User Manual SimRA: Simulation based on Random-graphs Algorithms

August 7, 2015

Contents

Chapter 1

SimRA: Simulation based on Random-graphs Algorithms

1.1 Introduction

SimRa is a framework, proposed in [1], for simulating generic and complex evolutionary scenarios of multiple populations with subdivision and admixture. The algorithm generates a network structure, called Ancestral Recombination Graph (ARG), that models coalescence and genetic exchange events (recombinations) with SNP as well as STR polymorphisms.

SimRA is based on backward simulation of the ARG. The details of the algorithm are presented in the paper [1].

If SimRA is used in published analysis, it should be cited as:

A.P. Carrieri, F. Utro, P. Laxmi. Sampling ARG of multiple populations under complex configurations of subdivision and admixture. (submitted 2015).

1.2 Requirements and availability

The backward simulator (SimRA) is provided as a Java executable via command line and has the following dependencies (tested on the versions listed):

- Java (version 1.6+);
- Apache Commons Math library. The jar file `commons-math3-3.5.jar` must be downloaded and stored in the directory `SimRA_lib`. The directory `SimRA_lib` must be located in the same directory of `SimRA.jar`. To obtain the binary `commons-math3-3.5.jar` file, download the release from: `http://commons.apache.org/proper/commons-math/download_math.cgi`.

We have tested the system on Windows, Mac OS X and Linux, SimRA should work on any operating system as long as the dependencies are included.

The zipped folder contain 2 files and 2 directory:

- `SimRA.jar` - executable Java file for the entire SimRA framework;
- `SimRA_lib` - directory where the jar file `commons-math3-3.5.jar` must be located;
- `example_1` and `example_2` are two directories containing two examples of execution of SimRA.

More precisely, in the directory `example_1` there are input and output files for the generation of a scaffold with three populations where two of them are admixed. The directory contains: a README text file describing the command line to execute SimRA for the given scaffold; and input directory and an output directory.

The directory `input` in `example_1` contains the following files:

- *scaffold_2admixed.jpg* - image file representing the scaffold that has been simulated as example. The scaffold is characterized by three contemporary populations (with IDs 1, 2 and 3), where two of them (populations 2 and 3) are admixed;
- *input_scaffold_2admixed.txt* - input file for SimRA defined by the user. The text file describes the structure of the scaffold together with all required parameters;

The directory directory `output` in `example_1` contains the following files:

- *output_scaffold_2admixed_SNP_pop1.txt,*
  output_scaffold_2admixed_SNP_pop2.txt,
  output_scaffold_2admixed_SNP_pop3.txt: three output files, one for each contemporary population. Each txt file contains SNP polymorphisms of the corresponding single contemporary population;
- *output_scaffold_2admixed_STRs_pop1.txt,*
  output_scaffold_2admixed_STRs_pop2.txt,
  output_scaffold_2admixed_STRs_pop3.txt: three output files, one for each contemporary population. Each txt file contains STRs polymorphisms of the corresponding single contemporary population;
- *output_scaffold_2admixed_L.txt*: an example of output file that contains the information about the structure of the scaffold generated by SimRA;
- *output_scaffold_2admixed_S.txt*: an example of output file that contains the information about the non-mixing or completely-linked segments in the extant samples;

- *output_scaffold_2admixed_Scaffold_Description.txt*: an example of output file that contains the detailed description of the scaffold generated by SimRA. In particular for each edge of the scaffold the number of nodes and edges, sample size, SNPs, the number of active lineages at the bottom and at the top of the edge are reported;

- *output_scaffold_2admixed_Scaffold_Cytoscape.txt*: an example of output file that contains the information about each edge in the whole scaffold. This file can be given in input to Cytoscape [2] in order to visualize the ancestral recombination graph;

The directory `example_2` is similar to `example_1` except that it is an example of execution of SimRA to generate the evolution with a single population.

Chapter 2

Command line executable

This chapter describes the use of the Java executable *SimRA.jar* for simulating multiple populations with subdivision and admixture. The input file is a .txt file containing the information about the structure of the scaffold and all the required parameters. The syntax of the input file is described in Sections 2.2. The outputs files generated by SimRA are described in Section 2.3.

2.1 Parameters

The jar file `SimRA.jar` can be executed by command line parameters.

```
java -jar SimRA.jar <whole_path_input_directory> <name_input_file>.txt
<whole_path_output_directory> <name_output_file> [-STR <STRs number>
<state> <μ^STR>]
```

Required parameters:

- `<whole_path_input_directory>`: whole path of the directory when the input file is stored;
- `<name_input_file>.txt`: name for the input file that has to have a ".txt" extension;
- `<whole_path_output_directory>`: whole path of the directory when the output files will be saved. The output directory must be created by the user before executing SimRA;
- `<name_output_file>`: name for the output text files without extension;

Optional parameters:

- `-STR`: it allows to get an output file containing information about STR polymorphisms;
- `STRs number`: number of STR loci - integer;

- `state`: initial state for each STR locus - integer;
- μ^{STR}: STR mutation rate in mut/locus/gen $\times 10^{-4}$ - non negative real number;

In order to get output files containing STR polymorphisms information, the parameter `-STR` and all others parameters between [] brackets (`STRs number, state,` μ^{STR}) must be provided.

To execute `SimRA.jar` it is necessary to reach the directory where the SimRA.jar is located. Some examples of command lines to execute `SimRA.jar` are the following:

- Without STR polymorphisms generation :

```
java -jar SimRA.jar ./ scaffold_2admixed.txt ./ output_scaffold_2admixed
```

This command line allows to store the output files in the same directory where the executable SimRA.jar is located. In this case the input file is located in the same directory of SimRA.jar.

- With STR polymorphisms generation:

```
java -jar SimRA.jar ~/Desktop/inputSimRA/ scaffold_2admixed.txt
~/Desktop/outputSimRA/ output_scaffold_2admixed -STR 40 10 6.9
```

The whole path must be specified if the input/output directories `inputSimRA, outputSimRA` are located in a different directory of the executable SimRA.jar.

2.2 Input file

SimRA takes a text file (".txt") defined by the user that described the structure of the scaffold and all the required parameters.

In the first line the user has to specify the common parameters for all the populations in the scaffold respecting the following syntax.

`g=<segment_length> r=<recombination_rate> mu=<SNPs_mutation_rate>`
where:

- `<segment_length>` is an integer in Kb;
- `<recombination_rate>` is a non negative real number in cM/Mb/gen$\times 10^{-8}$;
- `<SNPs_mutation_rate>` is a non negative real number in mut/bp/gen$\times 10^{-8}$.

An example of first line in the file is the following:

```
g=75 r=0.3 mu=4.0
```

The successive lines describe the contemporary populations. The parameters of each contemporary population (leaf of the scaffold) are specified in a single line. The description of the leaf populations is surrounded by `#begin_actual_populations` and `#end_actual_populations`. The syntax is the following.

```
#begin_actual_populations
[id=<id_population>,m=<sample_size>,N=<population_size>,end_time=<end_time>]
#end_actual_populations
```

where:

- `<id_population>` is an integer that univocally identify the population;
- `<sample_size>` is an integer representing the number of extant units in the contemporary population;
- `<population_size>` is an integer representing the effective population size;
- `<end_time>` is a non negative real number representing the time in which the backward reconstruction of the population has to stop.

An example is the following.

```
#begin_actual_populations
[id=1 m=50 N=10000 end_time=0.730]
[id=2 m=50 N=10000 end_time=0.096]
[id=3 m=50 N=10000 end_time=0.198]
#end_actual_populations
```

The last block of lines in the input file specify how the population merge or split during the evolution. This description is surrounded by `#begin_events` and `#end_events`. Each line in the block specifies the event (merging or splitting). More precisely if the event is a merge of two populations that generates a new ancestral population, the syntax to described the event and the parameters for the resulting ancestral population is:

```
<event_time> merge(<id_firstPopulation>,<id_secondPopulation>) [<id_resultingPopulation>
N=<effective_population_size> end_time=<end_time>]
```

where:

- `<event_time>` is a non negative real number representing the time in which the merging event occurs and corresponding to the time in which the population deriving from the merge starts to be constructed by the backward simulation process;
- `<id_firstPopulation>` is an integer identifying the first population involved in the merging event;
- `<id_secondPopulation>` is an integer identifying the second population involved in the merging event;
- `<id_resultingPopulation>` is an integer identifying the new ancestral population resulting from the merge;
- `<effective_population_size>` is an integer representing the effective population size of the resulting population from the merge;

- `end_time=<end_time>` is a non negative real number representing the time in which the backward reconstruction of the new ancestral population resulting from the merge has to stop.

An example is the following:

```
0.730 merge(1,6) [id=8 N=10000 end_time=1.250]
```

if the event is the splitting of on population that generates two new ancestral populations, the syntax to described the event and the parameters for the resulting ancestral populations is:

```
<event_time> split(<id_PopulationToSplit>) [<id_resultingPopulation_1>
N=<effective_population_size>,end_time=<end_time>] [<id_resultingPopulation_1>
N=<effective_population_size>,end_time=<end_time>]
```

where:

- `<event_time>` is a non negative real number representing the time in which the splitting event occurs and it corresponds to the time in which the two populations deriving from the splitting start to be constructed by the backward simulation process;
- `<id_PopulationToSplit>` is an integer identifying the population that is split at time `<event_time>`.
- `<id_resultingPopulation_1> and <id_resultingPopulation_2>` are two integers identifying respectively the two populations resulting from the splitting;
- `<effective_population_size>` is an integer representing the effective population size of the resulting population (`<id_resultingPopulation_1> or <id_resultingPopulation_2>`);
- `<end_time>` is a non negative real number representing the time in which the backward reconstruction of the new ancestral population (`<id_resultingPopulation_1> or <id_resultingPopulation_2>`) resulting from the splitting has to stop.

An example is the following:

```
0.198 split(3) [id=6 N=10000 end_time= 0.730] [id=7 N=10000 end_time=
0.745]
```

A complete example of input file that described a scaffold with three contemporary populations where two of them are admixed is the following:

```
g=75 r=0.3 mu=4.0

#begin_actual_populations

[id=1 m=50 N=10000 end_time=0.730]

[id=2 m=50 N=10000 end_time=0.096]

[id=3 m=50 N=10000 end_time=0.198]

#end_actual_populations

#begin_events
```

```
0.096 split(2) [id=4 N=10000 end_time=0.745] [id=5 N=10000 end_time=1.250]

0.198 split(3) [id=6 N=10000 end_time= 0.730] [id=7 N=10000 end_time=
0.745]

0.730 merge(1,6) [id=8 N=10000 end_time= 1.250]

0.745 merge(4,7) [id=9 N=10000 end_time=2.150]

1.250 merge(5,8) [id=10 N=10000 end_time=2.150]

2.150 merge(9,10) [id=11 N=10000 end_time=20.0]

#end_events
```

It is possible also to run SimRA to reconstruct just a single population. An example of input file in this case is simply:

```
g=75 r=0.3 mu=2.0

#begin_actual_populations

[id=1 m=30 N=10000 end_time=10.0]

#end_actual_populations

#begin_events

#end_events
```

2.3 Output files

For each contemporary population i in the scaffold SimRA provides two different kinds of text file named:
`<name_output_file>_SNP_popi.txt` ,
`<name_output_file>_STRs_popi.txt`
where `<name_output_file>` is the name chosen by the user for the output files. The file containing the information about STRs are optional. Moreover, SimRA provides four other kinds of text file that are named:
`<name_output_file>_Scaffold_Description.txt`
`<name_output_file>_L.txt`
`<name_output_file>_S.txt`
`<name_output_file>_Cytoscape.txt`

2.3.1 SNP Mutations in SNP.txt file

The `<name_output_file>_SNP_popi.txt` file contains information about SNP mutations for each extant unit in the sample of the contemporary population i. The first line has a number of columns that corresponds to the total number of SNPs in the samples. Each column in the first line is denoted by a number indicating the position of the SNP in the interval $[0, 1]$. All the other lines correspond to the extant units, thus if the sample size is m then the file will contains $m + 1$ lines where the each of the last m lines corresponds to a given extant unit. For each line, each column has value 1 if the corresponding extant unit has the SNP corresponding to the given column, 0 otherwise.

2.3.2 STRs mutations in STRs.txt file

SimRA outputs an optional `<name_output_file>_STRs_popi.txt` file containing information about STRs mutations in the contemporary population i. Each line in the file corresponds to an extant unit in the sample. Hence, the number of lines is equal to the sample size m. Each column corresponds to a given STR locus in the interval $[0, 1]$. Each value in a cell is the number of STRs mutation at the given STR locus for the corresponding extant unit.

2.3.3 Description of the Scaffold in Scaffold_Description.txt file

The `<name_output_file>_Scaffold_Description.txt` file contains the detailed description of the scaffold generated by SimRA together with all the characteristics of each edge.

The first seven lines report the parameters in common among all the populations in the scaffold that are SNP mutation rate, recombination rate and segment length.

```
FIXED PARAMETERS
- SNP mutation rate = 2.0
- recombination rate = 0.3
- segment length = 75
```

where the SNP mutation rate is a non negative real number in mut/bp/gen $\times 10^{-8}$; the recombination rate is a non negative real number in cM/Mb/gen $\times 10^{-8}$; the segment length is and integer in Kb.

The following two lines report respectively the number of edges in the whole scaffold that have more that one SNP and the number of edges with no SNP.

```
Number of edges with more than one SNP mutation = 412
Number of edges with no SNP mutation = 132
```

Moreover, for each population with ID i the following information is reported:

- ID of population: `POPULATION` i ;
- Kind of the population:
  `Population` i `is a LEAF of the scaffold` or
  `Population` i `derives by the SPLITTING of population:` j `at time 0.096` or
  `Population` i `derives by the MERGING of 2 populations:` j `and` l `at time 0.745`;
- total number of edge in the population;
- total number of nodes in the population;
- end time of the population;
- number of remaining active lineages at the end time of the population;
- total number of SNPs in the whole scaffold.

2.3.4 Structure information in L.txt file

The `<name_output_file>_L.txt` file contains the information about the structure of the ARG generated by SimRA.

For each contemporary population there are two different lines:

`0.0 create_pop pop:` i `size: 0`

`0.0 change_size pop:` i `size: 100000`

the first line means that SimRa generated the contemporary population with ID i. In the second line the value that follows `size:` corresponds to the effective population size N of the population i given in input by the user (in this example $N = 100000$).

For each contemporary population i, if the sample size specified by the user is m_i, then m_i lines corresponds to the m_i extant units of population i.

These are in the form:

`0.0 ADD node:` j `pop:` i

where j is the ID of the leaf node and i is the ID of the contemporary. The first value `0.0` in each of these lines means that each leaf node has associated time equal to 0.0. An example is the following:

```
0.0 ADD node: 0 pop: 1
0.0 ADD node: 1 pop: 1
0.0 ADD node: 2 pop: 1
```

and so on till the leaf node with ID $m - 1$.

All other lines in the file correspond to all nodes in the whole scaffold that are not extant units. All these nodes are ordered by the time they has been created and have global IDs based on this ascending ordering.

0.096 H split pop 2 in 4 and pop 5

More precisely, given a node n, if n is a coalescent node, there will be a line in the .txt file containing the following columns:

```
<time of the node> C <ID_left_son> <ID_right_son> -> <ID_node_n> pop:
<ID_population>
```

where <ID_left_son> <ID_right_son> are respectively the IDs of right and left son nodes of n, while <ID_node_n> is the ID of the node n.

An example relative to the node n with ID 20 in population 1 is the following:

```
3.8453186770370504E-5 C 4 19 -> 20 pop: 1
```

meaning that the node 20 is a coalescence node of 4 and 19 in population 1 and it has been generated at time $3.8453186770370504E - 5$.

If the node n is a recombination node, there will be two lines associated to n. The first line will be in the following form:

```
<time of the node> Y <ID_son> -> <ID_node_n> pop: <ID_population>
```

where <ID_son> is the ID of the son node of n and <ID_node_n> is the ID of n, as before.

An example is the following:

```
1.8451435106555335E-4 Y 13 -> 21 pop:  1
```

The second line, associated with the recombination node n as well, will be in the following form:

```
<time of the node> R <ID_node_n> -> <ID_left_parent> <ID_right_parent>
<ID_population> <split_point>
```

where <ID_left_parent> <ID_right_parent> are respectively the parent nodes of n, while <split_point> is a real value in $[0, 1]$ corresponding to the split point in the segment stored in the node n. An example is the following:

```
1.8451435106555335E-4 R 21 -> 40 33 1 0.618806344640725
```

2.3.5 Detailed SNP information in S.txt file

The `<name_output_file>_S.txt` file contains the information about the non-mixing or completely-linked segments in the extant samples. Thanks to these information it is possible construct the embedded trees that are in the ARG [3].

The lines in the file describe the contemporary population specifying the number of extant units for each of those. An example is the following:

```
paramfile:  params Pop 1 50 Pop 2 50 Pop 3 50
```

The other lines are:

```
paramfile:  params
params:  length <segment_length> mu <mutation_rate>
L <segment_length>
pos anc1 anc2 freq nodes...
```

where <sample_size> is the number m of extant units, <segment_length> is the length g of the segment and <mutation_rate> is the SNPs mutation rate. An example is the following:

params: length 75000.0 mu 1.0E-9

L 75000.0

```
pos anc1 anc2 freq nodes...
```

The file continues with the SNPs information for each un-mixing unit in the whole scaffold. Note that all the nodes in the whole scaffold are global IDs assigned in ascending ordered based on the time the nodes are created during the backward reconstruction process.

For each un-mixing unit, there is a header line indicated by >, as:

```
> [<start_pos>,<solid_length>] time 1 E[muts] = #mut (<SNPs_number>)
```

<start_pos> indicates the start position of the solid in the segment and <solid_length> indicates its length that is computed as <solid_length> = <start_pos> − <end_pos>, where <end_pos> is the end position of the solid in the segment. <SNPs_number> is the number of SNPs in the segment. An example is the following:

```
> [0.0,0.0030186525399157195] time 1 E[muts] = #mut (1)
```

In each segment, there is a line (tab separated) for each mutation, as:

`M <SNP_position> <father_node> <son_node> <leaves_number> <list_IDs_leaves>.`

The line is the composed as follows: the character `M` for mutation, the position of the mutation (`<SNP_position>`), the edge is indicated by the two nodes `<father_node>` `<son_node>`, the number of samples reached and the list of samples (separated by comma). An example is the following:

```
M 0.007110760691784903 3012 2997 20 0 1 2 3 4 5 6 7 8 9
```

2.3.6 Information about the edges in Cytoscape.txt file

The `<name_output_file>_Cytoscape.txt` file contains the information about each edge in the whole ARG corresponding to the scaffold. This file can be given in input to Cytoscape [2] in order to visualize the structure.

For each edge in each ARG/population of the scaffold there is a line that respects the following syntax:

`ID_node_father` <segment_intervals> `ID_node_son`

where `ID_node_father` is the ID of the father node in the edge, <segment_intervals> is the list of intervals carried by the edge and `ID_node_son` is the ID of the son node.

Some examples are the following:

```
75 [0.0,1.0] 74
71 [0.0,0.10834532925294793] 68
```

Bibliography

[1] A.P. Carrieri, F. Utro, and L. Parida. Sampling ARG of multiple populations under complex configurations of subdivision and admixture. *submitted at Bioinformatics*, 2015.

[2] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–504, 2003.

[3] F. Utro, M. Pybus, and L. Parida. Sum of parts is greater than the whole: inference of common genetic history of populations. *BMC Genomics*, 14:S10, 2013.