Manual to AnA-FiTS

Andre J. Aberer

December 27, 2012

This is the manual to AnA-FiTS, the ancestry-aware forward-in-time simulator. AnA-FiTS simulates genetic sequences of a population forward in time under a Fisher-Wright model with selection and dynamically changing population size. Currently, the feature set is not particularly extensive: the development model is to implement advanced features on demand. Instead, AnA-FiTS primarily focuses on runtime speed and to the best of my knowledge outperforms competing software by one to two order(s) of magnitude. In addition to a `ms`-like sequence output, AnA-FiTS provides the entire observable history of the sequences' neutral part (see Sect. 2), a structure similar to the ancestrial recombination graph.

Contents

1 Installation

For the installation of AnA-FiTS, please use a fairly recent `g++` compiler ($\geq$ 4.6). Other compilers may work, but have not been tested.

Another requirement is the **boost-library**. In specific, only the `ProgramOptions` package is required. On most ubuntu/debian system, installing boost is as simple as

```
$  sudo apt-get install libboost-all-dev
```

Otherwise just visit `http://www.boost.org/`. You may encounter problems with old boost-versions. If the program fails to compile with your version of boost, comment the line `#define PRODUCTIVE` in `src/config.hpp`.

Finally, an installation of python may be required (however this is not mandatory).

For compiling `AnA-FiTS`, enter

```
$  make
```

on the command line. An executable name `AnA-FiTS-git-version` is created. The *git-version* suffix in the name of the executable helps you (and me) to exactly identify which version of the code was used (in case of bugs or for reproducibility).

An executable `convertSeq` and `convertGraph` should be produced by your `make` call. This tiny tool converts the binary output of AnA-FiTS into human-readable output.

2 Quick Start

To see AnA-Fits in action, just enter

```
$  ./AnA-FiTS-v0.9-1-gfc23a8c -n myRun -s 3
```

on the command line. With two mandatory command line arguments you provide a run id (`-n`) and a seed for random number generation (`-s`). All further simulation parameters are set to default values: in the case above, we simulate a chromosome (with length 10 Mbp) for an effective population size (N_e) of 500 diploid individuals and a per-site per-sequence recombination/mutation rate of 1e-8. 10 % of the mutations simulated will be under selection (with parameter `-W =2 0.05 10 2000 0.05 1 500`, see Sect. 5 for details).

2.1 Output

AnA-FiTS produces three output files. The info file contains all information relevant to the run. In particular, this makes it possible for you to re-create the run later.

The file anafits_polymorphisms.myRun contains a `ms`-like sequence representation in binary format. This means that before having a look at the sequences data, you have to convert this file first calling:

```
$  ./convertSeq anafits_polymorphisms.myRun
```

or

```
$  ./convertSeq myOutput anafits_polymorphisms.myRun
```

For the first call the output is written to your console, in the second case, you create a file with name myOutput.

The rationale behind the binary format is runtime speed. Formatting the output can take up a considerable part of the total execution time. Note, that this is a feature: if you want to run a huge number of simulations, you can omit `convertSeq` and directly parse the binary output in your application. For instructions on how to parse the binary format, please write me an e-mail or try to extract the information from src/sequenceConversion.cpp.

Finally, AnA-FiTS also creates a graph file containing the observable history of the neutral part of the sequence. This is a by-product of the algorithm implemented to speed up simulation of neutral mutations, however it also is interesting information on its own. The graph file is in binary as well and can be converted with

```
$ ./convertGraph ./anafits_graph.myRun
```

2.2 Format

The format for the polymorphism file is as follows:

- line 1: chromosome id
- line 2: mutations that are fixed in the population (neutral and non-neutral, unordered). Mutations are separated by semi-colon. For each mutation, AnA-FiTS prints the
  - location in the sequence
  - generation of origin (starting with 0)
  - selection coefficient
  - base (if multiple mutations at the same position occurred, each mutation is represented as a separate column in the matrix)
- line 3: contains polymorphic mutations in the same format as in line 2. Each mutation corresponds to one column in the matrix below.
- line 4 to beginning of next chromosome or end: contains a matrix representing the sequences. Each column is a polymorphic mutation, each row stands for a sequence. Adjacent rows (e.g., 0 and 1, 2 and 3) are the two sequences for a diploid individual.

In rare instances, two distinct mutation events may lead to the same polymorphism or a back-and-forth mutation (e.g., $A \to C \to A$) leads back to the original state. These instances are adequately handled in AnA-FiTS, so we implement a true finite-sites model. Since we do not want to loose information in the output, both cases listed above yield additional columns in the polymorphism output. In case 1, we have two adjacent columns with the same mutation (i.e., base, location is equal), where some haplotypes have a 1 in one column and some in the other (none of them can have both). In the second case, haplotypes with a back-and-forth mutation will appear to have both the mutation ($\to$ C) as well as the back-mutation ($\to$ A). So keep in mind that for these rare instances the output does not directly represent the genotype (instead your script has to compensate for that, but on the plus side may use this additional information).

The format for the graph file is (see Sect 3 for further information on the graph):

- line 1: graph id (each chromosome produces a distinct graph, thus for organisms with two chromosomes, you will obtain two graphs)
- line 2: ids of nodes that survived into the present generation
- line 3 to end (resp. start of next distinct chromosome): node and edge information of the graph. Each entry is composed of:
  - id: the node id
  - node description (in brackets): for neutral mutations, this is the location in the sequence, the generation of origin and the base, for recombinations this identifies the break point and the generation of origin
  - edge information: id(s) of node(s) *parent* haplotypes (two parents for a recombination, one for mutations)

3 The Graph

For information on how the graph can be used / interpreted, please refer to the supplement of the main paper. Auxiliary information will follow in this section in a future version.

4 Important Information about Memory / Runtime

4.1 Restricting Memory for Ancestry

For the most part, fast execution times of AnA-FiTS come at the cost of highly increased memory consumption. You may find AnA-FiTS exceeding your desktops main memory,

when simulating more than 10,000 individuals. You can significantly relax memory requirements with the `-M` switch, for instance with

```
$  ./AnA-FiTS-v0.9-1-gfc23a8c -n myRun -s 3 -M 2G
```

you advise AnA-FiTS not to use more than 2 GB of main memory. This, however, is a soft constraint: for some calls, AnA-Fits will allocate more memory nonetheless (thus, some experimentation with the parameter may be necessary). If less memory is used, the total execution time will definitely increase, however in many cases the performance penalty is not severe.

Please note, if the `-M` is used, the same random number seed will not result in the exact same run for different values for `-M`.

4.2 Restricting Memory for Sequence Creation

Another tuning parameter that can be used in order to save memory is `-R` *num*. The parameter influences, how much memory is used after the BEG graph has been created (see supplementary). In specific, it sets the number of references needed for explicitly representing nodes in the graph. If no nodes are represented explicitly at all (e.g. with `-R` 1000), runtimes for AnA-FiTS will significantly go up. On the other side, the more nodes are represented, the more memory is needed. Thus, this step can become prohibitive memory-wise.

By default, AnA-FiTS sets this parameter automatically, such that not more than 5 % of the nodes of a graph are represented explicitly. If you happen to have a particularly large graph (because of high numbers of individuals and/or high rates), you may want to adjust this parameter (i.e., increase it to for instance `-R` 10), such that AnA-FiTS consumes less memory at this stage (while the runtime often is not severely increased). Note that this option does not influence the simulation and you should always obtain the exact same result, if you use different values of -R, but keep all further parameters constant (if this is not the case, please report this as a bug).

5 Information on Program Options

For most parameters, there is a short and long version, I will switch between both possibilities.

5.1 General

Input parameters for AnA-Fits are not scaled. For computing for instance how many recombinations we expected for a given chromosome per generation ($E[REC]$), the input value for recRate r is computed as follows:

$$E[REC] = r * \text{sequenceLength} * 2 * N_e.$$

5.2 Length of Simulation

Usually, in forward simulation, we simulate for $5 \cdot 2 \cdot N_e$ **number of generation**s (default value for AnA-FiTS). You can influence simulation length with `--SIM`.

5.3 Population Size

Initial population size (number of diploid individuals, N_e) is provided with `-N`. Note that in all forward simulation, this parameter is particularly expensive (since it also increases the total number of generation you have simulate). You can change the population size during simulation with the `--popEvent` options. Multiple popEvents are possible, just provide the parameter with options multiple times. The general format is `--popEvent <mode> <time> <args>...`, where

- *time* is the absolute generation number (e.g. generation 200)
- *mode* is one of the following:
  - c, with argument r. A discrete size change with rate $\frac{\text{new}}{\text{old}}$. If you want to model a spontaneous population reduction by factor 2 in generation 100, provide `--popEvent c 100 0.5`.
  - d, exponential decay with rate r. Note that, AnA-FiTS rounds to the next even number of individuals.
  - g, exponential growth with rate r (analogously to d)
  - k, logistic growth with rate r until the carrying capacity k is reached

Another example, if you want to undergo 500 individuals a bottleneck in generation 500 that wipes out half of the generation and then continue with logistic growth with rate 0.001 in generation 600 until 2,000 individuals are reached (however simulation ends before this is the case), enter:

```
$  ./AnA-FiTS-v0.9-1-gfc23a8c -T c 500 0.5 -T k 600 2000 0.001 -n run2 -s 3
```

5.4 Multiple Chromosomes

You can simulate **multiple chromosomes** (resp. unlinked loci) with `-L`. Just provide for each the chromosome the length

```
$  ./AnA-FiTS-v0.9-1-gfc23a8c -L 1000000 1000000 1000000 1000000 1000000 -n run3 -s 3
```

In the above call, 5 chromosomes, each of length 1 Mbp are simulated. For each chromosome a separate section in the graph and polymorphism file will be created. Elements in both files are in the corresponding order: sequence number 1 in locus 1 occurs in the same individual of the present generation as sequence 1 in locus 2; the first surviving node in graph 1 belongs to the same individual as the first surviving node in graph 2.

5.5 Selection Model

For simulating a neutral population, use `-w`. Otherwise, you can allow for non-neutral mutations with `-W`. For runtime, the most important thing is the fraction of mutations that are under selection. The following modes are available:

- 1 *coef* p_{pos} p_{neg}
  each mutation is assign a fixed selection coefficient. The sign of *coef* is positive with probability p_{pos} (i.e. the mutation is deleterious) or negative with probability p_{neg} (beneficial mutation).
- 2 p_{pos} α_1 β_1 p_{neg} α_2 β_2
  mixture of two Γ distributions: with probability p_{pos} a beneficial selection coefficient is drawn from $\Gamma(\alpha_1, \beta_1)$ and with probability p_{neg} a deleterious selection coefficient is drawn from $\Gamma(\alpha_2, \beta_2)$.
- 3 p_{sel} μ σ
  a mutation is non-neutral with probability p_{sel} and the coefficient is drawn from a normal distribution with a *positive* mean μ and standard deviation σ.
- 4 p_{sel} μ σ
  a normal distribution as in 3, however mean mu is negative this time. This means, instead of `-W 3 0.1 -0.001 0.001` you have to specify `-W 4 0.1 0.001 0.001`. This sub-optimal work-around has to be used because of a deficiency in the boost program options parser.

For this software, the fitness effect of a mutation is defined as $f := 1 + s$. Thus, a positive selection coefficient induces positive selection, a selection coefficient $\in [-1, 0)$ means purifying/deleterious selection.

So, if for instance we call

```
$  ./AnA-FiTS-v0.9-1-gfc23a8c -W 2 0.05 10 2000 0.05 1 500 -n run4 -s 3
```

then 10 % of all mutations are under selection. So effectively, the number of mutations that we have to simulate forward in time is reduced by a factor of 10. The remaining 90% of mutations will be simulated using the graph based algorithm as described in the main paper.

6 Validation

For various cases, we compared summary statistics of data simulated under AnA-FiTS to summary statistics simulated under either ms (neutrality) or SFS_CODE. You can reproduce these runs employing the `./utils/validate.py` script. If you encounter a deviation of sequences simulated under AnA-FiTS from the expectation, please report this as a bug.

7 Frequently Asked Questions

AnA-FiTS outputs the entire population, why not sample? This is a feature that will be implemented at some point in the future (will yield further runtime improvements). For now, please sample on your own or use a quick python script like this:
`$ ./convertSeq` *`ana-poly`* `| ./utils/samplePopulation.py` *`number sampleIndi`*
Note that this script merely samples haplotypes (not individuals), however it is consistent when used with multi-chromosome datasets (call w/o arguments for help).

What should I do, if I found a bug? Please report bugs either via the github ticket system or to andre *dot* aberer *at* h-its *dot* org.

8 Acknowledgments

I want to give credit to two great implementations, on which AnA-FiTS builds:

- libasmlib by Agner Fog (`http://www.agner.org`): I found it amazing that there is still vectorized integer division provided by Intel/AMD. Thanks to Agner for that.
- RandomLib by Charles Karney (`http://randomlib.sourceforge.net/`): A well-crafted and extremely fast random number generator. I found the source code to be very interesting to read.

AnA-FiTS comes with both libraries included for user convenience. If you find the libraries to be outdated, please update from the respective site (resp. please inform me about that).

9 Citation and License

If AnA-FiTS was useful for your scientific work, please cite as

- Andre J. Aberer, Alexandros Stamatakis. AnA-FiTS: Rapid Forward Genome Simulation With Ancestries. 2012. *unpublished.*

Note that, AnA-FiTS is released under the GNU general public license version 3. Refer to LICENSE.txt for further information.