 Island model

The Island model describes an array of *q* populations, each of constant size (Figure 9). Migration events may occur between any two populations.

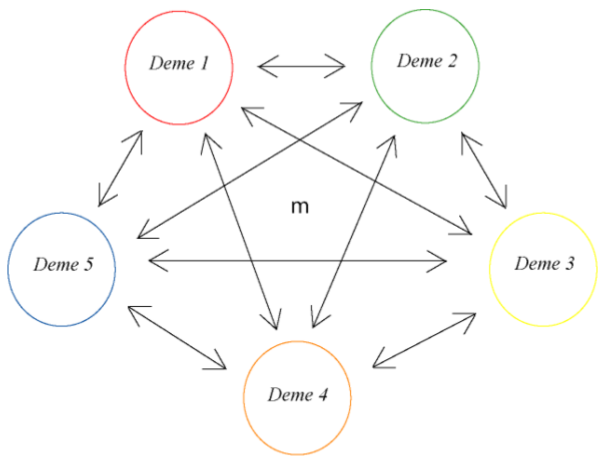


Figure 9. Island model. All demes have the same population size and the same migrant rate per population (deme).

4.3.2 Stepping-stone model

Under the stepping-stone migration events occur between neighboring populations (demes) (Figure 10).

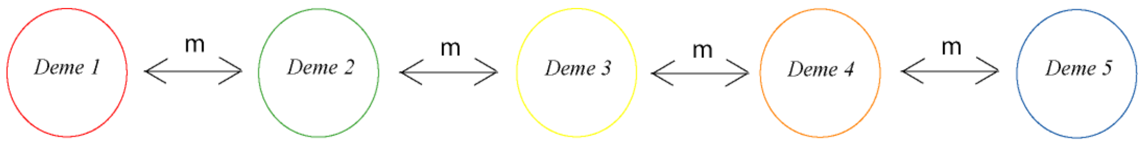


Figure 10. 1D stepping-stone model implemented in CoalEvol.

4.3.3 Continent-island model

The continent-island model implemented in *CoalEvol* allows symmetrical migration between a deme (continent) and the other demes (islands), see figure 11. Note that there is not possible to directly migrate individuals among islands.

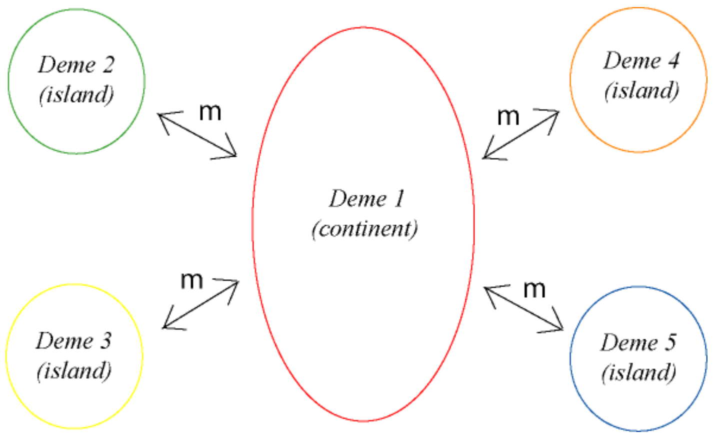


Figure 11. Continent-island model implemented in CoalEvol.

4.3.4 Temporal variation of migration rate

CoalEvol allows for temporal variation where a migration rate can be applied at different time periods (figure 12).

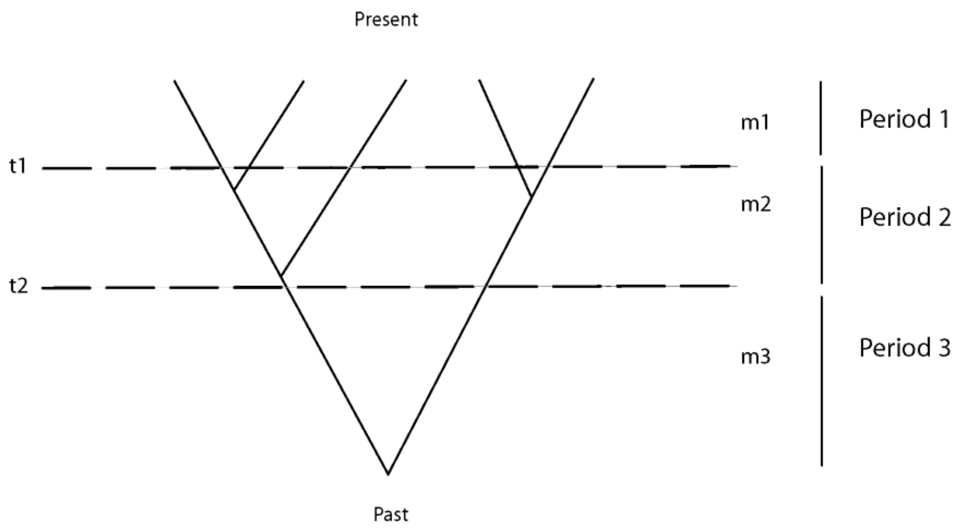


Figure 12. Temporal variation of the migration rate implemented in CoalEvol. The figure shows 3 temporal periods defined by two times (t1 and t2) and where each period implements a given migration rate (m1, m2 or m3).

4.4. Species/Populations tree

The evolution of species or populations can be user-defined. Two populations may converge at a given time in order to build a new population with the lineages of the previous populations. It always occurs under a migration model (although the migration rate can be set to 0). An example is shown in the figure 13. It is not possible use species/population trees in presence of tip dates if the time of the convergence of demes is younger than the time of the tip dates.

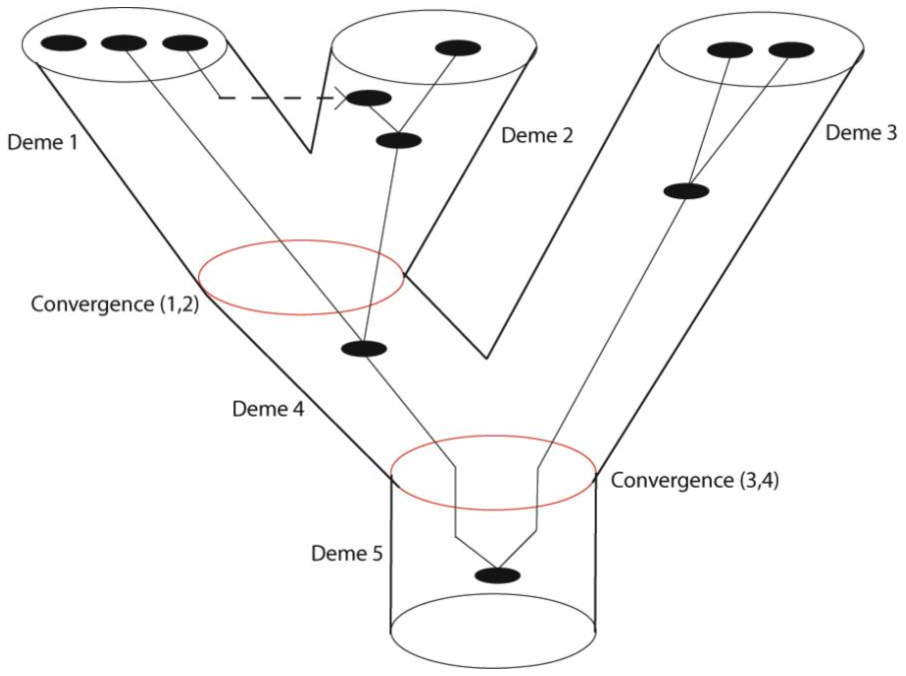


Figure 13. Ilustrative example of species/population tree. Two populations (demes) may coalesce into an ancestral population, these convergences are introduced by the user.

Importantly, the population size of the demes can be simulated through five different ways:

1. All demes can be simulated with a same and constant population size (introduced with the option -e, do not specify -g).
2. All demes can be simulated with a same and variable population size according to growth rate (introduced with the option -g0).
3. All demes can be simulated with a same and variable population size according to temporal periods (introduced with the option -g1).
4. Each deme can be simulated with a particular population size that is constant with time (introduced with the option -g2).
5. Each deme can be simulated with a particular population size that is variable with time (introduced with the option -g3) [e.g., 9].

The demographic terms are described in the following section.

4.5. Demography

CoalEvol allows for the specification of exponential population growth rate. The only modification concerns to the expected time to a coalescence, where *t* is the current time:

$$\textit{time to CA} \sim \frac{\log\left[e^{bt} + b\,Exp\left[\frac{k(k-1)}{2}\right]2N\right]}{b} - t$$

If the growth rate is negative, the coalescence time may be infinite (i.e., coalescence does not happen), and *CoalEvol* will stop and issue an error message. Alternatively, the user can define any number of demographic periods (Figure 14). β_i is the growth rate inferred for a demographic period *i* that goes from size N_{Bi} in the past to size N_{Ei} in I_i generations:

$$\beta_i = \frac{-\log\left(\frac{N_{Bi}}{N_{Ei}}\right)}{l_i}$$

The time to coalescence will be:

$$\text{Time to CA} \sim \frac{\log\left[Exp\left(\frac{k(k-1)}{2}\right)\beta_i 2N_{Ei}e^{-\beta(t-t_{i-1})}+1\right]}{\beta_i}$$

where *t* is the current time and t_i is the cumulative time from the present:

$$t_i = \sum_{j=0}^{j=i} l_j$$

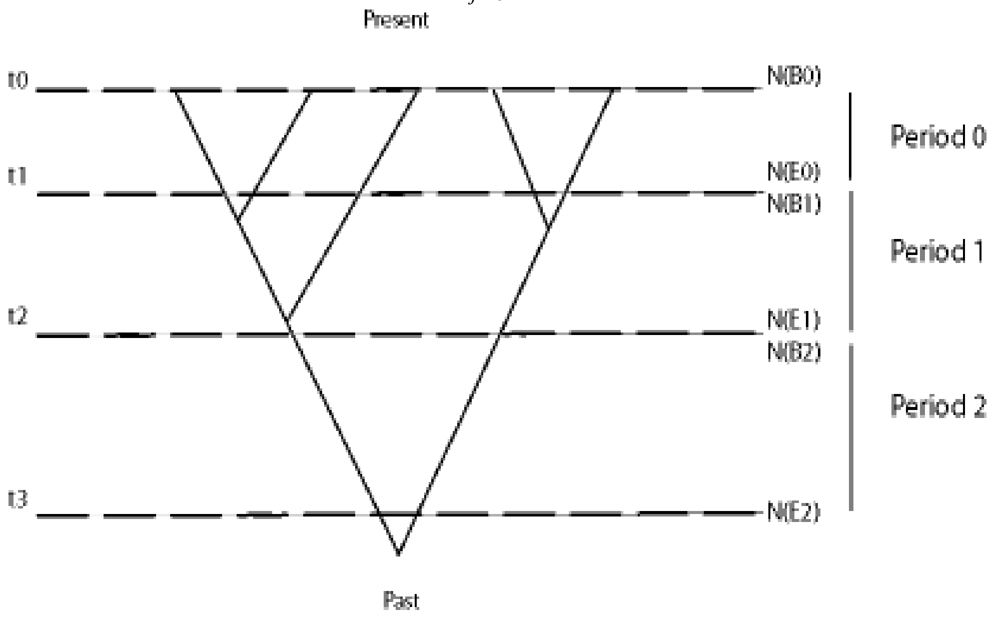


Figure 14. Demographic periods. The growth rate after the last period will the same as the one implied by the last period.

4.6. Tip Dates

Tip dates can be specified by a user to simulate sequences with different times, such as samples taken at different times. For this option, the user must introduce a generation time. Figure 15 shows examples of four samples taken at different times, the oldest sample contains only one sequence (seq00001), and the younger sample contains three sequences (seq00004, seq00006 and seq00005).

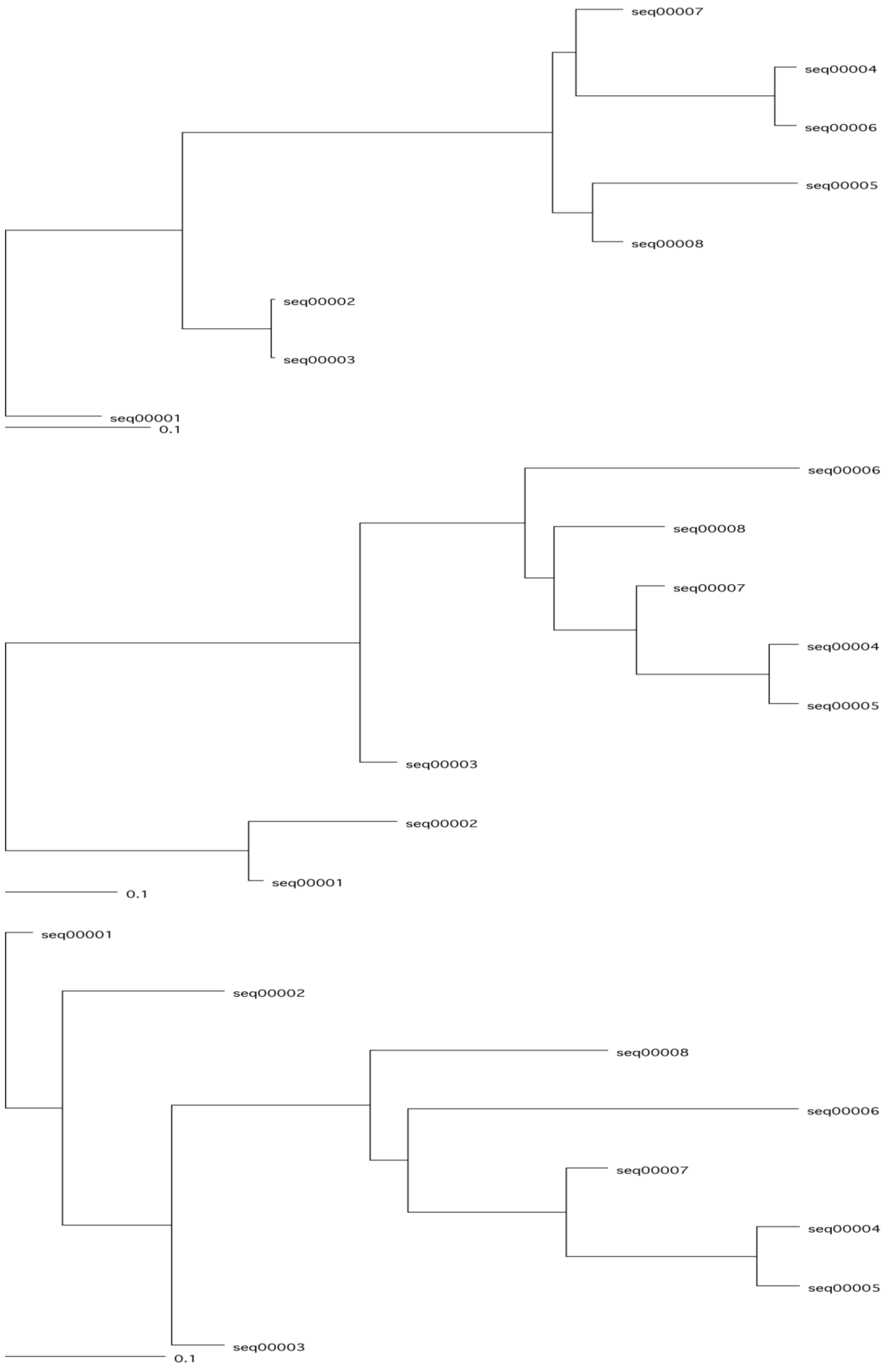


Figure 15. Tip dates simulation examples. Three examples of four samples taken at different times with 1, 2, 2 and 3 sequences per sample (from present to past).

4.7. Substitution models

Once the sample genealogy has been constructed, nucleotide or codon data can be simulated along its branches. *CoalEvol* implements multiple nucleotide, codon and amino acid models through the specification of different parameters (including base frequencies, relative substitution rates, transition/transversion ratio, nonsynonymous/synonymous rate ratio, a proportion of invariable sites and rate variation among sites) or by an implemented or user-defined empirical model. Conveniently, the sequence of the GMRCA can specified at random, according to the nucleotide or amino acid frequencies, or the user can directly specify a sequence.
Frequencies can be fixed for all sites or can vary across sites according to user-specified categories where each category has a probability to occur and its particular frequencies [37].
Importantly, variation in the substitution rate among sites can be user-specified by the shape of a gamma distribution (+G) and a proportion of invariable sites (+I).

4.7.1. Nucleotide models

CoalEvol implements the general time reversible model for nucleotide substitution (GTR) [13], a non reversible version (GTnR), and models nested therein, like JC [11], K80 [49], F81 [43] or HKY [12].

4.7.2. Codon models

CoalEvol implements the GY94 codon model [14], crossed with GTnR, GTR or any other nucleotide model, and with codon frequencies calculated from the nucleotide frequencies at each codon position (3×4): for example, GY94×GTR_3×4 or GY94×HKY_3×4. A key parameter of the codon models is the nonsynonymous/synonymous rate ratio (ω or *dN/dS)*, it reflects the type of selection acting at the molecular level:

$\omega > 1$: positive selection
$\omega = 1$: neutral evolution
$\omega < 1$: negative selection

The program assumes the universal genetic code without stop codons (i.e., a 61×61 matrix). Note that codon models imply much more computation time than nucleotide models.

-> *CoalEvol* implements this parameter in multiple different ways.

- Constant, M0: ω is constant across sites.
- Discrete user- categories: ω varies across sites according to some user-defined categories.
- M1 [16]: Two categories of ω value. In the first one ω is < 1, and in the second one ω is fixed = 1.

- M5 discrete gamma [16]: ω varies across sites according to a discrete gamma distribution.
- M5 continuous gamma [16]: ω varies across sites according to a continuous gamma distribution.
- M6 two continuous gamma [16]: ω varies across sites according to two continuous gamma distributions.
- M7 [16]: ω varies across sites according to a continuous beta distribution.
- M8 [16]: Two categories of ω value. At the first one ω varies across sites according to a continuous beta distribution, and at the second one the ω is introduced from the user and use to be = 1 or > 1.
- M9 [16]: Two categories of ω value. ω varies across sites according to a continuous beta and a continuous gamma distributions.
- M10 [16]: Two categories of ω value. ω varies across sites according to a continuous beta distribution and a continuous gamma distribution plus 1.

All these models can be crossed with amino acid constraints of any empirical amino acid model, according to the algorithm developed in [18]. Amino acid models available are: *Blosum62* [24], *CpRev* [25], *Dayhoff* [26], *DayhoffDCMUT* [27], *HIVb* [28], *HIVw* [28], *JTT* [29], *JonesDCMUT* [27], *LG* [30], *Mtart* [31], *Mtmam* [32], *Mtrev24* [33], *RtRev* [34], *VT* [35], *WAG* [36] and UserEAM (this last one is the name of the input file for a user-defined amino acid model).
In addition, all these models can be simulated with variable ω per branch.

-> *CoalEvol* allows the simulation of synonymous and nonsynonymous according to the MG94 codon model [19].

-> The HB codon model can also be simulated (see section 3.3) according to [21].

-> *CoalEvol* also implements three empirical codon models: ECMrest [22], ECMunrest [22] and ECMSchn2005 [23]. Indeed, it is possible to specify a user-defined empirical codon model by an input file (see section 3.3).

4.7.3. Amino acid models

CoalEvol implements a variety of empirical amino acid models: *Blosum62* [24], *CpRev* [25], *Dayhoff* [26], *DayhoffDCMUT* [27], *HIVb* [28], *HIVw* [28], *JTT* [29], *JonesDCMUT* [27], *LG* [30], *Mtart* [31], *Mtmam* [32], *Mtrev24* [33], *RtRev* [34], *VT* [35], *WAG* [36]. In addition, it is possible to specify a user-defined empirical amino acid model by an input file (see section 3.3).

5. Program benchmarking

CoalEvol has been widely validated and by several ways. The output of the program was contrasted with the theoretical expectations for the mean and variances for different parameter values, like the number of recombination and migration events or TMRCA, see [50]. Recombination hotspots were according to theoretical expectations, breakpoints were likely to occur in recombination hotspots regions [4]. Results obtained with *CoalEvol* and other programs [3] were compared under different scenarios.

Coalescent and user-specified tree/s where reconstructed by HYPHY [51] and PAUP* [52] from the simulated alignments. Parameters of substitution models were estimated from the simulated data using HYPHY [51], PAUP* [52] and other software. See diverse validations in [5, 40, 45, 53]. Amino acid models were verified using *ProtTest* [54]. Variable ω per branch was verified using the *GA-Branch* algorithm [55] implemented in the *datamonkey* web server [56].

6. History

CoalEvol allows:

- Simulation of nucleotide, codon and amino acid data under complex recombination, demographics and migration.

- Recombination with breakpoints at both intra and inter codon positions.

- Hotspots of recombination.

- Migration models: island model, stepping-stone model and island-continent model. Variable migration rate with temporal periods.

- Demographics according to temporal periods and growth rates.

- Tip Dates (temporal sampling).

- User-specified tree/s (instead of simulated coalescent trees).

- Nucleotide models: All substitution models (JC, ..., GTR).

- Codon models: GY94 (M0-M10) and user-probabilities for user-omegas; MG94. Crossed with any nucleotide substitution model.

Empirical codon models (ECM) (3), it also allows for user given models.

Codon models with physicochemical contributions of the encoded amino acids.

Variable omega per site and/or branch.

- Amino acid models (15), it also allows for user given models.

- Simulate variation of substitution rate among sites according to a gamma shape (+G) and proportion of invariable sites (+I).

- Print the simulated tree/s and the ancestral recombination graph (ARG).

- Print ancestral sequences: catMRCA (concatenate sequence from MRCAs of all recombinant fragments) and GMRCA.

- Output alignments by phylip, fasta and nexus formats.

Full history:

-> Version 2.0 (February 2008)

- Improve kappa Qij matrix for codon models.

- NetRecodon2.0.0. Recombination in codons.

- Option to print in a file the settings for the programmer (doSettingsFile).

- Option to print just the MRCA in separate files (doOutMRCAfiles).

- Option to print just the Branches of the Net in separate files (doBranchNetfiles). Bug solved in cMatrix.

- 2.0.8: Branches of Net files in a new format where a recombinant generates just a recombinant node.

- Version 2.0.9: Improved substitutions in codon models.

-> Version 2.2 (October 2008)

- For codon models in presence of recombination, NetRecodon print the GMRCA but no the concatenate MRCAs (sumMRCAs). Not for nucleotide models where

the printed MRCA is the concatenate MRCAs.

- NetRecodon prints both GMRCAs and the concatenate MRCAs for codon models. For nucleotide models only concatenate MRCAs.

-> Version 2.3 (October 2008)

- More codon models introduced: M1 (omega = 0 + omega = 1), M7 and three for M8: M8 (D.beta + omega > 1), M8a (D.beta + omega = 1), M8b (D.beta + omega < 1). Beta distribution have been developed by David Posada.

- Solved a small bug (now it prints the GMRCA file, free memory).

- Output file with the simulated omega per site, this is for M1, M7 and M8 codon models.

2.3.2 (October 2008)

- All codon heterogeneous models are with omega variable per codon, not per branch.

- Output file with the simulated omega per site for every heterogeneous codon model.

2.3.3 (October 2008)

- Option for haploid/diploid from the user.

2.3.4 (November 2008)

- Minor fixes.

- Print sequences for phylip output files with migration like: "s00001_p1..” and “outgrp_p0".

- Print output sequences also in fasta and nexus formats.

2.3.5 (November 2008)

- Improve M1 codon model, now P0 with omega <1 (not only necessary omega = 0) and P1 with omega = 1.

-> Version 2.4 (January 2009)

- Tip Dates. The dates of the tip lineages can be different. Thanks to David Posada for the code for this implementation!

In this version the tip dates only are working when there is not migration. e.g. 4 samples; 1995:seq1, 2003:seqs 4 and 6, and so on. Where the generation time is also introduced from the user.

2.4.1 (January 2009)

- Tip dates also in presence of migration.

In presence of convergence of demes, it cannot have tip dates with a time higher (older) that the time of the event of convergence of demes. Because at that time the deme of the tip date did not exist (it was converged).

- Minor fixes for printing codon models settings.

2.4.2 (March 2009)

- Minor fixes (improved memory).

3.0.0 (June 2009)

- Nucleotide evolution using the ARG, the network, such as for the codon

evolution. Now it prints both catMRCA and GMRCA as output files.

3.0.1 (July 2009)

- Breakpoints in trapped material are also considered, for both Gi and selected breakpoints.

5.0.5 (October 2009)

- Intracodon breakpoints taking into account pseudo-ancestral material (non ancestral material that can affect ancestral material in intracodon recombination).

Print sequence for the last node (GMRCA) and for the concatenated MRCA sequence for all material, for codon models (intracodon recombination).

5.0.6 (October 2009)

- Print sequence for the last node of the ancestral material (ancestral material GMRCA), for codon models (intracodon recombination).

-> Version 5.0.7 (September 2010 - February 2011)

- Print Network (only for the programmer). The particular case of, a recombination followed by a coalescent event of the two previous generated recombinant lineages, does not result in loop (it gives a linear lineage)!

- Fixed a bug in the counter of total non-synonymous and synonymous substitutions. Thanks to Suzanne English for her contribution!

5.0.8 (February 2011)

- Several memory improvements. Including for genome simulation.

5.0.9 (March 2011)

- Allowing for negative growth rates.

-> Version 6.0.0 (March 2011)

- Improvements in demography by periods: Nbegin Nend correction, timeCA when N is constant in the period, consideration of (cumDuration[period] - cumDuration[period-1]).

6.0.1 (August 2011)

- Fixed small bug in the migration scenario. Partial double variables improved.

-> Version 7.0.0 (October 2011)

- Change input file parameters. (-v) for R and NR matrices is the same input now.

- Implementation of amino acid substitution models:

Blosum62, CpRev, Dayhoff, DayhoffDCMUT, HIVb, HIVw, JTT, JonesDCMUT, LG, Mtart, Mtmam, Mtrev24, RtRev, VT, WAG

- An empirical amino acid substitution model can be introduced by the user (input file)

- Models were predicted using Prottest.

7.0.1 (November 2011)

- (-c) for several options of sequences output file.

- ARG can be printed to an output file (-). "Branches format", where networks can be directly visualized and analyzed using NetTest.*

7.0.2. (November 2011)
- Implementation of GY94 heterogeneous codon models: M6, M9 and M10.
- Codon model options (-m) reorganized with the new implementations.

7.0.3. (November 2011)
- Implementation ofMG94 codon model.
- Also MG94 codon models derived by crossing with nucleotide models (e.g., HKY, GTR, etc.) "Kosakovsky & Muse. Mol Biol Evol, 2005".

7.0.4. (December 2011)
- Implementation of empirical codon models: ECMrest ECMunrest (Kosiol et al, 2007) and ECMSchn2005 (Schneider et al, 2005). Thanks to Gina Cannarozzi for providing the codon matrix for the latter model!
- An empirical codon substitution model can be introduced from the user (input file).
- Update of the input file parameters.

7.0.5. (December 2011)
- Implementation of GY94 codon substitution models crossed with amino acid empirical models according to Wong WS, Sainudiin R, Nielsen R. BMC Bioinformatics. 2006, 7:148.

7.0.6. (December 2011)
- Improvements of output files: times and Networks.
- Implementation of GY94 codon models with different dN/dS per branch: GY94 M1-M10 codon models.
- Simulated dN/dS per branch was predicted using GA-Branch.

-> Version 7.1 (December 2011)
7.1.0. (December 2011)
- ti/tv, R and nR matrices is the same input setting now (-v).
- Fixed small bug in convergence of demes.
- Implementation of the Stepping-stone migration model.
- Print Usage function updated.

7.1.1. (January 2012)
- Implementation of the Island-continent migration model. A big deme (#1) is connected to small demes using a symmetrical migration rate. Small demes are not connected among them.

7.3.3. (January 2012)
- Temporal variation of migration rates.

-> Version 7.2 (February 2012)
7.2.0. (February 2012)
- Heterogeneous (hotspots) recombination. Thanks to David Posada, some functions where adapted from SNPsim (Wiuf and Posada, Genetics 2003).

7.2.1. (February 2012)

- Hotspots of recombination with and without migration.

7.2.2. (March 2012)

- Fixed a very small bug of hotspots of recombination with coding data.

- Update for reading parameters file, in particular for amino acid frequencies and demographic options.

-> Version 7.3 (April 2012)

7.3.0. (April 2012)

- Input tree/s from user. Thanks to David Posada, some functions where adapted from Mosaic (by D. Posada). Input trees were recovered from simulated alignments using PAUP and Hyphy.

7.3.1 (July-August 2012)

- A maximum number of characters for the name of taxas was fixed to 10.

- Implementation of the HB codon model (M.T. Holder et al., Phil. Trans. R. Soc. B (2008) 363, 4013-4021).

7.3.2 (August 2012)

- Variable frequencies at all levels (nucleotide, codon and amino acid) by user-specified categories. CAT models.

It also implies full HB codon model (M.T. Holder et al., Phil. Trans. R. Soc. B (2008) 363, 4013-4021) and CAT amino acid models (N. Lartillot and H. Philippe. Mol. Biol. Evol. (2004) 21, 1095-1109).

7.3.3 (August-December 2012)

- +G can be now applied per site, per region and per both site and region (before only per site). For the SGWE tool.

- Bigger input trees from input file (MAX_LINE).

- More robust generation of the random seed.

- HB codon model improvement, new cases included (equal frequencies).

- HB, minor bug solved in frequencies from command line.

7.3.4 (October 2013)

- Population/species tree with a particular (constant) population size per branch (deme)

7.3.5 (November 2013)

- Population/species tree with a variable population size per branch (deme)

7.3.5 (November 2019)

- Bug fixed in the implemented Muse & Gaut 1994 codon model.

7.4.0 (December 2024)

- Revised the printed recombination network (ARG)

*- Added additional output information about the simulated recombination events. This info is printed with the option of network file (*NetworkFile) in fasta format (c2 1 0)*

- Fixed some warnings.

7. Acknowledgments

This work was partially supported by grant BFU2004-02700 (MCyT) to DP and, by the the “Juan de la Cierva” fellowship JCI-2011-10452 to MA, from the Spanish Government. Several functions were taken from code provided by R. Nielsen and Z. Yang, other functions were provided from *SNPsim* [4] (by David Posada and Carsten

Wiuf) and *Mosaic* (by David Posada). We want to thank J. Carlos Mouriño at the Supercomputing Center of Galicia (CESGA) for extensive help with code parallelization. Thanks to Gina Cannarozzi for providing the matrix for an empirical codon model. Thanks to Darren Martin for comments on additional output information on the simulated recombination events. Thanks to all the users of *Recodon* and *NetRecodon* for reporting us bugs, comments and suggestions.

8. References

1. Kingman JFC: **The coalescent**. *Stochastic Processes and their Applications* 1982, **13**:235-248.
2. Hudson RR: **Properties of a neutral allele model with intragenic recombination**. *Theor Popul Biol* 1983, **23**:183-201.
3. Hudson RR: **Generating samples under a Wright-Fisher neutral model of genetic variation**. *Bioinformatics* 2002, **18**(2):337-338.
4. Wiuf C, Posada D: **A coalescent model of recombination hotspots**. *Genetics* 2003, **164**(1):407-417.
5. Arenas M, Posada D: **Coalescent simulation of intracodon recombination**. *Genetics* 2010, **184**(2):429-437.
6. Hudson RR: **Island models and the coalescent process**. *Mol Ecol* 1998, **7**:413-418.
7. Kimura M, Weiss GH: **The Stepping Stone Model of Population Structure and the Decrease of Genetic Correlation with Distance**. *Genetics* 1964, **49**(4):561-576.
8. Wright S: **Evolution in Mendelian populations**. *Genetics* 1931, **16**:97-159.
9. Heled J, Drummond AJ: **Bayesian inference of species trees from multilocus data**. *Mol Biol Evol* 2010, **27**(3):570-580.
10. Navascues M, Depaulis F, Emerson BC: **Combining contemporary and ancient DNA in population genetic and phylogeographical studies**. *Mol Ecol Resour* 2010, **10**(5):760-772.
11. Jukes TH, Cantor CR: **Evolution of protein molecules**. In: *Mammalian Protein Metabolism.* Edited by Munro HM. New York, NY: Academic Press; 1969: 21-132.
12. Hasegawa M, Kishino K, Yano T: **Dating the human-ape splitting by a molecular clock of mitochondrial DNA**. *J Mol Evol* 1985, **22**:160-174.
13. Tavaré S: **Some probabilistic and statistical problems in the analysis of DNA sequences**. In: *Some mathematical questions in biology - DNA sequence analysis.* Edited by Miura RM, vol. 17. Providence, RI: Amer. Math. Soc.; 1986: 57-86.
14. Goldman N, Yang Z: **A codon-based model of nucleotide substitution for protein-coding DNA sequences**. *Mol Biol Evol* 1994, **11**(5):725-736.
15. Anisimova M, Bielawski JP, Yang Z: **Accuracy and Power of the Likelihood Ratio Test in Detecting Adaptive Molecular Evolution**. *Mol Biol Evol* 2001, **18**(8):1585-1592.
16. Yang Z, Nielsen R, Goldman N, Pedersen A-MK: **Codon-substitution models for heterogeneous selection pressure at amino acid sites**. *Genetics* 2000, **155**:431-449.

17. Dutheil JY, Galtier N, Romiguier J, Douzery EJ, Ranwez V, Boussau B: **Efficient selection of branch-specific models of sequence evolution**. *Mol Biol Evol* 2012, **29**(7):1861-1874.
18. Wong WS, Sainudiin R, Nielsen R: **Identification of physicochemical selective pressure on protein encoding nucleotide sequences**. *BMC Bioinformatics* 2006, **7**:148.
19. Muse SV, Gaut BS: **A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome**. *Mol Biol Evol* 1994, **11**(5):715-724.
20. Pond SK, Muse SV: **Site-to-Site Variation of Synonymous Substitution Rates**. *Mol Biol Evol* 2005, **22**(12):2375-2385.
21. Holder MT, Zwickl DJ, Dessimoz C: **Evaluating the robustness of phylogenetic methods to among-site variability in substitution processes**. *Philos Trans R Soc Lond B Biol Sci* 2008, **363**(1512):4013-4021.
22. Kosiol C, Holmes I, Goldman N: **An empirical codon model for protein sequence evolution**. *Mol Biol Evol* 2007, **24**(7):1464-1479.
23. Schneider A, Cannarozzi GM, Gonnet GH: **Empirical codon substitution matrix**. *BMC Bioinformatics* 2005, **6**:134.
24. Henikoff S, Henikoff JG: **Amino acid substitution matrices from protein blocks**. *Proc Natl Acad Sci U S A* 1992, **89**(22):10915-10919.
25. Adachi J, Waddell PJ, Martin W, Hasegawa M: **Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA**. *J Mol Evol* 2000, **50**(4):348-358.
26. Dayhoff MO, Schwartz RM, Orcutt BC: **A model of evolutionary change in proteins**. In: *Atlas of protein sequence and structure.* Edited by Dayhoff MO, vol. 5, Suppl. 3. Washington D. C.; 1978: 345-352.
27. Kosiol C, Goldman N: **Different versions of the Dayhoff rate matrix**. *Mol Biol Evol* 2005, **22**(2):193-199.
28. Nickle DC, Heath L, Jensen MA, Gilbert PB, Mullins JI, Kosakovsky Pond SL: **HIV-specific probabilistic models of protein evolution**. *PLoS One* 2007, **2**(6):e503.
29. Jones DT, Taylor WR, Thornton JM: **The rapid generation of mutation data matrices from protein sequences**. *Comput Appl Biosci* 1992, **8**(3):275-282.
30. Le SQ, Gascuel O: **An improved general amino acid replacement matrix**. *Mol Biol Evol* 2008, **25**(7):1307-1320.
31. Abascal F, Posada D, Zardoya R: **MtArt: A New Model of Amino Acid Replacement for Arthropoda**. *Mol Biol Evol* 2007, **24**(1):1-5.
32. Yang Z, Nielsen R, Masami H: **Models of amino acid substitution and applications to mitochondrial protein evolution**. *Mol Biol Evol* 1998, **15**(12):1600-1611.
33. Adachi J, Hasegawa M: **MOLPHY version 2.3: programs for molecular phylogenetics based in maximum likelihood**. *Comput Sci Monogr* 1996, **28**:1-150.
34. Dimmic MW, Rest JS, Mindell DP, Goldstein RA: **rtREV: an amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny**. *J Mol Evol* 2002, **55**(1):65-73.
35. Muller T, Vingron M: **Modeling amino acid replacement**. *J Comput Biol* 2000, **7**(6):761-776.

36. Whelan S, Goldman N: **A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach**. *Mol Biol Evol* 2001, **18**(5):691-699.
37. Lartillot N, Philippe H: **A Bayesian mixture model for across-site heterogeneities in the amino-acid replacement process**. *Mol Biol Evol* 2004, **21**(6):1095-1109.
38. Griffiths RC, Marjoram P: **An ancestral recombination graph**. In: *Progress in population genetics and human evolution.* Edited by Donelly P, Tavaré S, vol. 87. Berlin: Springer-Verlag; 1997: 257-270.
39. Arenas M, Patricio M, Posada D, Valiente G: **Characterization of phylogenetic networks with NetTest**. *BMC Bioinformatics* 2010, **11**(1):268.
40. Arenas M, Posada D: **The effect of recombination on the reconstruction of ancestral sequences**. *Genetics* 2010, **184**(4):1133-1139.
41. Beaumont MA: **Approximate Bayesian Computation in Evolution and Ecology**. *Annu Rev Ecol Evol Syst* 2010, **41**:379-405.
42. Beaumont MA, Zhang W, Balding DJ: **Approximate Bayesian computation in population genetics**. *Genetics* 2002, **162**(4):2025-2035.
43. Felsenstein J: **Evolutionary trees from DNA sequences: A maximum likelihood approach**. *J Mol Evol* 1981, **17**:368-376.
44. Yang Z: **Among-site rate variation and its impact on phylogenetic analysis**. *Trends Ecol Evol* 1996, **11**(9):367-372.
45. Arenas M, Valiente G, Posada D: **Characterization of reticulate networks based on the coalescent with recombination**. *Mol Biol Evol* 2008, **25**(12):2517-2520.
46. Latter BD: **The island model of population differentiation: a general solution**. *Genetics* 1973, **73**(1):147-157.
47. Maruyama T: **A simple proof that certain quantities are independent of the geographical structure of population**. *Theor Popul Biol* 1974, **5**(2):148-154.
48. Matsen FA, Wakeley J: **Convergence to the island-model coalescent process in populations with restricted migration**. *Genetics* 2006, **172**(1):701-708.
49. Kimura M: **A simple method for estimating evolutionary rate of base substitutions through comparative studies of nucleotide sequences**. *J Mol Evol* 1980, **16**:111-120.
50. Hudson RR: **Gene genealogies and the coalescent process**. *Oxf Surv Evol Biol* 1990, **7**:1-44.
51. Kosakovsky Pond SL, Frost SD, Muse SV: **HYPHY: Hypothesis testing using phylogenies**. *Bioinformatics* 2005, **21**:676-679.
52. Swofford DL: **PAUP*: Phylogenetic Analysis Using Parsimony (*and Other Methods).** In., 4 edn. Sunderland, Massachusetts: Sinauer Associates; 2000.
53. Arenas M, Posada D: **Recodon: coalescent simulation of coding DNA sequences with recombination, migration and demography**. *BMC Bioinformatics* 2007, **8**:458.
54. Abascal F, Zardoya R, Posada D: **ProtTest: selection of best-fit models of protein evolution**. *Bioinformatics* 2005, **21**(9):2104-2105.
55. Pond SL, Frost SD: **A genetic algorithm approach to detecting lineage-specific variation in selection pressure**. *Mol Biol Evol* 2005, **22**(3):478-485.
56. Kosakovsky Pond SL, Frost SD: **Datamonkey: rapid detection of selective pressure on individual sites of codon alignments**. *Bioinformatics* 2005, **21**(10):2531-2533.