CellCoal manual v1.1

David Posada

January, 2020

Contents

1. About CellCoal

CellCoal simulates the somatic evolution of single-cells including genotypes resulting from single-cell DNA sequencing. CellCoal generates a coalescent tree and can simulate a sample of diploid genomes from somatic cells –no recombination– from a growing population, together with another genome as outgroup. A typical example would be a sample of single-cell genotypes from a growing tumor population together with a healthy cell used as outgroup. CellCoal implements multiple mutation models for single-nucleotide variants (0/1, DNA, infinite and finite site models, point deletions, copy-neutral LOH, cancer mutational signatures) and is able to generate a .VCF file with NGS read counts and genotype likelihoods considering technical artifacts like allelic imbalance, allelic dropout, sequencing error, amplification error, plus doublet cells.

2. Citation

If you use CellCoal, please cite it as:

Posada D. 2020. CellCoal: coalescent simulation of single-cell genomes. https://github.com/dapogon/cellcoal.

3. Getting started

1. **Download**: get the program from the GitHub repository at https://github.com/dapogon/cellcoal. Under the Code tab you will see a section call *release*. Click on it download the source code in .zip (or tar.gz) format. Then unzip the *cellcoal-x.y.z.zip* file (for example, cellcoal-1.1.0.zip). You should see now a folder called *cellcoal-x.y.z*. Move into the folder to compile and run the program.
2. **Compile**: type `make`. The program should compile without much problem in Linux/MacOSX. A *Makefile* is provided for the gcc compiler. The executable file will be located in the *bin* folder.
3. **Run**: `bin/cellcoal-x.y.z`. In this case, the program arguments will be read from a file called *parameters*. Alternatively, arguments can be entered directly in the command line, or from a specific parameter file with a different name.
4. **Inspect results**: Once the run is finished, different output files will be written by default inside a folder called *results*.

4. Basic usage

CellCoal does not have a Graphical User Interface (GUI). It works on the Linux/Mac command line in a non-interactive fashion. It can run in two main ways parsing its arguments from a parameter file, or parsing its arguments directly from the command line.

If no arguments are passed in the command line, CellCoal will look for a file called *parameters* in the current working directory:

```
bin/cellcoal-x.y.z
```

The *parameters* file is a text file that contains the different arguments for the program. The arguments correspond to different letters whose case (uppercase or lowercase) makes a difference. Comments within brackets are allowed. To avoid unexpected problems, users are encouraged to maintain the same order of the arguments as in the example files. For clarity, the example parameters files are divided in different blocks or sections including related arguments.

Alternatively, users can specify their own parameter file **(option -F)**, for example:

```
bin/cellcoal-x.y.z -Fexamples/parameters.ISM.binary
```

Otherwise, CellCoal will run under the arguments specified in the command line, using default values for the absent arguments. For example, one could type:

```
bin/cellcoal-x.y.z -n100 -s20 -l1000 -e10000 -g1.0e-04 -k1 -i1 -b1 -j250 -p0.0 -f0.3 0.2
0.2 0.3 -r0.00 0.03 0.12 0.04 0.11 0.00 0.02 0.68 0.68 0.02 0.00 0.11 0.04 0.12 0.03 0.00
-1 -2 -3 -4 -6 -9 -v -x -#200011
```

where *-n* is the number of replicates, *-s* is the number of sampled cells, *-l* is number of sites, *-e* is the effective population size, *-g* is the growth rate, *-k* is the root branch length ratio, *-i* is the rate variation among branches, *-b* specifies the alphabet (DNA), *-j* is a fixed number of mutations, *-p* is the proportion of alternative model sites, *-f* are base frequencies, *-r* is the mutation matrix, *-1* to *-x* control output options, and *-#* is the seed for random number generation.

A brief usage guide plus the current default values for the different parameters can be obtained with the argument -?:

```
bin/cellcoal-x.y.z -?
```

5. CellCoal implementation

The main steps of CellCoal are:

1. Coalescent simulation of the genealogy of a sample of cells (the user can also specify a tree with branch lengths).
2. Addition of an outgroup and optionally, modification of the branch lengths.
3. Simulation of a reference and an ancestral genome, with or without germline variation.
4. Addition of mutations, LOH and deletions along the tree in order to obtain the true genotypes.
5. Addition of technical errors without generating read counts (ADO + genotype errors) OR Addition of technical errors while generating read counts (ADO + amplification error + sequencing error.
6. Estimation of genotype likelihoods under various models
7. Print genotypes/haplotypes to files.
8. Print summary statistics.

5.1 Coalescent genealogy

CellCoal simulates a cell genealogy under the standard neutral coalescent, going backward in time (Fig. 2 left). Note that the coalescent assumes the cell sample comes from a much larger population, which in this case can be constant or growing. In the coalescent we use the *effective population size (N)*, a well-known quantity in population genetics that refers to the number of individuals in an ideal population.

In CellCoal, time is scaled in units of N generations

CellCoal can consider continuous exponential population growth (Fig. 2 middle) (Slatkin and Hudson, 1991) (SH91) and multiple demographic periods (Fig. 2 right). For the latter, 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 Hudson (2002).

CellCoal also implements the coalescent parameterization of Ohtsuki and Innan (2017) (OI17) for cancer cell populations. The only difference with the SH91 model is a factor of $2b$, where b is the birth rate, due the consideration of overlapping generations, and the use of absolute times (expected time until a birth event). Note that if we believe that the cancer cell population starts from a single ancestral cell then we do not get an exponential growth model exactly, so both OI17 and SH91 are approximations.

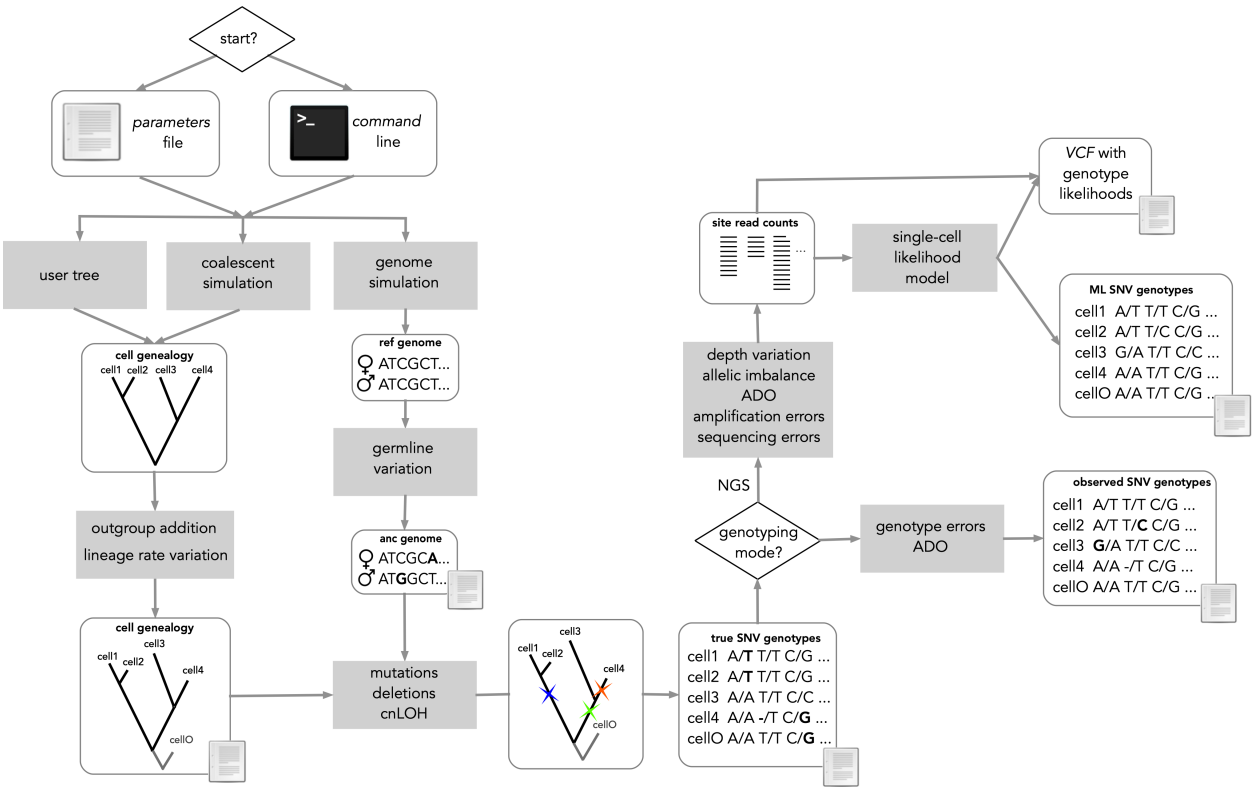

Figure 1: Fig 1. Main flow of CellCoal.

CellCoal implements two parameterizations of the coalescent with exponential growth, the first proposed the Slatkin and Hudson (1991) model, and the second developed by Ohtsuki and Innan (2017) >(OI17) specifically for cancer cell populations.

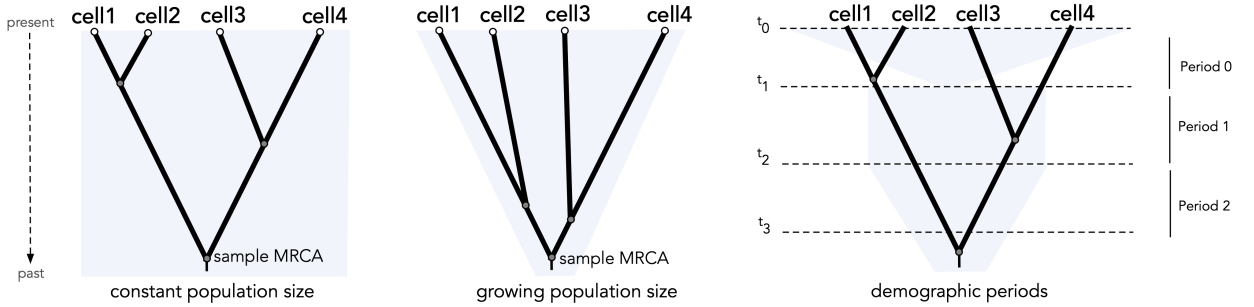

Figure 2: Fig. 2. A random coalescent genealogy for a sample of 4 cells under constant (left), and growing population size (middle) and demographic periods (right). The effective population size is indicated as blue background. In the right, from Period 0 to Period 1 the population size experiments a severe bottleneck.

5.1.1 Adding an outgroup

After the sample genealogy is simulated, two additional branches are added: a **root branch** and an **outgroup branch** (Fig. 3), in order to include an outgroup cell in the sample. In phylogenetics an outgroup is a distant taxon that serves as a reference and is often used for rooting purposes. The length of these two branches is controlled by the user (Fig. 3).

In a tumor scenario, the outgroup cell will be naturally a healthy cell (Fig. 4)

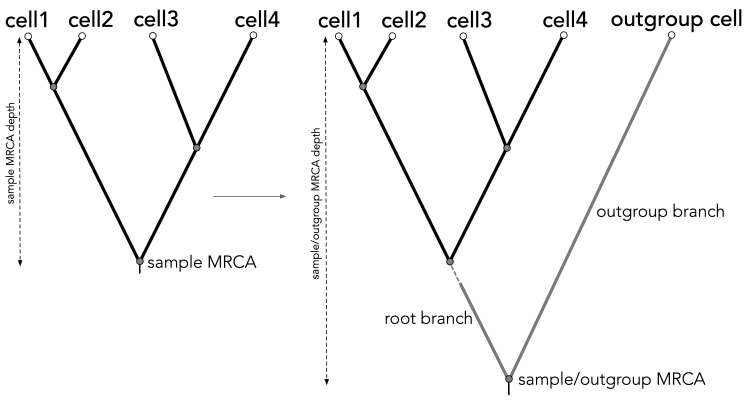

Figure 3: Fig. 3. Outgroup addition. The length of the root branch will be controlled by the user with the parameter -k, which specifies the length of this branch as a proportion of the sample MRCA mean depth. The length of the outgroup branch depends on the parameter -q, which is the length of this branch as a proportion of the sample/outgroup most recent common ancestor (MRCA) mean depth. In this example, the corresponding values were -p0.5 -k1.0.

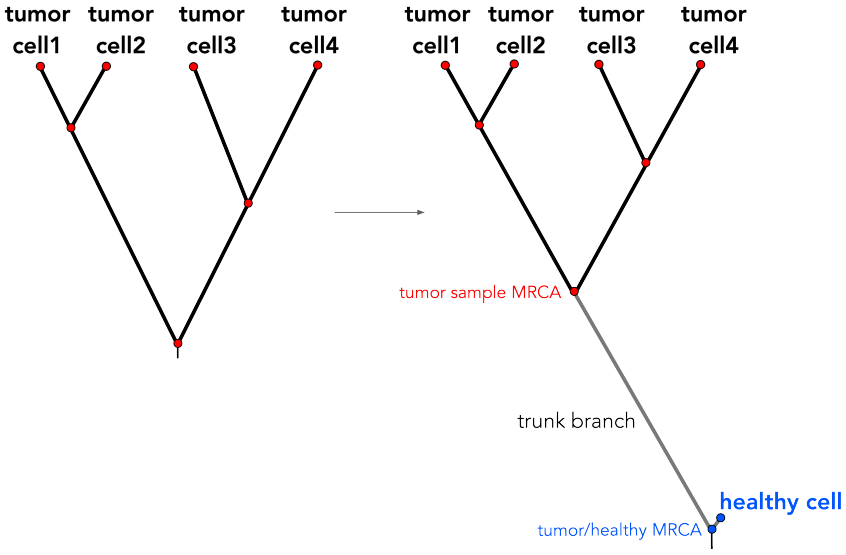

Figure 4: Fig. 4. Addition of a healthy cell in a tumor scenario. In this case we expect that the healthy cell has evolved much slower than the tumoral cells, so a small (or zero) outgroup might make more sense. In this example, the corresponding values where -p1 -k0.05.

5.1.2 Rate variation among lineages

The standard coalescent results in an ultrametric tree, where all the tips are at the same distance from the most recent common ancestor (MRCA) of the sample. These branches can be modified by introducing rate variation among linages/branches, here using multipliers sampled from a Gamma distribution (see Fig. 5). This approach was introduced by Yang (for an easy review see Yang 1996) to add substitution rate variation among different nucleotide sites in a DNA sequence. Biologically this is interesting if we want to simulate for example a situation under which some lineages evolve at different rates.

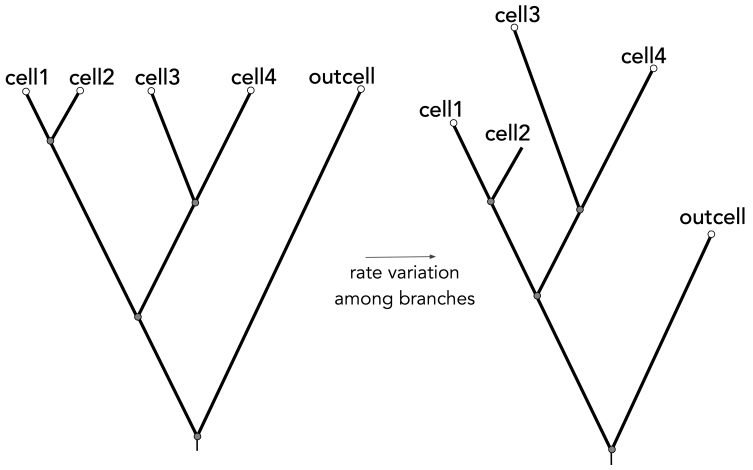

Figure 5: Fig. 5. Rate variation among branches. Rate variation among lineages can be a product for example of changes in the mutation rate.

5.2 Genotype evolution models

CellCoal simulates the evolution of cell genotypes along the genealogy by adding somatic mutations (single-nucleotide variants -SNVs-, copy-neutral loss of heterozygosity events -cnLOH-, and point deletions), starting from an ancestral genome at the *sample/outgroup MRCA*. CellCoal considers two possible **alphabets (0/1 or DNA; option -b)** and several single-nucleotide mutation models (infinite and finite sites models, including trinucleotide mutational signatures), which can be mixed along the genome.

The default SNV model is a diploid infinite site model (ISM).

Note that CellCoal does not simulate copy number alterations or structural variants.

5.2.1 Generation of reference and ancestral genomes

The first step to generate the cell genotypes is the generation of a **reference genome**. Under the binary alphabet, it will consist of all 0s. For DNA, the reference genome will be simulated at random according to the specified nucleotide frequencies, unless genetic signatures are specified, in which case the reference genome will be simulated according to the trinucleotide frequencies in the human genome (version GRCh37). Alternatively, for DNA data, the user can specify its own reference genome in a FASTA file.

The **ancestral genome** at the sample/outgroup MRCA of the genealogy will be simulated by introducing **germline variation (option -c)** (single nucleotide polymorphisms or SNPs) into the reference genome at a rate specified by the user. Logically, if this rate is 0, the reference and the ancestral genomes will be identical. Once the ancestral genome is in place, mutations are introduced according to the specified mutation model.

The reference genome can be simulated according to the trinucleotide frequencies in the human genome (version GRCh37); germline SNPS can be added afterwards.

5.2.2 Infinite-site SNV models

CellCoal implements several types of **single-nucleotide mutation models (option -m)**. Infinite-sites models (ISM) allow a given site to change only once. However, in a diploid scenario we can implement this model in at least two ways. Under the **diploid ISM**, a given site can change only once in either the maternal and paternal genomes. That is, the change will be for example either 0/0 -> 1/0 or 0/0 -> 0/1 but not both (where 0 is the ancestral allele and 1 the derived one). In consequence, under the diploid ISM, homozygous mutants (1/1) are not possible. A different implementation of the ISM, called here **haploid ISM**, considers instead maternal and paternal sites as different locations, so it allows for two somatic mutations to take place at the same site if they occur in the maternal and paternal genome. Thus, under this model 1/1 genotypes are possible. In both ISMs, the number of somatic mutations is distributed as a *Poisson* according to the total tree length (which is a function of the mutation rate and time), unless a **fixed number of mutations (option -j)** is requested. For the haploid ISM the expected number of mutations can be easily calculated:

$$E\,(\text{number of mutations}) = \Theta \sum_{i=1}^{i=n-1} \frac{1}{i}$$

where $\Theta = 4N\mu 2\text{L}$, *being* N* the **effective population size (option –e)**, μ the **mutation rate per site per generation (option -u)**, and L the **number of sites (option -l).** Alternatively, it is also possible to specify a **fixed number of mutations (option -j)** to take place along the genealogy.

The specific sites where the ISM mutations occur are selected uniformly along the genome, unless mutational signatures are requested (see below). The particular branches of the cell genealogy where mutations are located are selected according to their lengths. That is, longer branches will receive more mutations on average.

5.2.3 Infinite-site mutational signatures

For DNA data, mutational signatures (sensu Alexandrov *et al.* 2013) can be simulated under a **diploid infinite trinucleotide model (ISTM) (option -S)**, that considers the nucleotide context of the mutated site (bases immediately 5' and 3'). Currently, 30 distinct signatures can be incorporated (https://cancer.sanger.ac.uk/cosmic/signatures) and mixed in different proportions. As in ISMs, the number of somatic mutations is distributed as a *Poisson* according to the total tree length

Importantly, for computational efficiency, in the current implementation we assume that the genomic context is constant along the tree. That is, we select a site based on the context in the ancestral genome, before locating it in a given tree branch. Thus, this assumption will be broken in the case of contiguous mutations.

> IMPORTANT: In order to get the expected counts under each signature, one should always simulate long genomes, otherwise rare trinucleotide will not be available and, depending on the signature, could significantly bias the observed counts.

5.2.4 Infinite-site copy-neutral LOH

Copy-neutral loss of heterozygosity (cnLOH) (option -H) events (e.g., A/G -> A/A, or A/G -> G/G) can be added assuming a haploid ISM, which means that cnLOH cannot happen in the same site twice unless they occur in a different maternal/paternal genome. As for SNVs, the number of cnLOH events will be distributed as a *Poisson* according to the total cnLOH tree length (which in this case is a function of the cnLOH rate and time). The specific sites where the deletions occur are selected uniformly along the genome. The particular branches of the cell genealogy where deletions are located are selected according to their (cnLOH) lengths.

5.2.5 Infinite-site single-nucleotide deletions

Single-nucleotide deletions (option -d) (e.g., N/N -> –/N, or N/N -> N/–) can be added also assuming a haploid ISM. As for SNVs, the number of point deletions will be distributed as a *Poisson* according to

the total deletion tree length (which in this case is a function of the deletion rate and time). The specific sites where the deletions occur are selected uniformly along the genome. The particular branches of the cell genealogy where deletions are located are selected according to their (deletion) lengths.

5.2.6 Finite-site SNV models

CellCoal also implement several **finite-sites mutation models (option -m)** (FSM) for SNVs. In this case, multiple mutations can happen along a branch. Typically, these models assume a continuous Markov process and are very typical in phylogenetics, where they are often referred as substitution models (https://en.wikipedia.org/wiki/Models_of_DNA_evolution).

For 0/1 data, the model implemented is known as Cavender-Farris-Neyman or Mk2 model (see Lewis 2001), and is equivalent to a Jukes-Cantor (1969) model for two alleles. For DNA data, reversible (assume that the rate X->Y is the same as Y-<X) and non-reversible models, can be built by specifying the **nucleotide frequencies (option -f)** (at equilibrium; there are not expected to change along the genealogy a **transition/transversion rate ratio (option -t)** or a full 4x4 **instantaneous mutation rate matrix (option -r)**, plus **rate variation among sites (option -a)** (using a Gamma distribution; see Yang 1996). One can define in this way popular models like Jukes-Cantor (1969), Hasegawa-Kishino-Yano (1985) or General-Time-Reversible (Tavaré 1986). It is also possible to specify non-reversible models (i.e., asymmetrical rates among nucleotides).

5.2.7 Mixed ISM-FSM

Both ISM and FSM for SNVs can be incorporated by setting up a **proportion of sites expected to evolve under each type of model (option –p).** Each site is then assigned to each mutation category, ISM/FSM, with probability p and 1-p, respectively. Also, **relative mutation rates (option –w)** can also be specified for ISM and FSM sites

5.3 Single-cell genomics

CellCoal can simulate the technical artifacts resulting from single-cell genomics, like allelic dropout (ADO), allelic imbalance, sequencing error, amplification error, plus doublet cells. For this, CellCoal implements two different strategies:

1. **No sequencing reads are generated**; instead ADO and genotyping errors are directly imposed on the evolved genotypes at the tip of the genealogy.
2. **Read counts are simulated** at a variable depth along the genome, including allelic imbalance, ADO, amplification and sequencing errors. Then, genotype calling is performed under different maximum likelihood models for single-cells.

5.3.1 Fixed or variable allelic dropout (ADO)

During the amplification of single-cells, one or both alleles at a single site can be lost, so a truly heterozygous genotype might appear as a homozygote (e.g., 0/1 => 0/_ or 0/1 => _/1). CellCoal can remove alleles from single chromosomes at a **fixed ADO rate per genotype (option -D)**. Alternatively, ADO rates per genotype can be specified to vary **across cells (option -Q**) and/or **across sites (option -P)** according to a *Beta-binomial* distribution. In this case, the total ADO rate is the sum of both:

```
totalADO = ADOcell + ADOsite - (ADOcell * ADOsite)
```

CellCoal uses a *Beta-binomial* distribution (https://en.wikipedia.org/wiki/Beta-binomial_distribution) to add flexibility in the specification of different parameters, in particular to model variations across cells and sites. CellCoal uses a *Beta* parameterization based on mean and variance (see Fig. 6 for a few examples).

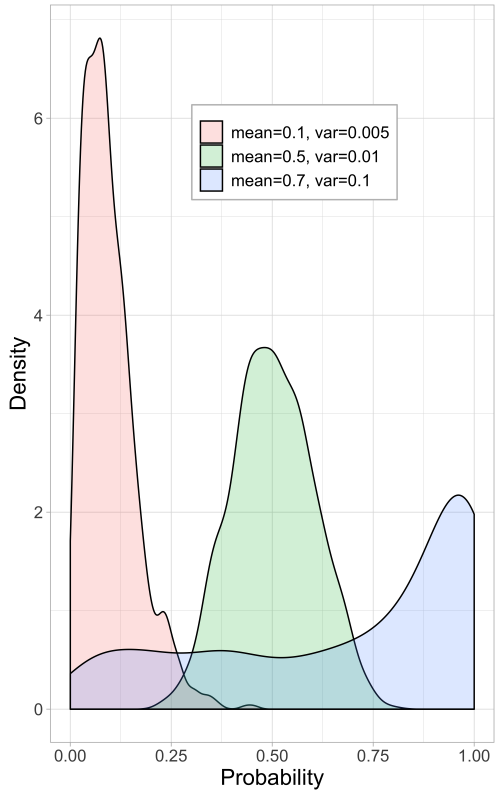

Figure 6: Fig 6. Examples of probability densities for three different *Beta* distributions.

5.3.2 Genotyping error

Genotype errors (option -G) can be directly introduced in the observed genotypes at a given rate per genotype, which is sampled from a *Beta-binomial* distribution (therefore, if desired, it can change across cells and sites). For DNA data, nucleotides are substituted by a different one according to a user-defined error matrix (option -X). These errors encapsulate into a single class different types of error than can take place along the single-cell genomics pipeline, like amplification, sequencing and calling errors.

5.3.3 Sequencing coverage

For each individual genotype the program will generate read counts at a certain sequencing coverage or depth. For a given site the number of reads will follow a *Poisson* distribution around the **mean coverage (option -C)**, unless **coverage dispersion (option -V)** is specified, in which case the coverage per site will follow a *Negative Binomial* distribution. Reads will randomly assigned to the maternal or paternal genome, according to a given **allelic imbalance (option -I)** whose *Beta-binomial* mean by default is 0.5 (i.e., the same probability for maternal or paternal read). If the genotype is haploid due to ADO, the expected **haploid coverage reduction (option -R)**, which is 0.5 by default, can also be specified.

5.3.4 Amplification error

In order to obtain enough DNA material from single-cells for sequencing library construction, the genomes of single cells are most often amplified. During this amplification step, the DNA polymerase can introduce nucleotide errors that can appear in a high proportion of the resulting templates for that site if occurring during the first amplification cycles. If many templates contain an error, many reads will be also be wrong for the specific site. The **amplification error (option -A)** for a given maternal/paternal site follows a *Beta-binomial* distribution with mean and variance specified by the user. The probability of the different

types of errors (e.g., A=>C, A=>G, A=>T ,etc.) can be indicated using an **4x4 error matrix (option –X)**. CellCoal considers two different amplification error models:

Four-template model: In this case all four bases can be present, due to *multiple* amplification errors, in the set of amplified templates (Fig. 7).

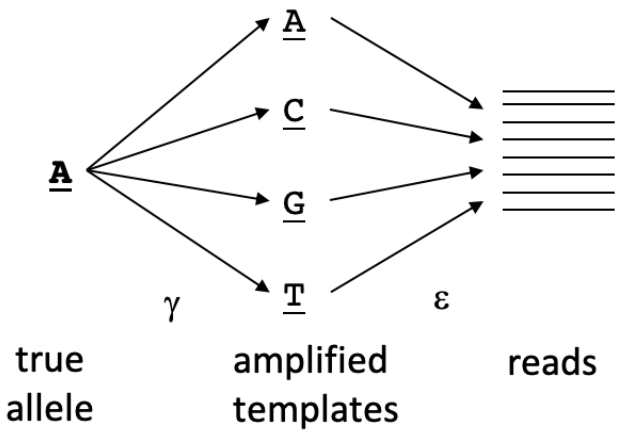

Figure 7: Fig 7. Four-template model for amplification (γ) and sequencing (ε) error.

Two-template model: In this case only the original base and a single alternative can be present, due to a *single* amplification error, in the set of amplified templates (Fig. 8).

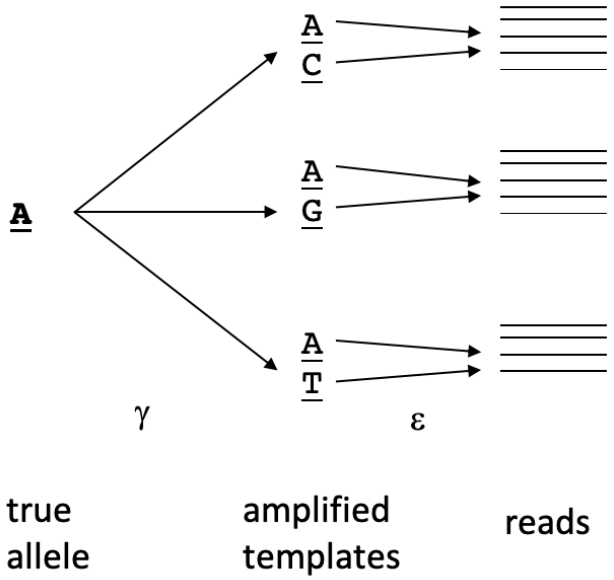

Figure 8: Fig 8. Two-template model for amplification (γ) and sequencing ($varepsilon$) error.

5.3.5 Sequencing error

Once the genomes have been amplified, during sequencing, additional errors can add to the reads. **Sequencing error (option -E)** is assumed to be constant across sites and cells. The probability of the different types of

errors (e.g., A=>C, A=>G, A=>T, etc.) uses the same 4x4 error matrix **(option − X)** as the amplification error.

5.3.6 Cell doublets

Sometimes, due to errors during the cell isolation process, a single sequencing library can contain DNA templates from two cells, and therefore reads which are assume to come from a single cell come in fact from two different cells. This is called a *doublet*. CellCoal can consider the effect of cell doublets by producing reads from two of the sampled cells and joining these reads as if they originated from a single cell, given a user-specified **doublet rate per cell (option -B)**.

5.3.7 Genotype likelihoods

CellCoal can simulate genotype likelihoods for single-cell NGS data, given the read counts simulated, and the sequencing and amplification errors specified by the user, plus ADO, under three different models: GATK standard, 4-template and 2-template models. The basic model used to calculate the genotype likelihoods is similar to that implemented in GATK (DePristo et al. 2011; see also Korneliussen et al. 2013):

$$\Pr\left(D|G=\{A_1,A_2\}\right)=\prod_{i=1}^{M}\Pr\left(b_i|G=\{A_1,A_2\}\right)=\prod_{i=1}^{M}\left(\frac{1}{2}\,p\left(b_i|A_1\right)+\frac{1}{2}\,p\left(b_i|A_2\right)\right)$$

where

G = genotype

A = allele

M = number of reads

bi = observed base in read i

ε = probability of sequencing error

Moreover, it is straightforward to include ADO in this computation, as in Zafar et al. (2016):

$$\Pr\left(D|G=\{A_1,A_2\}\right)=(1-\delta)\prod_{i=1}^{M}\Pr\left(b_i|G=\{A_1,A_2\}\right)+\delta\left(\frac{1}{2}\prod_{i=1}^{M}\Pr\left(b_i|G=\{A_1,-\}\right)+\frac{1}{2}\prod_{i=1}^{M}\Pr\left(b_i|G=\{-,A_2\}\right)\right)=(1-\delta)\prod$$

where δ is the probability of ADO at a given site.

Importantly, CellCoal can calculate the genotype likelihoods, with or without ADO, under three different models: GATK-like, 4-template and 2-template models, which only differ in how the probability of a particular read given the true allele, *p(b|A)*, is calculated, as explained in the next three subsections.

5.3.7.1 GATK-like model

It assumes the same sequencing error rate (ϵ) for all bases and sites, and no amplification error. In this case, the probability of a particular read given the true allele, *p(b|A)*, is:

$$p\left(b|A\right)=\begin{cases}\varepsilon/3, b\neq A\\ 1-\varepsilon, b=A\end{cases}$$

where

b = observed base in read

A = true allele

ε = probability of sequencing error per site

5.3.7.2 Four-template amplification error model

This model extends the previous one in order to consider the amplification error, which, together with the sequencing error, can be different for distinct bases **(option − X)** and that is sampled for each site from a *Beta-binomial* distribution with mean and variance specified by the user **(option -A)**. It allows for multiple amplification errors at a single site, and therefore that all four templates are possible (the correct one and the other three; see Fig. 6 above). In this case, *p(b/A)* is:

$$p\left(b|A\right) = \sum_{j=1}^{4} p\left(t_j|A\right)\ p\left(b|t_j\right) = \sum_{j=1}^{4} \gamma_{A\to t_j}\varepsilon_{t_j\to b}$$

where

$$\gamma_{i\to j} = \begin{cases}\gamma e_{i\to j}, i \neq j \\ 1-\gamma, i = j\end{cases} \quad , \quad \varepsilon_{i\to j} = \begin{cases}\varepsilon e_{i\to j}, i \neq j \\ 1-\varepsilon, i = j\end{cases}$$

and where

b = observed base in read

A = true allele

tj = amplified allele template

ε = probability of sequencing error for a given site

γ = probability of amplification error for a given site

$\gamma_{i\to j}$ = probability of amplification of base i into template base j

$\varepsilon_{i\to j}$ = probability of sequencing error from template base i to read base j

$e_{i\to j}$ = relative probability of amplification/sequencing error from base i to base j

Note that if the amplification and sequencing errors are constant, the probability of observing a read r coming from an allele A simplifies to:

$$p\left(b|A\right) = \begin{cases}\left(1-\gamma\right)\ \varepsilon/3\ +\ \gamma\ \frac{1-\varepsilon/3}{3}, b \neq A \\ \left(1-\gamma\right)\left(1-\varepsilon\right)\ +\ \gamma\ \varepsilon/3, b = A\end{cases}$$

5.3.7.3 Two-template amplification error model

This model is very similar to the previous one, but it assumes that only a single amplification error can occur at a single site, and that therefore only two templates (the "correct" one and a wrong one) are possible (see Fig. 7 above). In this case *p(b/A)* is:

$$p\left(b|A\right) = \sum_{\substack{j=1 \\ j\neq A}}^{4} p\left(t_j|A\right)\ p\left(b|t_j\right) = \sum_{\substack{j=1 \\ j\neq A}}^{4} e_{A\to t_j}\ \left[\left(1-\gamma\right)\ \varepsilon_{A\to b} + \gamma\ \varepsilon_{t_j\to b}\right]$$

where

b = observed base in read

A = true allele

tj = amplified allele template

ε = probability of sequencing error for a given site

γ = probability of amplification error for a given site

$\gamma_{i \to j}$ = probability of amplification of base i into template base j

$\varepsilon_{i \to j}$ = probability of sequencing error from template base i to read base j

$e_{i \to j}$ = relative probability of amplification/sequencing error from base i to base j

6. Arguments

6.1 Coalescent arguments

In this section we describe the arguments that control the simulation of the cell genealogy under the neutral coalescent. Here we can indicate the number of replicates, the number of cells in the sample, the number of sites or loci, and characteristics of the population from which the cells have been sampled, like effective population size, exponential growth or demographic periods.

```
[COALESCENT]
[number of replicates] n100
[number of cells] s20
[number of sites] l1000
[effective population size] e10000
[exponential growth rate] g1e-4 [ln2=0.6931 max]
[demographics] [h2 1000 100 40000 200 30000 20000]
[birth rate] K2
[death rate] L0.3
```

n#: number of replicates; integer [1-inf)

The number of cell samples to be generated. Each sample is an independent realization of the whole process (coalescent, mutations, NGS; see below).

s#: number of cells; integer [2-inf)

The number of cells to be generated for each sample.

l#: number of sites; integer [1-inf)

The total length, in base pairs or nucleotides, of the genomic region simulated. This length includes variable and non-variable sites.

e#: effective population size; integer [1-inf)

The effective size of the population from which the sample was theoretically drawn.

h# # # # demographic periods; several integers [1-inf)

The number of demographic periods, from present to past, and the effective population size during those periods. The first number specifies the number of periods. Then for each period there should be three consecutive numbers indicating the effective population size at the beginning and at the end of the period, and the duration of the period in generations. This option is incompatible with the *exponential growth rate option (-g)* below.

> These parameters are looking back in time, so it is not a good idea to imply a negative growth rate for the last period, as the coalescent time could become infinite in the past.

g#: exponential growth rate; double (-inf, +inf)

The rate of exponential growth per individual per generation. This option is incompatible with the **demographic periods (-d) option**.

> The coalescent looks back in time, so it is not a good idea to specify a negative growth rate, as the coalescent time could become infinite in the past.

K#: birth rate (OI17 model); double [0-inf)

Cell division rate (birth rate) per time unit, in the Ohtsuki and Innan (2017) model. Assumed to be constant over time.

L#: death rate (OI17 model); double [0-inf)

Cell death rate per time unit, in the Ohtsuki and Innan (2017) model. Assumed to be constant over time.

> In the OI model, birth rate >> death rate, so there is exponential growth with rate = birth rate - death rate

6.2 Genealogy modifiers

```
[GENEALOGY MODIFIERS]
[root branch length ratio] k0.5
[outgroup branch length ratio] q1.0
[rate variation among branches] i1.2
```

k#: root branch length ratio; double [0-inf)

Length of the root branch as a proportion of the mean depth of the *sample MRCA*. Note that this branch is attached before introducing rate variation among branches. See Figs. 1-2.

q#: outgroup branch length ratio; double [0-inf)

Length of the outgroup branch as a proportion of the mean depth of the *sample/outgroup MRCA*. Note that this branch is attached before introducing rate variation among branches. See Figs. 1-2.

i#: rate variation among branches; double (0-inf)

Amount of rate variation among lineages, specified with the alpha shape parameter of a continuous *Gamma* distribution with mean 1. The smaller the shape the strongest the heterogeneity introduced. See Fig. 4.

6.3 Mutation models

```
[MUTATION MODEL]
[alphabet binary:0 DNA:1] b1
[germline SNP rate] [c1e-5]
[mutation rate] u1e-6
[deletion rate] d1e-6
[CN\_LOH rate] [H1e-5]
[fixed number of mutations - ISM] [j1000]
[trinucleotide genetic signature - ISM] S2 11 0.3 22 0.7
[alternative mutation model ISMhap:0 Mk2:1 finiteDNA:2] [m0]
[proportion of alternative model sites] [p0]
[alternative/default model relative mutation rate] [w1]
[base frequencies] [f0.3 0.2 0.2 0.3]
[transition/transversion ratio] [t1.7]
[rate variation among sites] [a1.0]
[mutation matrix ACGT x ACGT] [r0.00 0.03 0.12 0.04
0.11 0.00 0.02 0.68
0.68 0.02 0.00 0.11
0.04 0.12 0.03 0.00]
```

b#: alphabet: integer [0/1]

Alphabet for the simulated characters, which can be binary or DNA nucleotides, as follows:

```
0 = binary (0,1 states)
1 = DNA (A,C,G,T sates)
```

c#: germline mutation rate: double [0-inf)

Germline mutation rate per site per generation, which is the same as the probability of introducing a mutation in the ancestral genome (i.e., a SNP in comparison with the reference genome).

u#: somatic mutation rate: double [0-inf)

Nucleotide mutation rate per site per generation.

d#: somatic deletion rate: double [0-inf)

Deletion rate per site per generation under a haploid ISM.

H#: copy-neutral loss-of-heterozygosity rate: double [0-inf)

Copy-neutral LOH rate per site per generation under a haploid ISM.

j#: fixed number of mutations: integer [0-inf)

Number of ISM mutations to be added along the genealogy.

S# # #: trinucleotide genetic signature/s: integer [1-inf), integer [1-30], double [0-1]

Genetic signatures to be implemented, and their proportions. The first number should be the number of distinct signatures, followed by the signature/s IDs and the expected proportion of mutations to be drawn from each particular signature. If these proportions do not add to one they will be rescaled. For example:

S1 5 1.0 implies that mutations will come from signature 5 only.

S2 11 0.3 22 0.7 implies that mutations will come from two signatures: from signature 11 with probability 0.3, and from signature 22 with probability 0.7.

> Note that to get the expected counts under each signature it is important to simulate long genomes, otherwise rare trinucleotide will not be available and, depending on the signature, could bias the observed counts.

m#: alternative mutation model: integer [0-2]

Alternative mutation model (the default model is ISM diploid). The options are as follows:

```
0 = ISM haploid
1 = MK2
2 = finite DNA
```

p#: proportion of alternative model sites: double [0-1]

Proportion of sites that will be simulated under the alternative model specified with the -m option.

w#: alternative/default model relative mutation rate: double [0-inf)

Ratio of the mutation rate for sites evolving under the alternative mutation model, relative to the mutation rate for the sites evolving under the null mutation (option -u).

f# # # #: base frequencies: double [0-inf)

Nucleotide ACGT frequencies at equilibrium for the DNA finite-site models. If they do not add to one they will be rescaled.

t#: transition/transversion ratio: double (0-inf)

Ratio of the number of *transitions* to the number of *transversions* for DNA finite-site models. This ratio, often referred as ti/tv, is 0.5 when *transitions* and *transversions* are equally probable, as there are twice as many possible *transversions* as *transitions* .

a#: rate variation among sites: double (0-inf)

Amount of mutation rate variation among sites, specified with the *alpha shape parameter* of a continuous gamma distribution with mean 1. The smaller the shape the strongest the heterogeneity introduced.

r#4×4: mutation matrix: double [0-inf)

This 4×4 matrix specifies the relative mutation rates (instantaneous substitution rates in the case of finite-site models; often called in this case as Q-matrix) among the different nucleotides, in order ACGT for rows and columns. It does not need to be symmetrical, but all diagonals should be zero. If this option is active, the transition/transversion ratio (-t) is ignored. For example, for the Jukes-Cantor (1969) model this matrix would be:

$$\mathbf{Q} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

For the General-Time-Reversible (GTR) (Tavaré 1986) model:

$$\mathbf{Q} = \begin{bmatrix} 0 & 2.1 & 0.3 & 5.6 \\ 2.1 & 0 & 1.5 & 7.8 \\ 0.3 & 1.5 & 0 & 1.4 \\ 5.6 & 7.8 & 1.4 & 0 \end{bmatrix}$$

And for a General-Time-Non-Reversible (GTnR) model:

$$\mathbf{Q} = \begin{bmatrix} 0 & 2.1 & 0.3 & 5.6 \\ 1.9 & 0 & 1.5 & 7.8 \\ 2.1 & 3.7 & 0 & 1.4 \\ 0.8 & 2.4 & 0.9 & 0 \end{bmatrix}$$

6.4 scWGA parameters

```
[scWGA PARAMETERS]
[fixed allelic dropout (ADO)] [D0.2]
[ADO per cell] [P0.1 0.01]
[ADO per site] [Q0.1 0.01]
[maternal allelic imbalance] [I0.5 0.01]
[amplification error] A0.2 0.01 0
[doublet rate per cell] [B0.0]
```

D#: fixed allelic dropout (ADO) rate: double [0-1]

Fixed probability of not amplifying one allele, per genotype and cell.

P#: ADO per cell (mean, variance): double [0-1] double (0-inf)

Probability of allelic dropout at a given genotype, per cell. This probability follows a *Beta-binomial* distribution with *mean* and *variance* specified by the user.

> WARNING: In CellCoal, for the *Beta-binomial* distributions the variance has to be bigger than 0 and smaller than *mean* × (1-*mean*).

Q#: ADO per site (mean, variance): double [0-1] double (0-inf)

Probability of allelic dropout at a given genotype, per site. This probability follows a *Beta-binomial* distribution with *mean* and *variance* specified by the user.

I#: maternal / paternal allelic imbalance: (mean, variance): double [0-1] double (0-inf)

Probability of a read coming from the maternal chromosome relative to the probability of coming from the paternal chromosome. This probability follows a *Beta-binomial* distribution with *mean* and *variance* specified by the user.

A# # #: amplification error (mean, variance, model): double [0-1] double (0-inf) int [0/1]

Expected *Beta-binomial* probability (*mean*) of introducing an amplification error per read per site, plus the *variance* of this error across sites, plus the amplification model (0: four-template-model; 1: two-template model). For DNA data, the specific amplification error is selected according to the error matrix (option -X).

B#: doublet rate per cell: double [0-1] double (0-inf)

Probability of a doublet per cell, which follows a *Beta-binomial* distribution with *mean* and *variance* specified by the user.

6.5 Genotyping parameters

```
[GENOTYPING PARAMETERS]
[genotyping error; no reads] [G0.1 0.01]
```

G#: genotyping error (mean, variance): double [0-1] double (0-inf)

Probability of introducing an error (false positive or false negative), as a result of amplification/sequencing/calling altogether, per genotype and cell.

This probability follows a *Beta-binomial* distribution with *mean* and *variance* specified by the user. Note that the variance has to be bigger than 0 and smaller than *mean* $\times$ (1-*mean*).

In the case of DNA data, the specific error is selected according to the error matrix (option -X).

If this option is active, no reads are simulated. In fact, it is not allowed to specify -G and -C (see below) at the same time.

6.6 NGS parameters

```
[NGS PARAMETERS]
[sequencing coverage; reads simulated] C10
[coverage dispersion] [V5]
[haploid coverage] [R0.5]
[sequencing error] [E0.00]
[error matrix ACGT x ACGT] X0 1 1 1
1 0 1 1
1 1 0 1
1 1 1 0
```

C#: sequencing coverage: integer [1-inf)

Expected number of reads per site and cell. It is the mean of a *Poisson* distribution if there is no dispersion (option -V). In the case of dispersed coverage (option -V $>$ 0), this is the mean of a *Negative Binomial distribution*. If there is no allelic imbalance (option -I = 0.5), then the reads are randomly assigned to the maternal/paternal chromosomes.

V#: coverage dispersion: double (0-inf)

Alpha shape of the gamma distribution in a Negative Binomial distribution (i.e., *dispersion* parameter) whose mean *mu* is the sequencing coverage (option -C). The smaller this value is, the more dispersed is the coverage.

> Note: A *Negative-binomial* distribution can arise as a mixture of *Poisson* distributions with mean distributed as a Gamma distribution with scale parameter (1 - *prob*)/*prob* and shape parameter *dispersion*, where *prob* = *dispersion*/(*dispersion*+*mu*). The variance is *mu* + *mu*2/*dispersion*.

R#: haploid coverage: double [0-1]

Expected proportion of reads (*expected coverage*, option -C) to be obtained from a haploid template (e.g., after a deletion or ADO) in comparison with the expected coverage for a diploid region.

E#: sequencing error: double [0-1]

Probability of introducing a (constant) sequencing error per read site. For DNA data, the specific sequencing error is selected according to the error matrix (option -X)

X#4×4: error matrix: double [0-inf)

This 4×4 matrix specifies the relative error rates among the different nucleotides due to sequencing or amplification errors (the matrix is the same for both types of errors), in order ACGT for rows and columns. It does not need to be symmetrical, but all diagonals should be zero.

6.5 Output options

All the output files produced by the program are printed inside folder called by default *results*, located at the same level as the executable. CellCoal will always output a least two files: a log file and a vcf file.

The *log file* contains a summary of the simulation settings. The *vcf* file contains the SNV genotypes in standard VAriant Calling Format (.VCF; https://samtools.github.io/hts-specs/VCFv4.3.pdf).

Other, optional outfiles can contain full information on genotypes (-1, -3) or haplotypes (-2, -4, -9), or about the trees (–6), times (–7). For a detailed description of these options see below.

```
[OUTPUT]
[print SNV genotypes] 1
[print SNV haplotypes] 2
[print full genotypes] 3
[print full haplotypes] 4
[print ancestors] 5
[print trees] 6
[print times] [7]
[print CATG] [8]
[print true haplotypes] 9
[print replicates in individual folders] v
[print consensus IUPAC haplotypes] [x]
[results folder name] oresults
[user tree file] [Tusertree.0001]
[user genome file] [Uusergenome.fas]
```

1: print SNV genotypes

Prints the observed (if no reads are produced) or maximum likelihood (ML) (if reads are produced, option -C) maternal and paternal genotypes at variable sites (single nucleotide variants – SNVs), to a file called *snv_gen* inside the *results* folder, in *Phylip* format (http://evolution.genetics.washington.edu/phylip/doc/sequence.html). If the option -v is active, each replicate will be printed to an individual file called *snv_gen.####*, where #### is the replicate number, in a subfolder *snv_genotypes_dir* inside the *results* folder.

2: print SNV haplotypes

Prints the observed/ML maternal and paternal haplotypes at SNV sites to a file called *snv_hap* inside the *results* folder. If the option -v is active, each replicate will be printed to an individual file called *snv_hap.####*, in a subfolder *snv_haplotypes_dir* inside the *results* folder. If the option -x is active, a single consensus sequence with IUPAC codes will be printed (see option -x below).

3: print full genotypes

Prints the observed/ML genotypes at all sites to a file called *full_gen* inside the *results* folder. If the option -v is active, each replicate will be printed to an individual file called *full_gen.####*, in a subfolder *full_genotypes_dir* inside the *results* folder.

4: print full haplotypes

Prints the observed/ML maternal and paternal haplotypes at all sites to a file called *full_hap* inside the *results* folder. If the option -v is active, each replicate will be printed to an individual file called *full_hap.####*, in a subfolder *full_haplotypes_dir* inside the *results* folder. If the option -x is active, a single consensus sequence with IUPAC codes will be printed (see option -x below).

5: print MRCA sequences

Prints genotypes/haplotypes also for the sample MRCA and sample/outgroup MRCA (see Figure 3).

6: print trees

Prints the cell genealogy for each replicate in Newick format.

7: print times

Prints for each replicate a list of the coalescent events and branch lengths in coalescent time.

8: print CATG

Prints read counts in CATG format (in addition to VCF). Basically, in the CATG format the first line has the number of samples and the number of sites. Following this is a transposed data matrix with the consensus base calls where each row is a site and each column a different sample. To the right of each site is a tab-separated list of counts of the four bases at that site in the order C, A, T, and G. More information at http://nbviewer.jupyter.org/gist/dereneaton/d2fd4e70d29f5ee5d195/testing_cat.ipynb#View-the-.cat-results-files. For example:

```
12 44500
1A0 1B0 1C0 1D0 ...
YCCCCCCCCCCC 10,0,10,0 20,0,0,0 20,0,0,0 20,0,0,0 ...
GGGGGGGGGGGG 0,0,0,20 0,0,0,20 0,0,0,20 0,0,0,20 ...
AAAAAAAAAAAA 0,20,0,0 0,20,0,0 0,20,0,0 0,20,0,0 ...
CCCCCCCCCCCC 20,0,0,0 20,0,0,0 20,0,0,0 20,0,0,0 ...
AAAAAAAAAAAA 0,20,0,0 0,20,0,0 0, 20,0,0 0,20,0,0 ...
```

9: print true haplotypes

Prints the true maternal and paternal haplotypes at all sites to a file called *true_hap* inside the *results* folder. If the option -v is active, each replicate will be printed to an individual file called *true_hap.####*, in a subfolder *true_haplotypes_dir* inside the *results* folder. If the option -x is active, a single consensus sequence with IUPAC codes will be printed (see option -x below).

v: print replicates in individual files

Prints the outputfiles for each replicate

x: print consensus IUPAC haplotypes

Prints maternal and paternal sequence as a single consensus sequence with IUPAC codes as follows:

IUPAC codes					
A/A => A	A/C => M	A/G => R	A/T => W	A/_ => a	A/- => a
C/A => M	C/C => C	C/G => S	C/T => Y	C/_ => c	C/- => c
G/A => R	G/C => S	G/G => G	G/T => K	G/_ => g	G/- => g
T/A => W	T/C => Y	T/G => K	T/T => T	T/_ => t	T/- => t
_/A => a	_/C => c	_/G => g	_/T => t	/ => -	_/- => -
-/A => a	-/C => c	-/G => g	-/T => t	-/_ => -	-/- => -

o$: results folder name

Name of the folder where the output data files will be stored, at the same level as the program executable. By default, this folder is called *results*.

T$: user tree file

Name of the file with a rooted tree with branch lengths in Newick format to be used as the cell genealogy for the simulation of genotypes, instead of a random coalescent tree.

```
((D:0.03,B:0.04):0.02,(C:0.06,A:0.07):0.05);
```

U$: user genome file

Name of the file with a diploid reference genome in FASTA format, with a maximum size of 106 sites.

```
>maternal genome
AAAAAAACCTGAAACCTTCAGAAGCTGTTTCAGTAGGTAAAGAAAAAGG...
>paternal genome
AAAAAAACCTGAAACCTTCAGAAGCTGTTTCAGTAGGTAAAGAAAAAGG...
```

6.6 Other options

```
[OTHER]
[tumor labels] [W]
[noisy] y1
[seed] #1542634309 [if no seed is specified, the current time will be used as seed]
```

-W: tumor labels Use tumor nomenclature for cell labels

Meaning	Standard label	Tumor label
sample cell 1	cell0001	tumcell0001
outgroup cell	outgcell	healthycell
sample MRCA	ingrroot	tumoralroot
sample-outgroup MRCA	outgroot	healthyroot

-y#: noisy

Amount of information to be printed to the screen:

```
0: does not print anything
1: + simulation summary
2: + replicate information
3: + calculation status and event information.
```

-##: seed

Seed for the random number generator. If no seed is specified, the current time will be used as seed.

6.7 Parameter file name at the command line

By default, the parameter file is called *parameters*. Alternatively, the user can indicate the name of the parameter file name at the command line with the **option -F**, like this:

```
bin/cellcoal-x.y.z -Fexamples/parameters.ISM.binary
```

7. Default settings

By default, CellCoal will simulate 10 replicates of samples with 20 ingroup cells plus one outgroup cell, obtained from an exponentially growing population (SH91 model) with a growth rate of 1e-03 and a current effective population size of 10000. The root and outgroup branch length ratios will be 0.5, and 1, respectively. Each cell will harbor 1000 genomic sites evolving with a mutation rate of 1e-07 under a DNA ISM diploid model. By default, there will be no ADO, amplification or sequencing error, and the sequencing depth will be 5X. Apart from the obligatory log ("*log*") and VCF ("*vcf*") files, by default only the SNV genotypes ("*snv_gen*") and the genealogies ("*trees*") will be printed to the results folder ("*results*") To run the program with the equivalent settings we could just type:

```
bin/cellcoal-x.y.z -n10 -s20 -l1000 -e10000 -g0.001 -k0.5 -q1 -b1 -u1.0e-07 -m0 -p0.0
-D0.0 -C5 -1 -6 -oresults -y1
```

8. References

Alexandrov LB et al. 2013. Signatures of mutational processes in human cancer. *Nature* 22: 415-421.

DePristo MA et al. 2011. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics* 43: 491–498.

Felsenstein J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution* 17:368-376.

Hasegawa M, Kishino H and Yano T. 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution* 22:160-174.

Hudson RR. 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* 18:337-338.

Jukes TH and Cantor CR. 1969. Evolution of protein molecules. Pp. 21-132 in H. M. Munro, ed. Mammalian Protein Metabolism. Academic Press, New York, NY.

Lewis PO. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. *Systematic Biology* 50: 913-925.

Korneliussen TS et al. 2013. Calculation of Tajima's D and other neutrality test statistics from low depth next-generation sequencing data. BMC Bioinformatics 14:289.

Ohtsuki H, Innan H. 2017.Forward and backward evolutionary processes and allele frequency spectrum in a cancer cell population. *Theoretical Population Biology* 117: 43-50.

Slatkin M, Hudson RR. 1991. Pairwise comparisons of mitochondrial DNA sequences in stable and exponentially growing populations. *Genetics* 129: 555-562.

Tavaré S. 1986. Some probabilistic and statistical problems in the analysis of DNA sequences. Pp. 57-86 in R. M. Miura, ed. Some mathematical questions in biology - DNA sequence analysis. Amer. Math. Soc., Providence, RI.

Yang Z. 1996. Among-site rate variation and its impact on phylogenetic analysis. *Trends in Ecology and Evolution* 11:367-372.

Zafar H, Wang Y, Nakhleh L, Navin N, Chen K. 2016. Monovar: single-nucleotide variant detection in single cells. Nat Meth. 13:505–507.