# Parameters for simulations using `retrocombinator`

Anindya Sharma

2021-04-20

TODO:

- Give a print message that describes the paramaters that have been run in an intuitive fashion
- For each object, have a print function that can do this
- By default, print as a message, but give users the option to dump to a log file
- Search parameter space and label parameters with what scenarios they are likely to lead to

## Parameters and their defaults

- `SequenceParams` represents what set of retrotransposons the simulation starts off with. It starts of multiple identical copies of the same sequence, where that sequence is sampled from a uniform A/G/C/T distribution. It is constructed by either specifying a sequence length or by specifying a file to a sequence:
  - `num_initial_copies : numeric` Number of initial retrotransposons **(default = 100)**
  - `seq_length : numeric` Sequence length (number of nucleotides) of the retrotransposons **(default = 5000)**
  - `seq_filename : character` Path to initial sequence
- `SimulationParams` represents how long the simulation will run for, and at what timescale. It comprises of the following:
  - `num_jumps : numeric` The number of steps (jumps) in our simulation **(default = 20)**
  - `timestep : numeric` How much real time does one step (jump) in our simulation measure, in millions of years **(default = 1)**
  - `max_active_copies : numeric` What is the largest population size of active sequences to keep track of? **(default = 500)**
- `MutationParams` represents how what nucleotide subtstition model will be used to modify the sequences during the simulation. It is constructed from one of the following `character` literals to the argument `model`:
  - `"JC69"` Jules and Cantor - 1969
  - `"K80"` Kimura - 1980 **(default)**
  - `"F81"` Felsenstein - 1981
  - `"HKY85"` Hasegawa, Kishino and Yano - 1985
  - `"TN93"` Timura and Nei - 1993
  - `"GTR"` General Time Reversible Model, Tavaré - 1986
- `FlagParams` represents the parameters used during a simulation in which we keep track of a retrotransposon's 'active' status. A retrotransposon that is active is capable of transposition or bursting, but potentially loses its active status if a point mutation affects a critical part of the sequence.
  - `length_critical_region: numeric` Sequence length (number of nucleotides) of the critical region of a retrotransposon **(default = 10)**
  - `prob_inactive_when_mutated : numeric` The probability that a point mutation to the critical region causes a sequence to become inactive **(default = 0.001)**
  - `max_inactive_copies : numeric` What is the largest population size of active sequences to keep track of? **(default = 500)**
- `BurstParams` represents how an active transposon will burst during transposition. It comprises of the following:

  - `burst_probability : numeric` The probability that an active retrotransposon will increase in copy number during a time jump of one timestep **(default = 0.1)**
  - `burst_mean : numeric` The Poisson mean for the distribution that specifies how many new sequences an active sequence will create during bursting **(default = 1)**
  - `recomb_mean : numeric` The expected number of template switches during recombination between two sequences (chosen from a Poisson distribution with this as its mean) **(default = 1.5)**
  - `recomb_similarity : numeric` How similar does an active sequence have to be with another sequence for them to be allowed to be recombine during transposition? **(default = 0.85)**
- `SpeciationParams` represents how we keep track of species during the simulation. It comprises of the following:
  - `selection_threshold : numeric` What sequence similarity to the original sequence do we wish to maintain? Sequences that diverge beyond this similarity level are dropped over the course of simulation **(default = 0.5)**
  - `species_similarity : numeric` What sequence similarity do two sequences have to be to each other for them to be considered to be of the same species? **(default = 0.7)**
  - `species_coherence : numeric` What proportion of the overall sequence similarity matrix of a species needs to score be above `species_similarity` before we decide the species has split into two species? **(default = 0.5)**
- `OutputParams` represents how and where the output of the simulation will be saved. It comprises of the following:
  - `file_out : character` Where should the simulation be saved? **(default = 'file.out')**
  - `num_out_seqs : numeric` How many times during the event of the simulation should we output the raw sequences (as nucleotide strings) themselves? **(default = 2)**
  - `num_out_init : numeric` How many times during the event of the simulation should we output the distance of sequences to the initial sequence? **(default = 10)**
  - `num_out_pair : numeric` How many times during the event of the simulation should we output pairwise distances between all pairs of sequences? **(default = 10)**
  - `num_out_species : numeric` How many times during the event of the simulation should we output the species tags of each of the sequences? **(default = 10)**
- `SeedParams` represents how to select the seed for randomisation for the simulation. It comprises of the following:
  - `to_randomise : logical` Should this simulation be run with a random seed to begin with? (The seed is based on system time) **(default = false)**
  - `to_seed : logical` Should this simulation be run with a specified seed to begin with? (This overwrites `to_randomise`) **(default = true)**
  - `seed : numeric` If `to_seed` is `TRUE`, what should the initial seed for the random number generator be? **(default = 0)**

## How to use

To run a simulation, just call the main function `simulate()` and it will run with the default parameters.

```
simulate()
```

To overwrite any parameters, first create objects to represent the parameters you wish to overwrite. Say for example you want to change the recombination mean, and the flagging parameters, run the following code. Whenever a parameter is not specified explicitly, the default will be used.

```
burst_params <- BurstParams(recomb_mean = 1.0)
flag_params <- FlagParameters(length_critical_region = 0,
                              prob_inactivity_when_mutated = 0)
simulate(BurstParams = burst_params, FlagParams = flag_params)
```