# Documentation for SimSpliceEvol2

**Abstract**

SimSpliceEvol2 is a tool for simulating the evolution of gene sequences that integrates exon-intron structure evolution and sets of transcript evolution through alternative splicing events. It takes a guide gene tree as input and generates a gene sequence with its transcripts for each node of the tree, from the root to the leaves. SimSpliceEvol2 is useful for evaluating tools for spliced sequence analysis, such as spliced alignment methods and transcript phylogenies reconstruction methods.

## Getting started with SimSpliceEvol2

If you're a newcomer to SimSpliceEvol2, it's recommended to begin here with the section 1

## Details of the evolution simulation (2)

1. Structure evolution model (see section 2.1)
   (a) Evolutionary events acting on the exon-intron structure of genes (see subsection 2.1.1)
   (b) Evolutionary events acting on the set of transcripts of genes (see subsection 2.1.2)
2. Sequence evolution model (see section 2.2)
   (a) Exon sequence evolution (see subsection 2.2.1)
   (b) Intron sequence evolution (see subsection 2.2.2)

## Tutorial (3)

## Possible errors (4)

# 1 Quickstart : SimSpliceEvol2 installation and fundamental usage

## 1.1 Program Installation and running

Installing SimSpliceEvol2 is straightforward. First, download the latest SimSpliceEvol2 release (e.g., SimSpliceEvol2.0.0) to your computer. Ensure that you chose the correct distribution according to your Operating System (Linux or Windows) and Python (Python version 3.6 or higher) is installed on your system. See the `https://simspliceevol.cobius.usherbrooke.ca/release` for accessing the releases page.

To launch the graphical version (GUI) of SimSpliceEvol2, simply double-click the application, or run it from the command line without any options:

```
./simspliceevol2
```

If you want to run the command line version of SimSpiceEvol2, you need to open a console.

- **LINUX**: If you are using a Linux OS, you likely know how to open a terminal ☺.
- **WINDOWS**: To open the Command Prompt on Windows, press Windows + R. A small dialog window will appear; type cmd and click OK.

## 1.2 Using the graphical user interface (GUI) of SimSpliceEvol2

The main interface displays the options that can be set as you wish. As the program has many options, they are sorted out in different groups:

- **Mandatory option**: This option, indicated in red, represents the guide tree with its branch lengths in Newick format.
- **Output**: output folder
- **All other options**: options by theme
  - **constants**
  - **parameters acting on exon sequences**
  - **parameters acting on exon-intron structure**
  - **parameters acting on the set of transcripts**

After setting the options to your desired values, initiate the task execution by clicking the ***run simulation*** button located in the bottom.

Please note that the command line corresponding to your selected options is generated when clicking the ***generate command*** button and displayed in the window. You can use the GUI to effortlessly create the command line needed for a single example. Simply copy this command (using the ***copy command*** button) and modify it to execute on multiple datasets using SimSpliceEvol2 command line capabilities.

## 1.3 Using the command line version of SimSpliceEvol2

When using SimSpliceEvol2, the required step is to specify the file containing the Newick format of the guide tree you wish to utilize for the simulation, done through the ***-i*** option. If you're unsure, simply pick the example guide tree distributed with the application (***small.nw***).

```
./simspliceevol2 -app no -i small.nw
```

If you attempt to run SimSpliceEvol2 without any options, it won't display any help; instead, it will launch its graphical mode. To start the command-line version, you need to specify two options: ***-app***, which should be set to ***no***, and ***-i***. By default, the ***-app*** option is set to ***yes***.

Additionally, note that if incorrect options are used or mandatory ones are missing, the available options will be displayed. To see all of them, you can enter the following command:

```
./simspliceevol2 -app no -h
```

# 2 Details of the evolution simulation

SimSpliceEvol2 integrates two evolutionary models, operating in tandem. The first model, known as the structure evolution model 2.1, influences the evolution of the exon-intron structure and the resulting transcript set of the gene. The second model, the sequence evolution model 2.2, governs the evolution of the gene sequence itself. In the input guide tree, branch length denotes the anticipated number of substitution events per codon in coding sequences along that branch. To determine the expected numbers of other evolutionary events affecting the exon-intron structure, transcript set, or sequence along a branch, we employ a linear model. This model assumes that all rates are linearly correlated with the substitution rate.

## 2.1 Structure evolution model

The structure evolution model implemented in SimSpliceEvol2 builds upon the Christinat-Moret model of transcript evolution, as outlined in [1]. This model operates on two levels of evolution: one at the gene level, influencing the exon-intron structure, and the other at the transcript level, governing the sets of transcripts generated from a gene. Below, we outline the evolutionary events incorporated into the model at each level.

### 2.1.1 Evolutionary events acting on the exon-intron structure of genes

The evolution of the gene's exon-intron structure is influenced by three fundamental events: **exon loss**, **exon gain**, and **exon duplication**.

- **Exon loss**: Exon loss occurs when an exon is eliminated from the gene's structure and no longer belongs to any transcript. This event is simulated by deleting the exon segment from the gene sequence.
- **Exon gain**: Exon gain involves the emergence of a new exon segment within an existing intron. This is achieved by generating a new exon segment and inserting it into an existing intron segment.
- **Exon duplication**: Exon duplication entails the tandem replication of an exon segment with the insertion of an intron segment between the two copies.

The sequential application of these events along a branch of the guide tree results in modifications to the gene's exon-intron structure.

In a branch of the guide tree, where the length `c_s_r` represents the anticipated number of substitution events per codon, the expected number of exon-intron structure change (**EIC**) events per exon on the branch is determined by multiplying `k_eic` by `c_s_r`, where `k_eic` is a constant specified by the user.

**Note** : If you wish to increase the number of exon-intron structure change (**EIC**) events on a branch with a small `c_s_r`, you can achieve this by adjusting the constant `k_eic`. By increasing this constant, you can observe the desired changes and obtain the desired results.

To determine the influence of **EIC** events on the evolution of gene exon-intron structures, three additional user-defined parameters are utilized. These parameters represent the relative frequencies of exon-intron structure changes attributed to exon loss (eic_l), exon gain (eic_g), and exon duplication (eic_d), respectively. We recommend ensuring that the sum of these three relative frequencies equals 1.0. Although the software allows for the sum of these relative frequencies to exceed 1.0 based on user preferences, it's worth noting that the webserver will restrict this possibility.

**Example: (Gene structure evolution model)** Taking into account the guide tree depicted in Figure 1 and the gene structure of `A`, we aim to determine the exon-intron structure of gene `B`. This determination involves considering the codon substitution rate (the branch length) between genes `A` and `B`. To achieve this, we follow the sequence of events: loss $\rightarrow$ gain $\rightarrow$ duplication. Note that the value if `n`, i.e. the number of exons can be different after each event.

1. The number of exon lost in `B`, denoted by `#exons_lost` equals to:

$$\texttt{\#exons\_lost} = \mathit{round}(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_eic} \times \texttt{eic\_l}) \tag{1}$$

   At this stage, gene `A` comprises 5 exons, with default values for `k_eic` and `eic_l` set at 5 and 0.4, respectively. So,

$$\texttt{\#exons\_lost} = \mathit{round}(5 \times 0.065 \times 5 \times 0.4) = \mathit{round}(0.65) = 1$$

   One exon (`exon_5` in the figure) will be randomly selected from the 5 transcripts and removed from the structure, resulting in a total of 4 exons.

2. The number of exon gained in `B`, denoted by `#exons_gained` equals to:

$$\texttt{\#exons\_gained} = \mathit{round}(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_eic} \times \texttt{eic\_g}) \tag{2}$$

   At this stage, gene `A` comprises 4 exons, with default values for `k_eic` and `eic_g` set at 5 and 0.5, respectively. So,

$$\texttt{\#exons\_gained} = \mathit{round}(4 \times 0.065 \times 5 \times 0.5) = \mathit{round}(0.52) = 1$$

   One exon (`exon_6` in the figure) will be added at a randomly position in the gene structure, resulting in a total of 5 exons.

3. The number of exon duplicated in B, denoted by `#exons_duplicated` equals to:

$$\texttt{\#exons\_duplicated} = round(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_eic} \times \texttt{eic\_d}) \quad (3)$$

At this stage, gene A comprises 5 exons, with default values for `k_eic` and `eic_d` set at 5 and 0.1, respectively. So,

$$\texttt{\#exons\_duplicated} = round(5 \times 0.065 \times 5 \times 0.1) = round(0.1625) = 0$$

If `#exons_duplicated` is equal to 0, no exon will be selected for duplication. However, if `#exons_duplicated` is greater than or equal to 1, then the corresponding number of exons will be randomly selected for tandem duplication. For example, if `#exons_duplicated` is set to 1 and `exon_2` is chosen, an `exon2_dup` will be inserted near `exon_2`. The position of `exon2_dup`, whether before or after `exon_2` in the gene structure, will also be determined randomly.

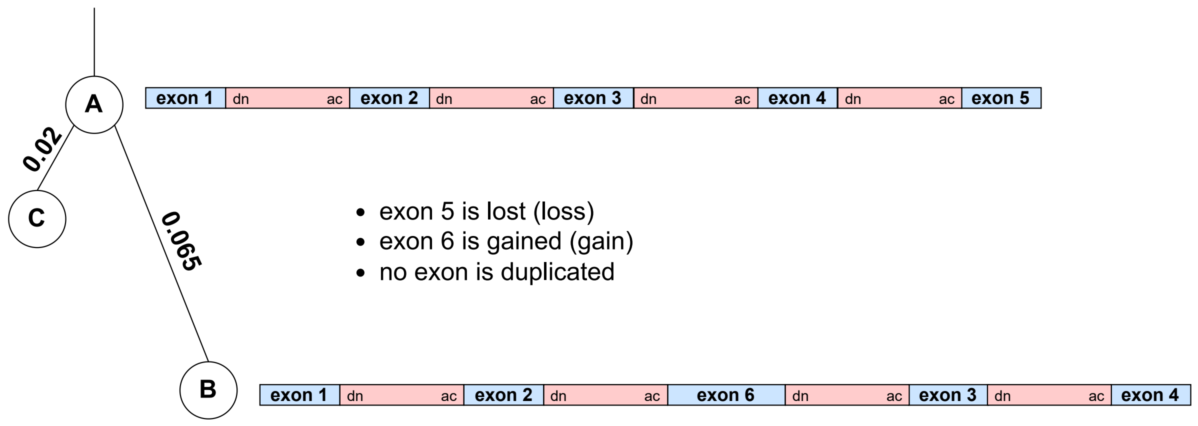
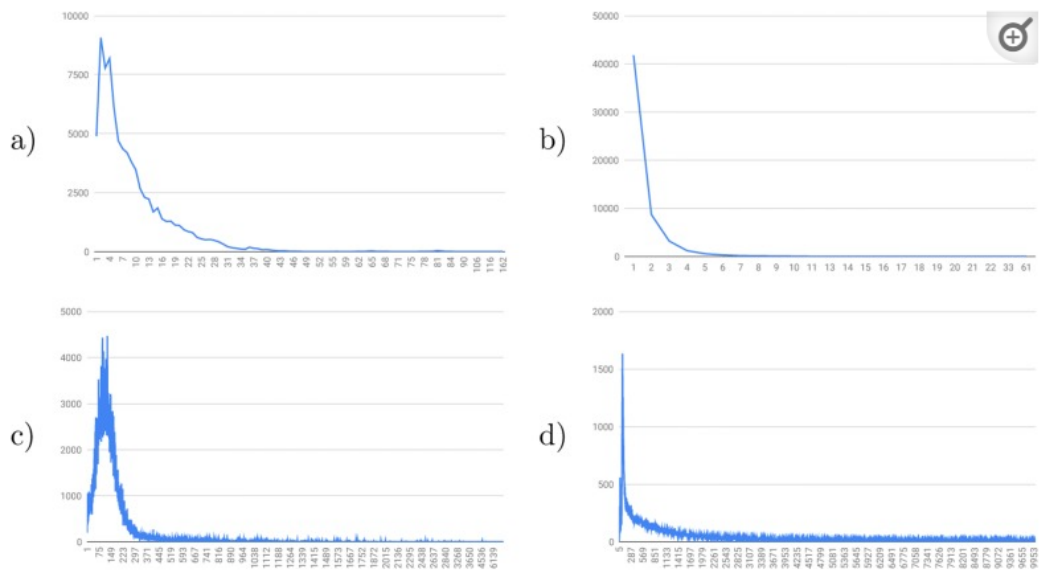


Figure 1: **Simulation**: Evolutionary events acting on the exon-intron structure of genes

**Special case: What process is involved in generating the gene structure at the root of the guide tree?** Using the probability distributions (2), the approach initially determines the quantity of exons for the gene. It samples the maximum number, `m`, of exons per transcript, then defines the gene's total number of exons as `k_nb_exons` $\times\ m$, where `k_nb_exons` $\geq 1$ represents a user-defined constant.

To simulate the structure of gene A, we begin by sampling the maximum number, `m`, which follows a normal distribution with $\mu = 9.62$ and $\sigma = 8.54$ (see figure 2). For example, if `m` $= 3.5$, we then multiply it by the constant `k_nb_exons`, which by default is 1.5. Consequently, the number of exons in A is calculated as 5.25, rounded to the nearest whole number, resulting in 5.

**Note**: Increasing the constant `k_nb_exons` allows for easily attaining a greater number of exons in the gene structure at the root.



Probability distributions of the composition of gene sequences derived from Ensembl Compara 96 coding genes. **a** Number of exons per cDNA transcript (mean: 9.62; std:8.54). **b** Number of cDNA transcripts per gene (mean: 1.45; std:1.08). **c** Exon length (mean: 170.36; std:258.86). **d** Intron length (mean: 3730.30; std:20126.39)

Figure 2: Probability distributions of the composition of gene sequences derived from Ensembl Compara 96 coding genes.

### 2.1.2 Evolutionary events acting on the set of transcripts of genes

The evolution of transcript sets produced from genes is influenced by both changes in the gene's exon-intron structure and events occurring at the transcript level. At the transcript level, two elementary events can impact the set of transcripts: transcript loss and transcript creation.

Transcript loss occurs when a gene ceases to produce a specific type of transcript due to mutations or changes in gene expression regulation. Simulating a transcript loss event involves halting the generation of a transcript starting from the node where it is lost. It's important to note that the model assumes that the loss of an exon at the gene level results in the loss of all transcripts containing that exon.

Transcript creation occurs when a gene begins producing a new type of transcript, involving a new combination of exons from its exon-intron structure. Alternative splicing determines which alternative exons are present or absent in each transcript of a gene. Simulating a transcript creation event involves generating a new transcript either randomly from the set of all possible isoforms or by applying alternative splicing to an existing transcript.

In a branch of the guide tree with length `c_s_r`, the anticipated number of transcript change (`TC`) events per transcript on the branch is determined by multiplying `k_tc` by `c_s_r`, where `k_tc` is a user-defined constant.

**Note** : If you wish to increase the number of transcript change (**TC**) events on a branch with a small `c_s_r`, you can achieve this by adjusting the constant `k_tc`. By increasing this constant, you can observe the desired changes and obtain the desired results.

The six user-defined parameters `tc_rs`, `tc_a5`, `tc_a3`, `tc_es`, `tc_me`, and `tc_ir`, which are used for generating transcripts at the root of the guide tree, also serve as the relative frequencies of `TC` events by *random selection* (`rs`), *alternative 5' splice-site selection* (`a5`), *alternative 3' splice-site selection* (`a3`), *exon skipping* (`es`), *mutually exclusive exons* (`me`), and *intron retention* (`ir`), respectively.

An additional user-defined parameter, the relative frequency of `TC` events by transcript loss `tc_tl`, is utilized to specify the relative proportion of `TC` events caused by transcript loss. While we advise ensuring that the sum of these seven relative frequencies equals 1.0, users are not constrained by this recommendation within the software, although limitations may be imposed by the webserver.

**Example: (Transcript evolution model)** Considering the guide tree illustrated in Figure 3, along with the gene structure of `gene_1` and its corresponding set of transcripts, our objective is to ascertain the set of transcripts for gene `gene_3`, presuming that its structure has already been determined(in the step exon-intron evolution). This determination involves considering the codon substitution rate (the branch length equals to 0.2) between genes `gene_1` and `gene_3`. To achieve this, we follow the sequence of events: comparison of genes structure → loss → rs → `es` → `me` → `a5` → `a3` → `ir` . Please note that the value `n`, representing the number of transcripts, may vary after each event.

1. The number of transcripts lost in `gene_3` by comparing its structure with the structure of its parent `gene_1`, denoted by `#transcript_lost_structure`. In Figure 3, we observe that out of 7 transcripts, five are lost. It's crucial to note that if an exon is lost in the gene structure, the transcripts containing this exon are automatically lost as a consequence.

   $$\texttt{\#transcript\_lost\_structure} = 5$$

   Five transcripts have been removed from the `gene_3`, resulting in a total of 2 transcripts available.

2. The number of transcript lost in `gene_3`, denoted by `#exons_lost` equals to:

   $$\texttt{\#transcript\_lost} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_tl}) \tag{4}$$

   At this stage, 2 transcripts are available, with default values for `k_tc` and `tc_tl` set at 5 and 0.1, respectively. So,

   $$\texttt{\#transcript\_lost} = ceil(2 \times 0.2 \times 5 \times 0.1) = ceil(0.2) = 1$$

   One transcript (`1#7` in the figure) will be randomly selected for removal from the current set of available transcripts.

3. The number of transcript created, denoted by `#transcript_created` equals to 1 if `tc_rs` is *null*, then only one transcript (one with all the exons in the gene structure) is selected. Alternatively, if `tc_rs` $\neq 0$, we randomly select a number $m$ of transcripts, following a normal distribution with a mean ($\mu$) of 1.45 and a standard deviation ($\sigma$) of 1.08 (refer to Figure 2). In Figure 3, let $m = round(x)$, where $x$ is randomly selected from the mentioned normal distribution. As a result, one transcript is added to the set of available transcripts (transcript `3#0` in Figure 3).

4. The number of transcript available that undergone `es`, denoted by `#transcript_es` equals to:
$$\texttt{\#transcript\_es} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_es}) \tag{5}$$
At this stage, 2 transcripts are available, with default values for `k_tc` and `tc_tl` set at 5 and 0.3, respectively. So,
$$\texttt{\#transcript\_es} = ceil(2 \times 0.2 \times 5 \times 0.3) = ceil(0.6) = 1$$
One transcript (`3#0` in the figure) will be randomly chosen to undergo an `es` event. An exon in the transcript's exon composition will then be randomly selected for removal, resulting in the formation of the new transcript (`3#2` in the figure) derived from the `es` event.

5. The number of transcript available that undergone `me`, denoted by `#transcript_me` equals to:
$$\texttt{\#transcript\_me} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_me}) \tag{6}$$
At this stage, 3 transcripts are available, with default values for `k_tc` and `tc_tl` set at 5 and 0.2, respectively. So,
$$\texttt{\#transcript\_me} = ceil(3 \times 0.2 \times 5 \times 0.2) = ceil(0.6) = 1$$
One transcript (`3#0` in the figure) will be randomly selected to undergo an `me` (mutually exclusive exons) event. An exon in the transcript's exon composition will then be randomly chosen for removal in `3#0`, and one exon either before or after the selected exon will also be removed. This process forms the new transcript (`3#3` in the figure), which is derived from the `me` event. SimSpliceEvol operates only the tandem mutually exclusive exons.

6. The number of transcript available that undergone `a5`, denoted by `#transcript_a5` equals to:
$$\texttt{\#transcript\_a5} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_a5}) \tag{7}$$
At this stage, 4 transcripts are available, with default values for `k_tc` and `tc_a5` set at 5 and 0.1, respectively. So,
$$\texttt{\#transcript\_a5} = ceil(4 \times 0.2 \times 5 \times 0.1) = ceil(0.4) = 1$$

One transcript (`1#1` in the figure) will be randomly selected to undergo an `a5` event. An exon in the transcript's exon composition will then be randomly chosen for either adding a 3' prime sequence or removing it. It's important to note that the number of nucleotides chosen is a multiple of 3 and is selected from the range between 6 and 54, representing at most one-third of the minimum length of an exon (see Figure 2). This process results in the formation of the new transcript (`3#4` in the figure), derived from the `a5` event.

7. The number of transcript available that undergone `a3`, denoted by `#transcript_a3` equals to:

$$\texttt{\#transcript\_a3} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_a5}) \tag{8}$$

   At this stage, 5 transcripts are available, with default values for `k_tc` and `tc_a3` set at 5 and 0.1, respectively. So,

$$\texttt{\#transcript\_a3} = ceil(5 \times 0.2 \times 5 \times 0.1) = ceil(0.5) = 1$$

   One transcript (`3#0` in the figure) will be randomly selected to undergo an `a3` event. An exon in the transcript's exon composition will then be randomly chosen for either adding a 5' prime sequence or removing it. It's important to note that the number of nucleotides chosen is a multiple of 3 and is selected from the range between 6 and 54, representing at most one-third of the minimum length of an exon (see Figure 2). This process results in the formation of the new transcript (`3#5` in the figure), derived from the `a3` event.

8. The number of transcript available that undergone `ir`, denoted by `#transcript_ir` equals to:

$$\texttt{\#transcript\_ir} = ceil(\texttt{n} \times \texttt{c\_s\_r} \times \texttt{k\_tc} \times \texttt{tc\_ir}) \tag{9}$$

   At this stage, 6 transcripts are available, with default values for `k_tc` and `tc_ir` set at 5 and 0.1, respectively. So,

$$\texttt{\#transcript\_a3} = ceil(6 \times 0.2 \times 5 \times 0.1) = ceil(0.6) = 1$$

   One transcript (`3#2` in the figure) will be randomly selected to undergo an `ir` event. An exon in the transcript's exon composition will then be randomly chosen for either adding a 5' prime intronic sequence or 3' prime intronic sequence. This process results in the formation of the new transcript (`3#6` in the figure), derived from the `ir` event.

**Special case: What process is involved in generating the set of transcripts at the root of the guide tree?** The process is the same as mentionned in the previous point 3

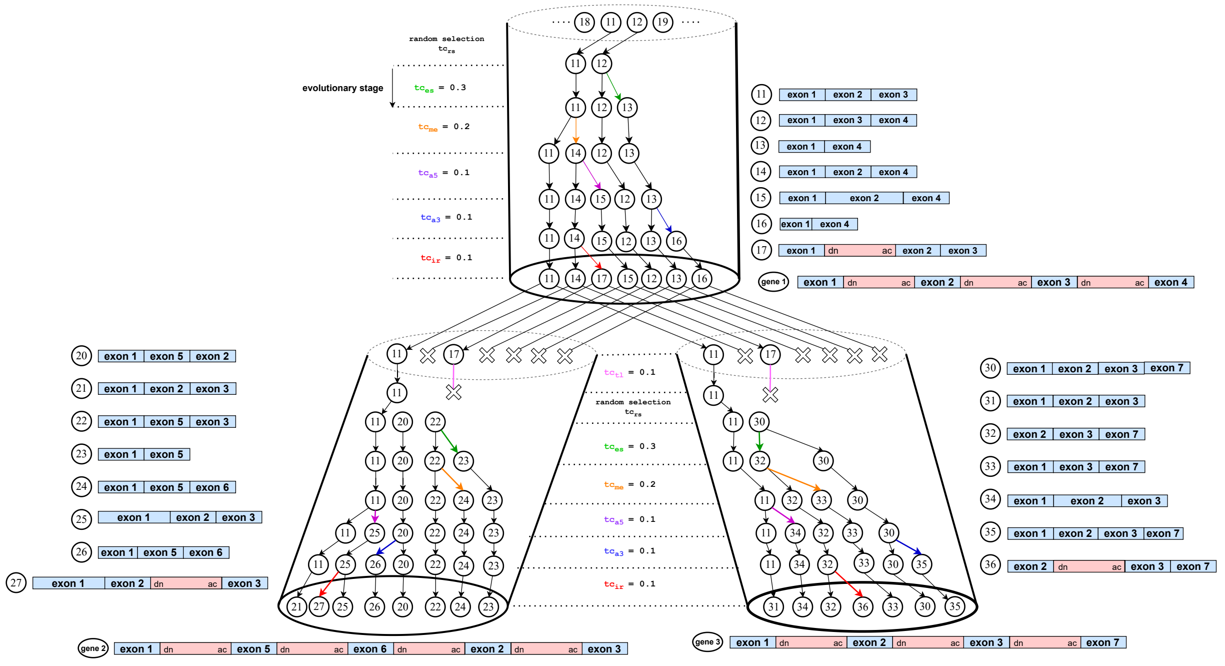


Figure 3: Illustration of the transcript evolution simulation framework. The figure depicts the phylogeny resulting from the simulated evolution of transcripts in a guide gene tree. The guide gene tree depicted as 3 cylinders consists in the evolution of two extant genes, Gene2 and Gene3, from an ancestral gene, Gene1. The bottom surfaces of the cylinders represent the two leaves (Gene2 and Gene3) of the guide gene tree and their ancestor (Gene1). The circles included in each gene represent the transcripts of the gene. The exon-intron structures of each gene is diplayed, as well as the exon composition of each transcript. The evolution history consists of evolutionary stages. The root nodes of the transcript phylogeny correspond to transcript creations. Transcript conservations during an evolutionary stage are illustrated by black directed edges. The colored directed edges illustrate alternative splicing events : green for exon skipping(es), orange for mutually exclusive exons(me), purple for 5' alternative splice site(a5), blue for 3' alternative splice site(a3), and red for intron retention(ir). In addition, a cross sign represents the loss of a transcript. The values of user input parameters are (`tc_rs` $\neq 0.0$, `tc_tl` $= 0.1$, `tc_a5` $= 0.1$, `tc_a3` $= 0.1$, `tc_es` $= 0.2$, `tc_me` $= 0.1$, and `tc_ir` $= 0.1$) and constants factors values (`k_tc` and `c_s_r`) are given such that `k_tc` $\times$ `c_s_r` $= 1$ where `c_s_r` is the length of the branch. For instance, regarding transcripts in Gene2, the number of transcripts undergoing the intron retention event is equal to 1, which corresponds to the ceiling result of `k_tc` $\times$ `c_s_r` $\times\, n \times$ `tc_ir`, where $n = 7$ represents the number of source transcripts at this particular evolutionary stage ({**1#1**, **2#0**, **2#2**, **2#3**, **2#4**, **2#5**, **2#6**}).

## 2.2 Sequence evolution model

Apart from the evolution of the exon-intron structure and the set of transcripts of genes, the sequence of genes also evolves through insertion and deletion (indel) events and substitution events. Numerous methods have been developed for simulating coding and non-coding sequence evolution. In the case of SimSpliceEvol, we opted not to develop a new sequence evolution simulation model. Instead, we utilized existing sequence evolution simulation models, such as indel-Seq-Gen [2], for coding exon and non-coding intron sequences.

### 2.2.1 Exon sequence simulation

The model encompasses both codon substitution and indel processes. Codon substitution events along a branch of the guide tree are simulated for each exon segment, based on the branch length `c_s_r`, which represents the expected number of substitution events per codon on the branch. Each substitution event is generated using an empirical codon substitution matrix[3], which provides the probability of transition between any two types of codons.

Indels are also simulated based on the branch length `c_s_r`. The expected number of indel events per codon on the branch is determined as `k_indel` $\times$ `c_s_r`, where `k_indel` is a user-defined constant. The length of each indel event is drawn from an empirically derived distribution of indel lengths [4]. The codon sequence of an inserted segment is generated using the Markov chains described in Table 1 in [5].

Two user-defined parameters are employed to specify the proportion of insertion and deletion events. The relative frequencies of insertion and deletion events are denoted by `c_i` and `c_d`, respectively, such that `c_i` $+$ `c_d` $= 1.0$.

Deletions are executed before insertions, followed by substitutions.

1. Thus, for instance, the overall expected number of codon deletion events on an exon segment, choose randomnly, composed of $n$ codons is calculated as $n \times$ `c_d` $\times$ `k_indel` $\times$ `c_s_r` for a branch of length`c_s_r`.

2. the overall expected number of codon insertion events on an exon segment, choose randomnly, composed of $n$ codons is calculated as $n \times$ `c_i` $\times$ `k_indel` $\times$ `c_s_r` for a branch of length `c_s_r`

SimSpliceEvol2 emphasizes the indels within the generated multiple alignment, allowing for the assessment of the frequency of indel events in a particular sequence.

### 2.2.2 Intron sequence simulation

The model mirrors the exon sequence evolution model, with the exception that the substitution and indel processes are simulated at the nucleotide level.

**How are the sequences generated?** The length of each exon and intron, along with the associated splice sites for each intron, are sampled. Once the exon-intron structure is established, the nucleotide sequence for each exon and intron is generated using Markov chains constructed from the nucleotide probabilities (Tables 1 and 2 in [5]).

An exon sequence is assembled as a series of independent codons generated sequentially until the length of the exon is attained. Each codon is progressively built using three Markov chains. Initially, a zero-order Markov chain generates the first nucleotide x1 of the codon, followed by the utilization of a first-order Markov chain to generate the second nucleotide x2, and finally, a second-order Markov chain is employed to generate the third nucleotide x3. It's important to note that the model stipulates that the length of an exon must be a multiple of 3.

Conversely, an intron sequence is formed as a sequence of independent nucleotides flanked by splice sites chosen from an empirical splice site distribution, comprising 98% for GT-AG, 1% for GC-AG, and 1% for all other types of splice sites. The nucleotides of an intron are generated sequentially until the length of the intron is achieved.

## 3 Tutorial

asasa

## 4 Possible errors

In this section, we outline several scenarios in which errors may occur while using SimSpliceEvol2. For each of these situations, the program will terminate with an error code. Please don't hesitate to reach out if you need further assistance or encounter any problems. We'll prioritize resolving your issue and then update the manual by including details about this problem and its resolution.

- SimSpliceEvol2 might encounter a situation where it cannot find a transcript different from those already existing in the pool. In such cases, relaunching the simulation may be necessary to resolve the issue.

## References

[1] Y. Christinat and B. M. Moret. "Inferring transcript phylogenies". In: *2011 IEEE International Conference on Bioinformatics and Biomedicine*. IEEE. 2011, pp. 208–215.

[2] C. L. Strope et al. "Biological sequence simulation for testing complex evolutionary hypotheses: indel-Seq-Gen version 2.0". In: *Molecular biology and evolution* 26.11 (2009), pp. 2581–2593.

[3] A. Schneider, G. M. Cannarozzi, and G. H. Gonnet. "Empirical codon substitution matrix". In: *BMC bioinformatics* 6 (2005), pp. 1–7.

[4] M. S. Chang and S. A. Benner. "Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments". In: *Journal of molecular biology* 341.2 (2004), pp. 617–631.

[5] E. Kuitche, S. Jammali, and A. Ouangraoua. "SimSpliceEvol: alternative splicing-aware simulation of biological sequence evolution". In: *BMC bioinformatics* 20 (2019), pp. 1–13.