CoreSimul

User Manual

***CoreSimul*: A genome simulator for prokaryotes with homologous recombination and selection.**

Louis-Marie Bobay

Department of Biology, University of North Carolina Greensboro, 321 McIver Street, PO Box 26170, Greensboro, NC 27402, USA

Contact information
Louis-Marie Bobay
321 McIver Street
Greensboro, NC 27402, USA
Email address: ljbobay@uncg.edu
Phone: 336-256-2590

May 2020

Availability and implementation: *CoreSimul* can be freely downloaded at https://github.com/lbobay/CoreSimul. *CoreSimul* is written in Python 3.7 but is also compatible with Python 2.7. It requires the Python library NumPy. *CoreSimul* is compatible with Mac and Linux operating systems. No installation is required.

Usage:

```
python coresimul_master.py control.txt
```

Control file options:

Required parameters:

OUTPUT= output_directory #output folder
TREE= path_to_tree_file #tree file used for the simulations in nwk format

Optional parameters:

GC= x # GC content (default = 50%)
LENGTH=x # Genome length in base pairs (default = 10,000bp)
RESCALE= x # Tree branches rescaling coefficient (default = 1, no rescaling)
RHO=x # Recombination rate ρ (default = 0, no recombination)
DELTA=x # Average recombination tract length δ in base pairs (default = 100bp)
MIN_DELTA = # Minimal recombination tract length δ in base pairs (default = 1bp)
CODONS=x,x,x # Mutation rate at codon positions 1, 2 & 3 (default = 0.33,0.33,0.33, uniform rates)
SEQUENCE= path_to_sequence # Genome to use for the simulations in FASTA (if used options GC and LENGTH become obsolete. Default= “none”)

SUB_MODEL= model # Name of the substitution model to use: JC69 (Jukes and Cantor, default), K2P (Kimura 2 parameters), K3P (Kimura 3 parameters) or GTR (General Time Reversible).
SUB_RATE= x # Parameters for substitution model.
JC69: no parameters required
K2P: 1 parameter (kappa) (e.g. SUB_RATE = 1.5, same as KAPPA=1.5)
K3P: 3 parameters (a,b,c); a: transition rate; b: A<–>C and G<–>T; c: A<–>T and C<–>G.
GTR: 6 parameters (a,b,c,d,e,f)

a: A<–>G
b: A<–>C
c: A<–>T
d: G<–>C
e: G<–>T
f: C<–>T

KAPPA= x # Transition/transversion parameter kappa (κ) if using K2P model ($\kappa = 1$ if not specified). Can also be specified with SUB_RATE.

GAIN_RATE = x # this parameter allows to introduce gene gains in the concatenate. The rate of gene gains is defined relative to the substitution rate (if used, the rescaling coefficient of the substitution rate therefore impacts the rate of gene gains as well). The probability of gene gains is then proportional to branch length. The concatenate is assumed to be a succession of protein coding genes (999bp each). Genes can be gained between the 999bp segments (not within). The gained genes are all set at a length of 999bp with the same GC% as the rest of the simulated genome. Other parameters apply (substitution model, codon bias, recombination…). Note that gene gains are assumed to result from non-homologous recombination and are therefore completely independent from homologous recombination. Note, however, that events of homologous recombination can occasionally lead to gene gains if the recombined fragment encompasses an entire gene and that the recipient genome is missing the corresponding gene in the donor genome. Individual genes will be written in the **genes** folder.

LOSS_RATE = x # this parameter allows to introduce gene losses in the concatenate. The rate of gene losses is defined relative to the substitution rate (if used, the rescaling coefficient of the substitution rate therefore impacts the rate of gene losses as well). The probability of gene losses is then proportional to branch length. The concatenate is assumed to be a succession of protein coding genes (999bp each). Only entire genes can be lost, not fragments. As a result, each lost gene is necessarily a 999bp fragment. Gene losses are assumed to result from deletions and are completely independent from homologous recombination. Note, however, that events of homologous recombination can occasionally lead to gene deletions of the recombined fragment encompasses an entire gene and that the donor genome is missing the corresponding gene in the recipient genome. Individual genes will be written in the **genes** folder.

EXP_COEFF = x # The probability of homologous recombination depends on the levels of sequence divergence. By default, this parameter is not used, and any pair of sequences has the same probability to recombine, regardless of their sequence identity. If used, the probability of recombination depends on the sequence divergence such as: $p = 10^{-\pi\Phi}$, with p the probability to recombine, π the sequence divergence and Φ the slope of the relationship between sequence divergence and sexual isolation. The value of Φ is species-specific and has been determined experimentally for several species. If EXP_COEFF is set to "yes", Φ will be set to 18.1 by default, as determined experimentally for *Streptococcus pneumoniae* and which was found to present an intermediate value of Φ compared to other species (Majewski et al. 2000). Note that this option can be time-consuming and that we do not recommend using it when sequences are very divergent (e.g. 20% divergent).

Example of control files:

```
# Example 1
OUTPUT= results_Acinetobacter_pittii
TREE=Acinetobacter_pittii.tree
GC=45
LENGTH=100000
RESCALE= 0.80
RHO=0.5
DELTA=  100
CODONS=0.15,0.07,0.78
SEQUENCE=none
SUB_MODEL=K2P
KAPPA=1.6
```

```
# Example 2
OUTPUT= results_Acinetobacter_pittii
TREE=Acinetobacter_pittii.tree
RESCALE= 0.80
RHO=0.5
DELTA=  100
CODONS=0.15,0.07,0.78
SEQUENCE=input.fa
SUB_MODEL=GTR
SUB_RATE=0.30,0.1,0.20,0.15,0.1,0.25
```

Description

CoreSimul is a forward-in-time simulator that generates a set of bacterial genomes based on a phylogenetic tree. As suggested by its name, *CoreSimul* aims at simulating the core genome—the set of genes conserved across all genomes—of a population or a species. The *CoreSimul* process starts by generating a random core genome sequence of length L and with a GC-content GC specified by the user. An input sequence can also be provided by the user. The sequence is assumed to represent a concatenate of protein coding genes without intergenic DNA. This sequence is then evolved *in silico* following a branching process that mimics the input tree. The rate of substitutions *m* is based on the branch length of the input tree, and this rate can be modified by the user with a rescaling coefficient. Although the overall rate of substitution is imposed by the input tree, the sequence can evolve at different rates across codon positions and the relative rates of the three codon positions can be specified by the user. In addition, different substitution models can be modeled (JC69, K2P, K3P and GTR). Finally, the genomes are evolved with a recombination rate ρ, which is defined relative to the substitution rate *m*. Recombination events are internal to the simulated dataset and no imports from external sources are modelled.

In order to mimic more realistic conditions, the different sequences present at any given time are evolved simultaneously and only sequences overlapping in time are allowed to recombine with one another (i.e. recombination with ancestral sequences is not allowed). Concretely, the phylogenetic tree is divided in multiple "time segments" of overlapping branches. For each time segment *t*, each sequence receives a number of mutations M_t and a number of recombination events R_t defined by a Poisson process of mean $m_t.l$ and $\rho_t.l$, respectively, with *l* the length of the branch in the time segment, m_t the mutation rate and ρ_t the recombination rate. The mutation and recombination events are then introduced in a random order in the different sequences of the time segment: a random sequence of the time segment is pulled and a mutation event or a recombination event is introduced randomly (this step is repeated until all the mutation and recombination events specific to each sequence have been introduced). The donor sequence of each recombination event is pulled randomly from the set of sequences in the time segment. The position of each recombination event is chosen randomly along the sequence and its size it defined by a geometric distribution of mean δ specified by the user (genomes are assumed to be linear).

Note that if gene gains and losses are simulated, it will introduce indels in the simulated genome. Homologous recombination only occurs if both extremities of the recombined sequence are present in both the donor and the recipient genomes. In the case a recombination event occurs in a region were only one extremity of the recombined sequence is present if the donor and recipient genome, only the portion of the sequence shared by both genomes is exchanged.

During the simulation the number of polymorphisms (SNPs) exchanged by each recombination event is recorded to generate the statistic η, which represents the average number of polymorphic alleles exchanged by recombination. Using this statistic, the effective recombination rate *r/m* is defined from the relationship $r/m = \rho.\eta.\delta$. The effective recombination rate *r/m* is frequently used to measure recombination rates and represents the number of polymorphisms exchanged by recombination relative to the number of polymorphisms introduced by mutation.

Simulating gene gains and losses

Gene gains and losses are simulated as the result of non-homologous horizontal gene transfers from a source external to the dataset and are therefore independent from homologous recombination events. The simulated genome is assumed to be a string of protein-coding genes each being 999bp long with no intergenic region. Entire segments of 999bp are gained or lost and new genes are randomly introduced between genes in the concatenate (never within). As a consequence of gene gains and losses, the concatenate will contain indels (each indel should correspond to one or more genes, each 999bp long). Genes that are gained or lost can engage in homologous recombination if homologous sequences exist in the donor and the recipient genome.

Output files

All output files are generated in the output folder specified with the OUTPUT= option. The simulated sequences are written in the file **genomes.fa**. The file **detail.txt** includes basic information and paramerers about the simulation and the file *rm.txt* contains the effective rate of recombination *r/m* defined above. In the case where gene gains and/or gene losses are simulated *CoreSimul* will also return the file **core.fa**, which contains the concatenate of the core genes only (i.e. the genes present in all the genomes) and all individual gene alignments will be written in a folder named **genes**. Other intermediary files used by *CoreSimul* are generated in this folder.

Notes and recommendations for users

CoreSimul requires a phylogenetic tree and the topology and the branch lengths of the tree are used to simulate the genome dataset. Note that phylogenetic algorithms do not model homologous recombination and that any homoplasy in the alignment that is the result of recombination will be inferred as multiple independent mutation events when constructing the phylogeny. As a result, the tree used to run the simulation will yield a core genome alignment with higher levels of polymorphisms than the real alignment. The rescaling coefficient parameter can be used to rescale the branch lengths of the tree in order to obtain simulated core genomes with levels of polymorphisms that more closely reflect the real dataset.

Recombination rates are usually expressed as the effective recombination rate *r/m* which represents the number of alleles exchanged by recombination relative to the number of alleles introduced by mutations (see above). The number of alleles exchanged by recombination depends on the absolute recombination rate ρ and the average length of the recombining fragments δ. Values of δ are thought to range typically from 50 to 1000bp in bacteria but larger recombination events have been occasionally reported. It is more complex to anticipate what would constitute realistic values of ρ but we recommend running trials with different recombination rates ρ and to refer to the corresponding *r/m* values calculated in the output. Values of *r/m* typically range from 0 to 5 but higher values have been reported.

Relative substitution rates across codon positions allow to mimic the effect of purifying selection acting on protein coding genes. These relative substitution rates depend on the strength of selective pressures acting on the species or population and ideally these rates should be determined empirically based on the relative levels of polymorphism. Typically, we recommend 0.20, 0.10, 0.70 for codon positions one, two and three, respectively.