FITS Documentation

Release 1.3.3

Tal Zinger

Jun 10, 2019

Contents

FITS (Flexible Inference from Time-Series data) is described in a paper published in Virus Evolution:

Tal Zinger, Maoz Gelbart, Danielle Miller, Pleuni S Pennings, Adi Stern, **Inferring population genetics parameters of evolving viruses using time-series data**, *Virus Evolution*, Volume 5, Issue 1, January 2019, vez011, https://doi.org/10.1093/ve/vez011

FITS comes in two distributions:

- **Graphical user interface (GUI)** distribution for Windows and MacOS
- **Command line interface (CLI)** distribution for Linux, Windows and MacOS

CHAPTER 1

Before you start

1.1 Downloading and installing FITS

FITS is ready to run out of the box (i.e. provided as pre-compiled binaries) for Windows, macOS and Linux. All versions are available at https://github.com/SternLabTAU/FITS/releases.

The command line version (**CLI**) is identified by file names beginning with **fits**X.X.X, and the graphical version (**GUI**) is identified by file names beginning with **fits_gui**X.X.X (X.X.X being the version of FITS).

Download the appropriate archive (e.g. fits_gui1.3.3_MacOS.tar.gz for the GUI version compiled under MacOS), and extract all files from the archive to a folder of your choice.

Windows: Right-click on the downloaded archive, and choose "Extract all". Choose a target folder, and press the "extract" button, then "finish". For the **CLI** - open a command prompt window and cd to the created folder, then run fits. For the **GUI** - simply run fits_gui.exe. Depending on your security settings, Windows may warn you that fits_gui.exe can't be run because it is from an unidentified developer. Choose to run it anyway. FITS should now be fully operable.

Mac: Double-click on the downloaded archive within Finder. Archive Utility should create a folder with a name matching the archive (e.g. fits_gui1.3.3_MacOS for fits_gui1.3.3_MacOS.tar.gz). For the **CLI** - open a Terminal window and cd to the created folder, then run fits. For the **GUI** - run fits_gui.app. Depending on your security settings, macOS may warn you that "fits_gui.app can't be opened because it is from an unidentified developer". To override this warning, go to System Preferences Security & Privacy General and press the "Open Anyway" button, the "Open". FITS should now be fully operable.

Linux: For the **CLI** - Extract the downloaded archive, and run ./fits. Because FITS for Linux open a command prompt window and cd to the created folder, then run fits. If you get a error saying "GLIBCXX was not found", you will need to compile it from source on your system (see below).

1.2 Compiling from source

The most recent version of FITS is available here. In order to compile, FITS requires the Boost library 1.69 and a C++11 supporting compiler. We used gcc 8.2 on Linux (Centos), Clang provided with Xcode9 on MacOS (High

Sierra) and MinGW provided with Qt 5.12 on Windows 10 & 7.

Note: Before you start, make sure GCC is in the `PATH`.

1.2.1 Compiling the command line interface

1. Download and extract the Boost library 1.69 to a convenient location.
2. Compile all *.cpp files, referring the compiler to the Boost libraries, e.g.: `g++ -std=c++11 -O3 -o fits1.3.2 -I/path/to/boost/ -L/path/to/boost/libs *.cpp`

   (here we tell gcc to use c++11 standard, use optimization (-O3) and name the output file (-o) **fits1.3.2**).
3. Run FITS with no command line arguments to get the help text
4. Run FITS with the proper syntax (see *Using the command line interface*) in order to generate data or infer the required parameter

1.2.2 Compiling the graphical user interface

In order to compile the GUI, you will need (in addition to Boost) the Qt framework, along with Qt Creator. Both are available in open source license here.

Note: When installing the Qt framework, make sure you also install the included MinGW compiler.

1. Extract the source code zip folder in your favorite location.
2. Open **Qt Creator**.
3. Click **File>Open file or project** and locate the project file (**fits_gui.pro**)
4. Within the project file, replace placeholder text next to `__INCLUDEPATH__` with the path to the boost library
5. Click "fits_gui" in the left toolbox and make sure the configuration is set to **Release**.
6. Click **Build>Build All**. After build completion, FITS executable will be found in the Build directory (you can see where it is under **Projects>Build directory** that is available from the left toolbar.
7. Move fits_gui.exe to a new folder. Open the console and navigate to that folder.
8. - For **windows**: Locate windeployqt.exe under bin directory in the Qt installation folder. Run windeployqt.exe with fits_gui.exe as its sole argument: `path/to/windeployqt.exe fits_gui.exe`.
   - For **MacOS**: Locate macdeployqt.exe under bin directory in the Qt installation folder. Run macdeployqt with fits_gui as its sole argument: `path/to/macdeployqt fits_gui`.

1.3 FITS input

FITS requires two types of input: *Data file* and *Parameters file*.

1.3.1 Data file

This file is expected to hold observed allele information from the system under study. FITS expects a tab-delimited textual file, with following columns:

1. `gen` for the generation of the observation
2. `allele` for the observed state
3. `freq` for the measured frequency for that state
4. `position` for the position number for which the frequency data is given (optional)

Note: FITS assumes the columns to appear in the above order.

Note: The allele with the highest frequency at the first available time point will be defined as WT (w=1).

Table 1: Example data file

gen	allele	freq	position
0	0	1	1
0	1	0	1
1	0	1	1
1	1	0	1
2	0	0.99999	1
2	1	1e-05	1
3	0	0.9999899998	1
3	1	1.00002e-05	1
4	0	0.9999899998	1
4	1	1.00002e-05	1
5	0	0.9999600016	1
5	1	3.99984e-05	1

You can also download an `example`.

Note: For each generation, the sum of frequencies for the different alleles should be 1.

Note: FITS accepts allele frequencies at a given loci. Sequencing techniques tend to vary in their accuracy, so sometimes the provided allele frequencies may be inaccurate. If using inaccurate input, FITS inferences may be inaccurate as well. Specific examples include:

1. Inference of fitness of highly deleterious mutations where the accuracy threshold of sequencing is worse than the mutation rate.
2. Inference of mutation rate from neutral alleles when the number of generations X the mutation rate is lower than the accuracy threshold of the sequencing.
3. Inference of mutation rate or fitness when very shallow sequencing is available (due to limited sampling or limited sequence coverage).

1.3.2 Parameters file

This file provides FITS with population genetics parameters information of the system under study. Each line in this file represents a different parameter to set, where a space exists between the name of the parameter and its value: `<parameter_name> <parameter value>`.

Note: If you want to put comments within the parameters file, just add # at the beginning of the comments' lines.

You can also download an `example`.

General parameters

Parameter name	Type	Description
N	Integer	Size of population
sample_size	Integer	Size of observed population (e.g., sequenced genomes)
bottleneck_size	Integer	Size of the population transferred on a bottleneck event
bottleneck_interval*	Integer	Number of generations separating between bottleneck events (default: 0)
num_alleles	Integer	Number of alleles observed in all loci
mutation_rateX_Y	Float	Rate of mutation of allele X to allele Y. Not required if mutation rate is to be inferred
fitness_alleleX	Float	Fitness value assigned to allele X. Not required if fitness is to be inferred
logistic_growth*	Float	1: model the population growth throughout the generations with a logistic growth model (default: 0)
logistic_growth_K	Float	Logistic model - upper bound
logistic_growth_r	Float	Logistic model - proportionality constant

*parameter value of 0 means disabled/off; positive values mean enabled/on.

ABC parameters

Parameter name	Type	Description
num_samples_from_prior	Integer	How many simulations to perform
acceptance_rate	Float	Fraction of best simulations to utilize for the inference of the parameter.

Single simulation

Parameter name	Type	Description
num_generations	Integer	Number of generations to simulate
init_freq_alleleX	Float	Initial frequency of allele X

Fitness inference parameters

Parameter name	Type	Description
fitness_prior	Text	One of the following: uniform (for Uniform distribution) log_normal (based on Bons et al. 2018) fitness_composite smoothed_composite (default) See the distribution of the above priors on a (0,2) fitness here
min_fitness_alleleX	Float	The minimum fitness value (inclusive) that may be assigned to allele X
max_fitness_alleleX	Float	The maximum fitness value (exclusive) that may be assigned to allele X

Mutation rate inference parameters

X and Y are alleles defined in the data file (i.e., 0 and 1).

Parameter name	Type	Description
min_log_mutation_rateX_Y	Float	Minimum (inclusive) n for mutation rate 10^n from alleleX to allele Y
max_log_mutation_rateX_Y	Float	Maximum (exclusive) n for mutation rate 10^n from alleleX to allele Y

Population size inference parameters

Parameter name	Type	Description
Nlog_min	Float	Minimum (inclusive) exponent n for population size 10^n
Nlog_max	Float	Maximum (exclusive) exponent n for population size 10^n

1.4 FITS output

1.4.1 General

FITS infers population genetics parameters using the Approximate Bayesian Computation (ABC) method. The output of this method is a distribution of values that explain the observed allele frequencies with the highest probabilities (also called *the posterior distribution*). A common practice is to take the **median** of this distribution as the inferred value of the parameter under study.

The results below are outputted for all inferences.

Result header	Description
median	The median value of the posterior distribution.
MAD	Median Absolute Deviation index (MAD) of the posterior distribution.
min	The minimum value in the posterior distribution.
max	the maximum value in the posterior distribution.
pval	The result of a Levene's test, comparing between the prior and posterior distributions. Successful inference process should yield a posterior distribution that is significantly different from the prior.

Note: If the p-value for Levene's test is not significant (>=0.05) or Nμ is small (<1), FITS will mark the relevant line in the results with an asterisk (*). This result is considered **unreliable**.

1.4.2 Fitness inference results

In addition to the *General* reported values, in fitness inference more data are available:

Result header	Description
allele	The allele for which the results are reported
DEL(%)	The proportion of the posterior distribution with values below 1.
NEU(%)	The proportion of the posterior distribution with values equal to 1.
ADV(%)	The proportion of the posterior distribution with values above 1.
category	A possible classification of the allele into {LETHAL,DEL,NEU,ADV}, based on the inferred fitness value.

Note: Some fitness priors rarely choose the exact value of 1 and therefore **NEU(%)** will approach zero, even for neutral alleles.

1.4.3 Mutation rate inference results

FITS infers the mutation rates between all defined alleles. Accordingly, the output table contains the target allele in the first row and the source allele in the first column.

Note: In Evolve & Resequence (E&R) studies, when the population is homogeneous at first generation, in the absence of more information the inference of the rates between the minor allele to the major will be insignificant, so the **pval** should be taken into account.

1.4.4 Population size inference results

See the *General* inference results.

CHAPTER 2

Using FITS

2.1 Using the graphical interface

Note: On Windows, please extract all files from the downloaded archive, and only then run the (extracted) executable. Running the program directly from the compressed archive will not work.

After opening FITS, the following screen will be visible:

Click the `Browse...` button near the `Parameters` label to load a parameters file (example *Parameters file*).

Note: The loaded parameters may be viewed using the `View` button.

From the given parameters, FITS will automatically identify the possible inference mode (in the example below, Fitness inference mode).

To load the data file, click `Browse...` near the `Data` label just below the `Parameters` label. Locate and select the *Data file*.

Note: FITS expects the data file to be tab-delimited. If using Office Excel, save your worksheet as `tab delimited` file. Verify the content and the format of the file if FITS fails to run.

Within the `Actions` area, FITS will automatically suggest available actions according to the parameters available in the parameters file. Press `Go!` to perform the selected action. FITS will show a progress bar and estimated time to completion.

The inference results are given in the `Output` area. It may be copied to the clipboard (for example, to be pasted into a spreadsheet). Inference output, prior and posterior distributions may be exported to text files.

The inference results are explained in the *FITS output* page.

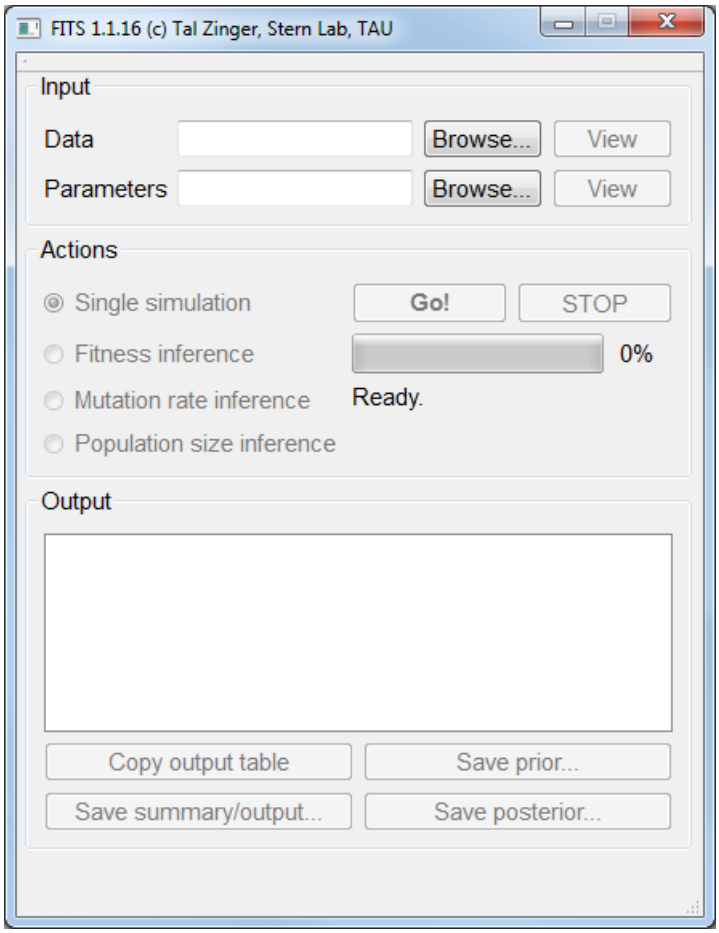

Fig. 1: FITS main screen.

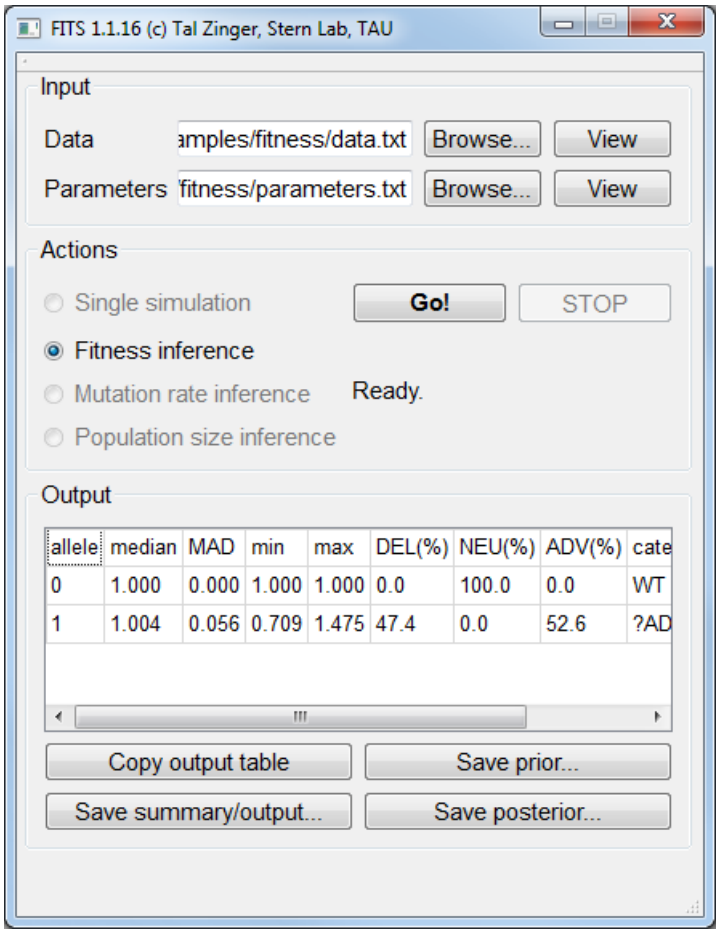

Fig. 2: FITS after making an inference.

2.2 Using the command line interface

Running fits with no parameters prints the help screen to the console, listing possible usage syntaxes. For fitness inference, as an example, the syntax is:

```
fits -fitness <param_file> <actual_data_file> <posterior_file> <summary_file>␣
↪(optional: <prior_file>)
```

2.3 Use cases

2.3.1 Fitness inference

In Evolve & Resequence (E&R) studies, a population is grown for a period of time under a given condition and sampled at several time points. The frequencies of genetic variants or phenotypes for the different time points are measured, and we'd like to infer the fitness that is associated with each specific variant (or phenotype). An example for such frequency data, sampled for 15 generations and determined for frequency is described here:

The size of the population is estimated to be 100,000. Therefore the parameter `N 100000` was set.

The mutation rate is estimated to 1:100,000 (or 10^{-5}). Therefore the parameter `mutation_rate0_1 1e-05` was set.

We expect the fitness values of the phenomena to be between 0 (the minimum possible fitness value) and 2 (very adaptive fitness). Therefore the parameter `min_fitness_allele1 0.0` was set, to indicate zero minimal expected fitness and `max_fitness_allele1 2.0` as well, to indicate the maximal possible fitness value of two.

The prior we chose for this analysis was smoothed_composite, a prior that is built towards typical fitness landscapes. Therefore the parameter `fitness_prior smoothed_composite` was set.

We want the ABC framework to perform 100,000 simulations, and accept the fitness value from the best 1,000 simulations. Therefore the parameter `num_samples_from_prior 100000` was set, to indicate 100,000 simulations, and the parameter `acceptance_rate 0.01` was set, to indicate that the top 1% simulations will be used to decide on the fitness value of this allele.

The data file for a simulated neutral allele (fitness of 1) under a populations size of 10^5 and a mutation rate of 10^{-5} is available `here`. The corresponding parameters file is available `here`.

The inferred fitness value by FITS was practically 1:

2.3.2 Mutation rate inference

A common problem is the inference of the rates of mutations between two (or more) alleles. FITS supports such inference by harnessing prior knowledge about the fitness of the mutant allele(s) (say, from competition essay) and and the size of the population. A particular example for a case where such inference can be highly accurate is the usage of frequencies of multiple positions with equal fitness. In many biological entities synonyomus mutations

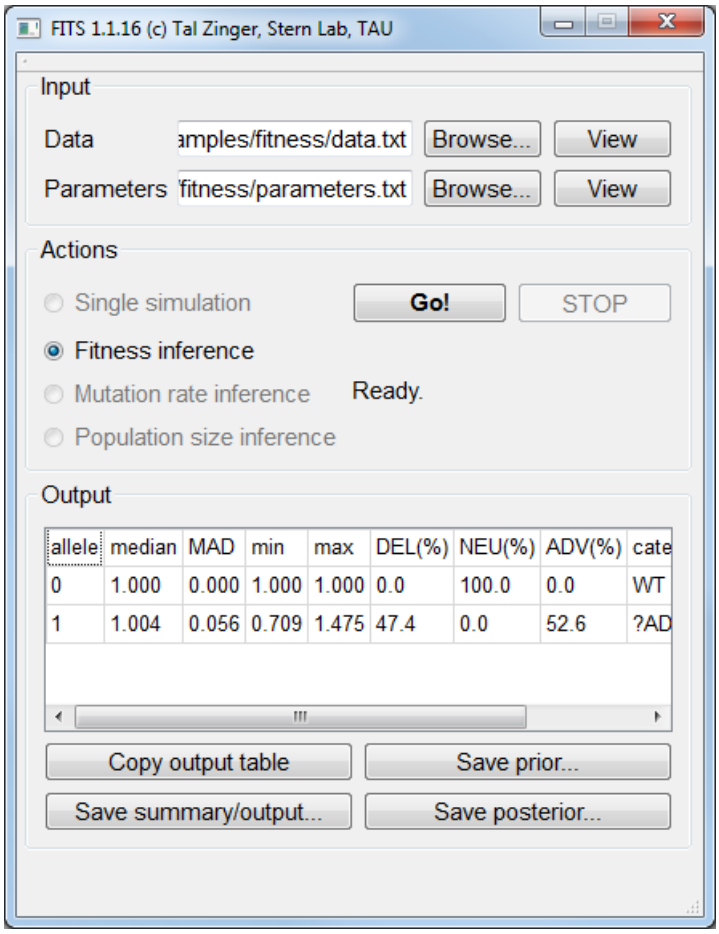

Fig. 3: FITS inferred a fitness value of 1.004 for a simulated neutral allele.

approach neutrality and therefore may be used for mutation rate inference. In this example we'll highlight how this can be done. We simulated 10 independent loci using fitness value of 1, population size of 100,000 and mutation rate of 10^{-5} and measured their frequencies for 15 generations.

The minimum and maximum considerable mutations rates should be provided within the parameters file, using the log value. For this example, we use considerable mutation rates between 10^{-7} and 10^{-3}, which will be defined between the wildtype allele (0) and the mutant allele (1) and vice-versa. For providing the minimal log mutation rate between the wildtype allele and the mutant we set `min_log_mutation_rate0_1 -7` and its reciprocal `min_log_mutation_rate1_0 -7`. For providing the maximal log mutation rate between the wildtype allele and the mutant we set `max_log_mutation_rate0_1 -3` and its reciprocal `max_log_mutation_rate1_0 -3`.

We used neutral alleles and therefore set the wildtype and mutant alleles' fitness to be one: `fitness_allele0 1.0` and `fitness_allele1 1.0`.

The data file for simulated neutral alleles (fitness of 1) under a populations size of 10^5 and a mutation rate of 10^{-5} is available `here`. The corresponding parameters file is available `here`.

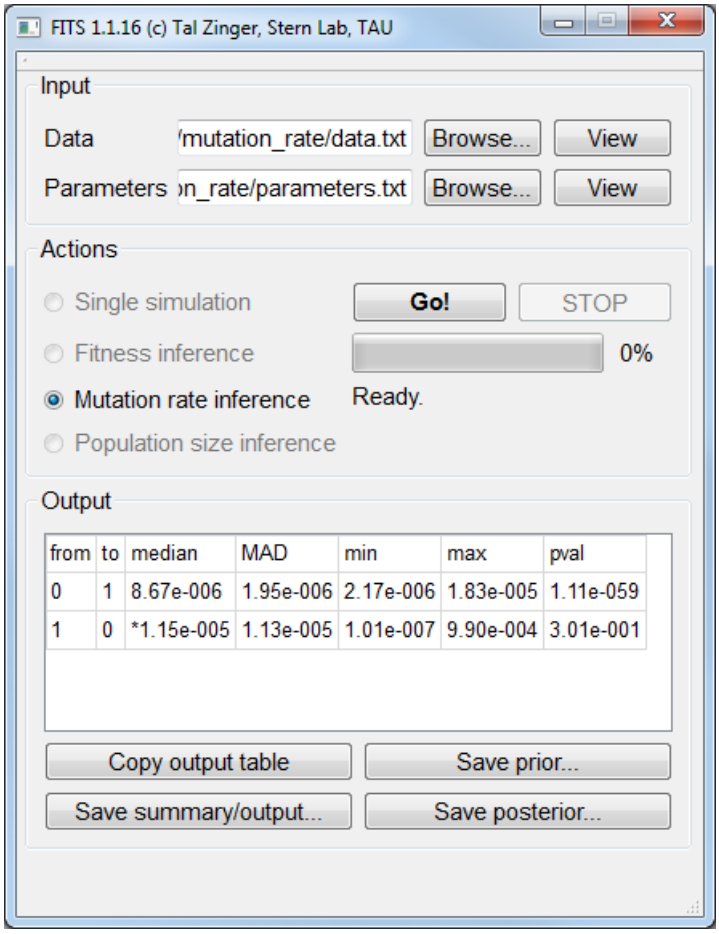

Fig. 4: FITS inferred $0 \rightarrow 1$ mutation rate of $8.67*10^{-6}$, and $1 \rightarrow 0$ mutation rate of $1.15*10^{-5}$.

2.3.3 Population size inference

If the mutation rates are known and the fitness of the measured allele is known, then the population size parameter may be inferred. Similar to the mutation rate inference, this can be performed by using frequency data from several loci that has the same fitness values. Here we simulated 10 neutral positions for 15 generations, using a population size of 100,000 and a mutation rate of 10^{-5}.

Our prior knowledge suggests that the population size may be in the range between 10^4 and 10^7. We therefore set the parameter `Nlog_min 4` to indicate minimum population size of 10^4. We also set the parameter `Nlog_max 7` to indicate the maximum population size of 10^7.

Our prior knowledge also suggests that the alleles we measured are neutral, and therefore we set the wildtype and mutant alleles to have a fitness of 1: `fitness_allele0 1.0` and `fitness_allele1 1.0`.

Other parameters such as mutation rates, number of simulations and the sampling rate are similar to the *Fitness inference* example. The example data file is available `here`. The corresponding parameters file is available `here`.

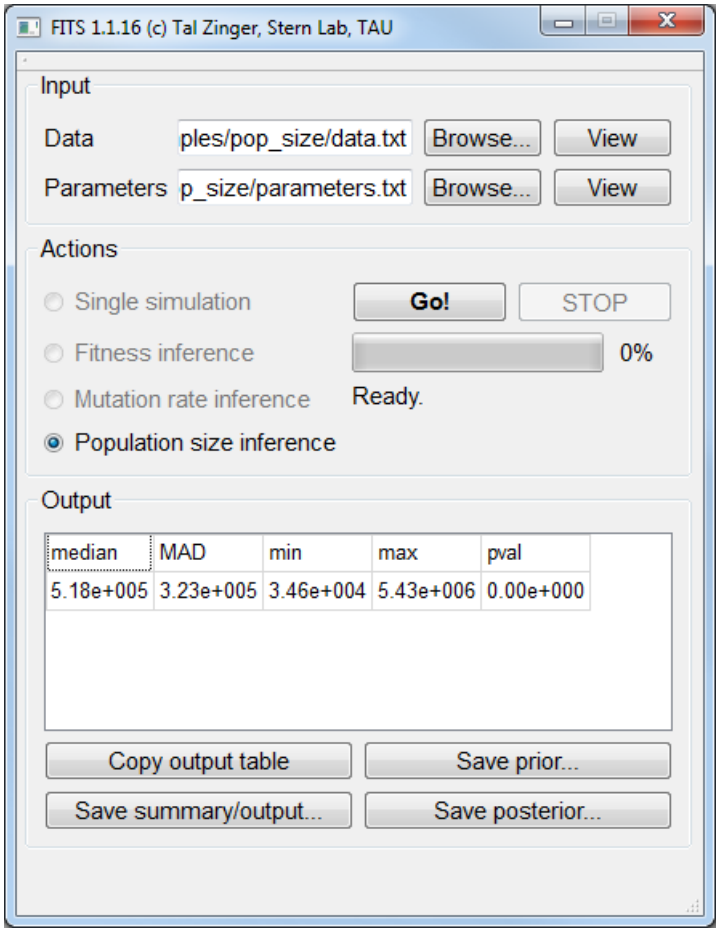

Fig. 5: FITS inferred a population size of $5.18*10^5$.

2.3.4 Trajectory simulations

Sometimes, we wish to have frequency data generated. Since simulations are a cornerstone on which FITS is relying on, it is possible to ask the framework to perform simulations of frequencies for given mutation rates, population size and fitness value.

In order to do so, we need to provide these three parameters as described in previous examples. For this example, we'll use a mutation rate of 10^{-3}, a fitness of 1.02 and a population size of 10^5.

We wish to simulate two alleles only. We therefore set `num_alleles 2` to indicate two alleles.

We set for the two alleles the fitness values of 1 for the wildtype and 1.02 for the mutant: `fitness_allele0 1.0` and `fitness_allele1 1.02`.

We set the corresponding (equal) mutation rates: `mutation_rate0_1 0.001` and `mutation_rate1_0 0.001`.

The population size is set by defining `N 100000`.

The last two things to consider are the frequency of the alleles in the beginning of the simulation, and the number of generations to simulate. Here we will assume that the wildtype allele is fixated for the beginning of the simulation. We'll therefore set `init_freq_allele0 1` and `init_freq_allele1 0`. To control for the number of generations (100 in our example) we set `num_generations 100`.

Note: There's no need to load a *Data file* in order to perform the simulations.

2.3.5 Considering Sample Effect

Population bottlenecks are common during any evolutionary scenario, whether *in vivo* or *in vitro* in experimental populations. In particular during serial passaging, only a fraction of the progeny will be carried on to the next passage. An additional layer of sampling also exists: the number of genomes that are sampled to be sequenced may often be much smaller than the population size itself. FITS can account for both of these types of sampling effects by using the following three parameters: `bottleneck_size`, `bottleneck_interval` and `sample_size` , as illustrated below:

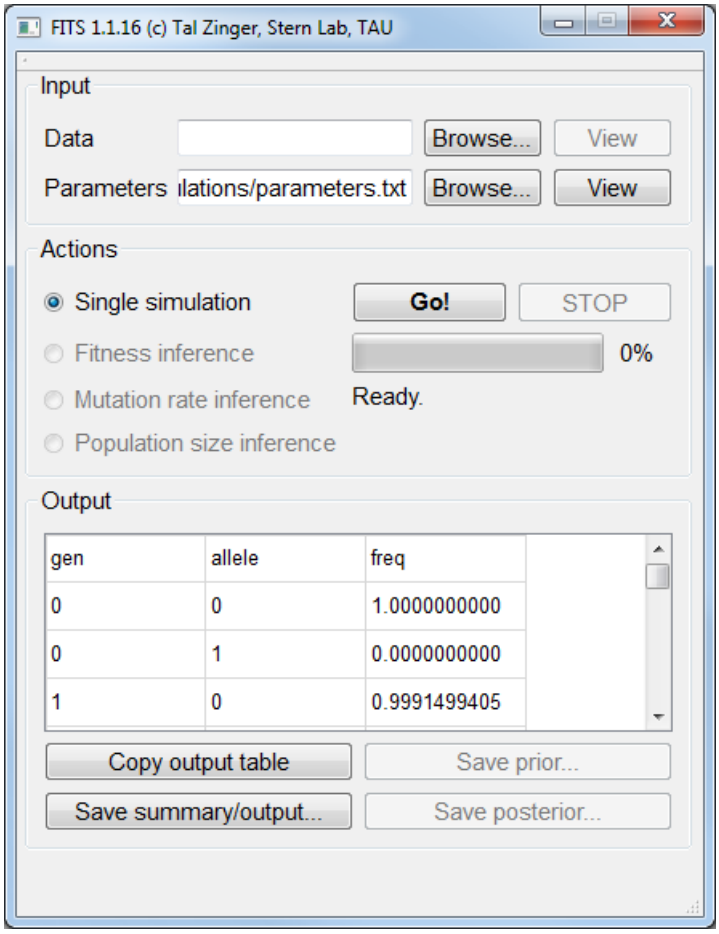

Fig. 6: Simulation results are available in the **Output** area.