PReFerSim v 1.0

Here we provide a user's guide for PReFerSim where we outline how to download, compile and run this program. We also provide some troubleshooting for common error messages at the very end of the document. For a detailed overview, and a description of the theory and math behind this program, please refer to our paper: Ortega-Del Vecchyo *et al.* 2016. PReFerSim: Fast simulation of demography and selection under the Poisson Random Field model. *Submitted*

1. OVERVIEW

This program performs simulations under the Poisson Random Field model, where it is assumed that the number of independent mutations arising each generation follows a Poisson distribution. The change in frequency of the independent mutations across generations follows a Wright-Fisher model with selection.

This program can model any demographic scenario and a wide variety of distributions of fitness effects for the mutations arising each generation. This program can be used to:

- Follow the allele frequency trajectories of alleles that end at a certain present-day frequency in the population.
- Obtain a list of segregating sites, along with their frequencies and selection coefficients.
- Print the site frequency spectrum.
- Print the number of heterozygous and homozygous derived genotypes per individual in segregating sites.
- Print the number of segregating sites at different time points during the simulation.
- Print the sum of the derived allele frequency of the segregating variants at different time points during the simulation.
- Print the genetic load at different time points during the simulation.
- Print the number of variants fixed across all times during the simulation.

2. CITATION

PReFerSim should be cited as follows: Ortega-Del Vecchyo, D., Marsden C.D. Lohmueller, K.E. (2016). PReFerSim: Fast simulation of demography and selection under the Poisson Random Field model. *Submitted*

3. SYSTEM REQUIREMENTS

PReFerSim was written in C and is intended to run on Unix, or Unix-like operating systems, such as Linux or Mac OS X. To use PReFerSim with Mac OS X, you must install the Mac OS X software developer kit usually included with Xcode. The program can also be used in a Windows operating system, but it requires the installation of Cygwin. If you use Cygwin, make sure to install GNU's gcc compiler when downloading or updating Cygwin to be able to compile PReFerSim.

In order to run PReFerSim *GSL must be installed* on your computer (available here: http://www.gnu.org/software/gsl/).

4. Download

PReFerSim is provided free of charge for use under the GNU General Public License v3.0 license. The program and associated example and help files can be downloaded, from: https://github.com/LohmuellerLab

Unzip the downloaded folder with the following command:

```
unzip PReFerSim_v1.zip
```

The following files are present in the folder:

Alleles11.txt.- A list of allele ID's to follow allele frequency trajectories.
Bottleneck.txt.- Sample demographic history file
DemographicHistoryBreedDogs1000.txt.- Sample demographic history file
DemographicHistoryTwoConstantSizes.txt.- Sample demographic history file
Examples.sh.- A set of examples to run PReFerSim
FChangedExample.txt.- Sample file showing changing inbreeding coefficients over different epochs
GetListOfRunsWhereFrequencyMatches.pl.- perl script used to extract list of alleles at a certain frequency from PReFerSim output. Used when looking at allele frequency trajectories.
ParameterFileX.txt.- Variety of example parameter files
PReFerSim.c.- The C code for PReFerSim
SelPointTest.txt.- Example file showing a discrete probability distribution of fitness effects
SelUnifBounds.txt.- Example file showing a distribution of fitness effects with uniform intervals.

5. Compiling

5.1. Install GSL

In order to run PReFerSim, the program must be compiled prior to first use. As noted in the System Requirements above, this requires GSL. For example, you can follow these directions to install GSL on a mac http://connor-johnson.com/2014/10/12/using-the-gnu-scientific-library-on-a-mac/

Please note it is critical that you install gsl is in your PATH. For example, following the link directions on a mac, you would use the following command.

```
PATH=/usr/local/include:$PATH
```

5.2. Compile PReFerSim

From with the directory where PReFerSim.c is located, you can compile PReFerSim with the following command:

```
gcc -g -o PReFerSim PReFerSim.c -lm -lgsl -lgslcblas -O3
```

This will create a new executable file called 'PReFerSim' in your folder.

We have experienced some errors compiling PReFerSim on El Capitan on mac associated with locating the GSL library in the PATH. In those cases, you should provide the location of the gsl library when compiling. For example:

```
gcc -g -o PReFerSim PReFerSim.c -lm -lgsl -lgslcblas -O3 -
I/usr/local/include -L/usr/local/lib
```

6. General Usage

In order to run PReFerSim, the user must provide a parameter file with the options on how the program should be run (discussed in detail below).

The program can then be run with five arguments (all required) following a syntax similar to the one detailed below:

```
GSL_RNG_SEED=1 GSL_RNG_TYPE=mrg ./PReFerSim ParameterFile1.txt 1
```

Where:

`GSL_RNG_SEED=1` # This defines the random seed number used by the program (1 in this case). You should set the seed to a random value yourself to ensure that independent runs are truly independent.
`GSL_RNG_TYPE=mrg` # This defines the random number generator used (mrg)
`./PReFerSim` # This executes the program
`ParameterFile1.txt` # This is the parameter file which includes the instructions needed for the program to run
`11` # This is the replicate number. This is specified by the user and will be used in the output files to facilitate running the program in parallel.

7. Overview parameter file

The parameter file specifies the options that you will use to run PReFerSim.

Every option of the parameter file must be followed by one optional colon, one or more blank spaces or tabs, and the value of the parameter. Every option of the program must be typed exactly as shown in the manual, although the program can recognize changes in the capitalization of letters.

Example:
MutationRate: 4

Table 1 gives an overview of the parameter options that can be specified in the parameter file. Each of these is described in detail in the next section, and example parameter files are given in the PReFerSim folder.

Table summarising parameter options for the parameter file

	Req/opt	Input type	Default	Summary
General parameters				
DemographicHistory	Required	File	NA	Details on population size changes
MutationRate	Required	Decimal > 0	NA	Mutation rate: Θ (4Nμl in diploids, 2Nμl in haploids)
h	Optional	Decimal number between 0 and 1	0.5	Dominance coefficient of mutations
Parameters specifying DFE				
DFEType	Required	point/ gamma/ lognormal/ beta/ pointprob/ unifbounds	NA	Distribution of fitness effects to use.
DFEPointSelectiDonCoefficient	Required with point DFE	Decimal	NA	Selection coefficient ‘s’ value for all mutations
SFSNoBurnIn	Optional with DFEPointSelecti	0/1 (1 implement	0	Start with a number of mutations at different frequencies that reflects an equilibrium site

	onCoefficient	s this option)		frequency spectrum. Used to skip the Burn-in process. Can only be used with DFEType: point and h = 0.5
DFEParameterOne	Required with gamma/lognormal/beta DFE	Decimal > 0	NA	First parameter of the distribution of fitness effects
DFEParameterTwo	Required with gamma/lognormal/beta DFE	Decimal > 0	NA	Second parameter of distribution of fitness effects
DFEParameterThree	Required with gamma/lognormal DFE	Non negative integer	NA	Ne used in scaled selection parameter 2Ns
DFEPointProbSelectionFile	Required with pointprob DFE	File	NA	File with 2Ns values and their probabilities
DFEPointProbNe	Required with pointprob DFE	Non negative integer	NA	Ne used in scale selection parameter 2Ns
DFEUnifBoundsSelectionFile	Required with uniform-bound DFE	File	NA	File with 2Ns values and their probabilities
DFEUnifBoundsNe	Required with uniform-bound DFE	Non neg integer	NA	Ne used in scale selection parameter 2Ns
Parameters specifying relaxation of selection				
EpochOfRelaxation	Optional. However, if used all four relaxation of selection parameters must be given.	Non neg integer	NA	Epoch number to begin relaxed selection. The most ancient epoch has a number equal to 0, and the numeration for subsequent epochs continues going forward in time.
RelaxationSelectionThreshold		Decimal	NA	Mutations below this selection coefficient will have their selection coefficient relaxed
RelaxationSelectionType		0/1	NA	Specifies how to relax selection coefficient. The two options, 0 and 1, are explained in section 8.3.
RelaxationSelectionCoefficient		Decimal	NA	Used to calculate new selection coefficient
Parameters specifying inbreeding				
FixedFValue	Optional	Decimal	0.0	Inbreeding coefficient
ChangedFValue	Optional	File		Inbreeding coefficient for each epoch in demographic history file
Parameters for output options				
FilePrefix	Optional	Text	Output	Prefix for outfiles
PrintSNPNumber	Optional	0/1 (1 invokes this option)	0	Print number of segregating sites/gen
PrintSumOfS	Optional	0/1 (1 invokes this option)	0	Print sum of selection coefficients across all segregating sites/gen
PrintSumDAF	Optional	0/1 (1 invokes this option)	0	Print sum of derived allele frequencies across all SNPs/gen
PrintWeightedSumOfS	Optional	0/1 (1 invokes this option)	0	Print sum of (allele frequency * selective coefficients) across all SNPs/gen
PrintGenLoad	Optional	0/1 (1 invokes this option)	0	Print genetic load/gen

PrintFixedSites	Optional	0/1 (1 invokes this option)	0	Print details about fixed and extinct mutations
PrintSFS	Optional	0/1 (1 invokes this option)	0	Print present day SFS
PrintSegSiteInfo	Optional	0/1 (1 invokes this option)	0	Print details on segregating sites
PrintSampledGenotypes	Optional	0/1 (1 invokes this option)	0	Gather genotype information for every segregating mutation.
n	Optional	Non neg integer	20	Sample size used for printing output
LastGenerationAFSamplingValue	Optional	0/1 (1 invokes this option)	NA	Samples alleles for each mutation in the last generation and prints their frequencies to the output of PrintSegSiteInfo.
Options for printing allele frequency trajectories				
AlleleTrajsInput	Optional	File	NA	ID's of alleles you want trajectories for
AlleleTrajsOutput	Optional	Text	NA	Name for trajectories output file

A number of example parameter files are provided in the download folder with the prefix "ParameterFile".

8. PARAMETERS IN THE PARAMETER FILE

Below we provide the specific details of how to set the parameters outlined in Table 1.

8.1. General parameters

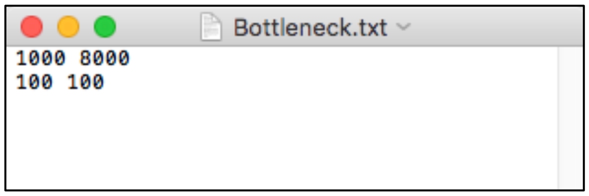
DemographicHistory: (Required)

Specifies a file that describes population size changes through time. This file lists one or more epochs, each on its own line and ordered from oldest to most recent. Each epoch is described by two values; the population size in number of chromosomes (i.e. 2N in diploid organisms) and length of an epoch (i.e. number of generations). There should be a blank space between the two values on each line. All of the numbers in the file must be nonnegative integer numbers.

When you run the program, it is advisable that in the first epoch you run the program for a long time to achieve mutation-drift-selection balance at the end of that first epoch (This is called the burn-in period). In our experiments, we have seen that running the program for 16N generations is sufficient to achieve mutation-selection balance.

Example:
DemographicHistory: Bottleneck.txt

Bottleneck.txt is a text file provided in the downloaded folder and shown here. Two epochs are described in the file, one per line. During the first epoch the population size (2N) is 1,000 and lasts 8,000 generations. This is followed by an epoch with a population size of 2N = 100 for 100 generations.



```
1000 8000
100 100
```

Further examples are given in the download folder:

- DemographicHistoryBreedDogs1000.txt
- DemographicHistoryTwoConstantSizes.txt

MutationRate: (Required)

This specifies the value of θ (4Nμl in diploids, 2Nμl in haploids) in the first epoch specified by the demographic history. This parameter is equal to the multiplication of 2 * the number of chromosomes (in terms of the most ancestral population size) * mutation rate per bp μ * number of sites l (bases). A Poisson number of mutations with a mean equal to half of what is specified by this parameter enter the population every generation. The mutation rate is re-scaled automatically when the population changes its size in following epochs. As an example, if the population size is doubled in one particular epoch, the mutation rate is also doubled in that epoch. The mutation rate must be a nonnegative decimal.

We recommend to not try θ values higher than 2000 due to restrictions with the number of sites that can be simulated in each run of PReFerSim. The number of sites simulated in each run cannot exceed 2,000,000,000. PReFerSim will make an estimate of the number of sites that will be simulated each run given the mutation rate and demographic history given to the program. If this number exceeds 2,000,000,000 then the program will end and ask you to lower the mutation rate given in MutationRate. If you wish to simulate a large number of sites, we recommend doing many independent runs of PReFerSim using the same parameter file but a different random seed and replicate number (see section 9 for an example). Note that under the PRF model, every new mutation is independent. Therefore, it is exactly the same to run PReFerSim once to simulate 10 million sites to run PReFerSim ten times to simulate 1 million sites.

Example:
MutationRate: 4

h: (Dominance coefficient, Optional)

h is the dominance coefficient of mutations. The fitnesses of the genotypes are A1A1 = 1; A1A2 = 1-h*(2*s); A2A2 = 1-(2*s). Set h to 1.0 for dominant mutations, 0.5 for additive mutations, 0 for recessive mutations.
This must be a decimal number between 0 and 1. [Default value = 0.5].

Example:

h: 0.5

8.2. Specifying the distribution of fitness effects (Required)

To run PReFerSim, you must specify the distribution of fitness effect you wish to use. The fitnesses of the genotypes are A1A1 = 1; A1A2 = 1-h*(2*s); A2A2 = 1-(2*s), where A2 is the derived allele. Under any conditions that the program is run, the value of s for any segregating site cannot exceed 0.5. If it does, PReFerSim will set the value of s to 0.5 for that segregating site. Six distributions of fitness effects are available within PReFerSim: *point, gamma, lognormal, beta, pointprob, unifbounds*. *point* assumes that all the mutations have the same selection coefficient, s. *gamma* and *lognormal* assume a gamma and a log-normal distribution of fitness effects, which have been previously used to model the distribution of fitness effects (DFE) in humans (Boyko et al., 2008). *Beta*, *pointprob* and *unifbounds* provide highly customizable DFE's that can be used to model a large family of DFE's. We will describe these in more detail below.

The DFE is specified by the required parameter DFEType as well as a number of additional parameters unique to each DFE. Below we describe the parameter options needed for each DFE.

8.2.1. ## Constant distribution of fitness effects ##

DFEType: (Required)
DFEType specifies the distribution of fitness effects. To specify a constant distribution of fitness effects whereby all the mutations share the same selection coefficient 's', use: DFEType: point. One further parameter, DFEPointSelectionCoefficient, must be specified with this DFEType.

DFEPointSelectionCoefficient: (Required with point DFE)
Specifies the selection coefficient 's'.
DFEPointSelectionCoefficient must be a decimal number. This number can take positive and negative values. If the value is positive, the derived variant is under

negative selection. If this value is negative, the derived variant is under positive selection.

SFSNoBurnIn: (Optional with DFEPointSelectionCoefficient)
WARNING - This option can only be used if the option 'DFEPointSelectionCoefficient' is used *and* h = 0.5.
If this option is used, the program first will generate a Site Frequency Spectrum that is dependent on the parameter gamma = 2Ns, the number of chromosomes in the first epoch (ignoring any inbreeding that takes place in the first epoch) and the mutation rate. Following theory from the Poisson Random Field model, the SFS has a number of mutations at different frequencies that has a Poisson distribution with a mean value equal to formula (2) from Bustamante et al. 2001 *Genetics*, which is based on the derivations made by Sawyer and Hartl, 1992, *Genetics,* particularly their formula (11). The forward simulations are then performed starting with a number of mutations at different frequencies as defined by the SFS. Therefore, this option skips the need of running the simulations for a long time during the first epoch in the burn-in period, saving computational time. Instead of conducting the burn-in, the simulation is seeded with the expected patterns of genetic variation under the stationary distribution of mutation, selection, and drift.
IMPORTANT: You cannot use this option if you want to follow the allele frequency trajectory of an allele using the AlleleTrajsInput and AlleleTrajsOutput options. You cannot do that because if you used the burn-in option and followed the allele frequency trajectories, the allele frequency trajectories might not end at the time where the allele emerged, but at the time where the simulation started after we generated an equilibrium SFS using the SFSNoBurnIn option. This would give an incomplete view of the full trajectory trajectory of alleles.
To use this option, you must set the *SFSNoBurnIn:* option equal to 1 . [Default value = 0]

Example:
SFSNoBurnIn: 1

Example of how to specify a point distribution of fitness effects:
DFEType: point
DFEPointSelectionCoefficient: 0.0

Example parameter files using constant DFE:
ParameterFile1.txt, ParameterFile11.txt, ParameterFile11_B.txt, ParameterFile12.txt, ParameterFile13.txt, ParameterFile13_B.txt

8.2.2. ## Gamma distribution of fitness effects ##

DFEType: (Required)

To specify a gamma distribution of fitness effects, use "DFEType: gamma". Note, here the population-scaled selection parameter, 2Ns (s is positive), will follow a gamma distribution. Three further parameters must be specified with this DFEType.

DFEParameterOne: (Required with gamma DFE)
The shape parameter of the gamma distribution.
This must be a decimal number bigger than zero.

DFEParameterTwo: (Required with gamma DFE)
The scale parameter of the gamma distribution. Here the mean of the gamma distribution is equal to the multiplication of the scale parameter times the shape parameter.
This must be a decimal number bigger than zero.

DFEParameterThree: (Required with gamma DFE)
The effective population size N used in the scaled selection parameter 2Ns. Essentially, when you simulate a gamma DFE, you draw 2Ns values for each new mutation from the specified gamma distribution. These 2Ns values are transformed to s values based on the N value provided in DFEParameterThree.
This must be a nonnegative integer.

Example of how to specify gamma distribution of fitness effects. This example uses the parameters of the gamma distribution of fitness effects inferred by Boyko et al. (2008) in African Americans:
DFEType: gamma
DFEParameterOne: 0.184
DFEParameterTwo: 319.8626
DFEParameterThree: 1000

Example parameter files using gamma DFE:
ParameterFile2.txt, ParameterFile4.txt, ParameterFile6.txt, ParameterFile9.txt

8.2.3. ## LogNormal distribution of fitness effects ##

DFEType: (Required)
To use a lognormal distribution of fitness effects, use "DFEType: lognormal". Here the scaled selection parameter, 2Ns (s is positive), will follow a lognormal distribution. Three further parameters must be specified with this DFEType.

DFEParameterOne: (Required with lognormal DFE)
The mean of the lognormal distribution. This must be a decimal number.

DFEParameterTwo: (Required with lognormal DFE)

The standard deviation of the lognormal distribution. This must be a nonnegative decimal number.

DFEParameterThree: (Required with lognormal DFE)
The effective population size N used in the scaled selection parameter 2Ns. For each mutation, the 2Ns values associated to each mutation will be transformed to s values based on the N parameter. This must be a nonnegative integer.

Example of how to specify the log normal distribution of fitness effects:
DFEType: lognormal
DFEParameterOne: 5.02
DFEParameterTwo: 5.94
DFEParameterThree: 1000

Example parameter files using log normal DFE:
ParameterFile8.txt

8.2.4. ## Beta distribution of fitness effects ##

DFEType: (Required)
To specify that the selection coefficient s follows a beta distribution of fitness effects, where s must be positive, use DFEType: beta. Two further parameters must be specified with this DFEType.

DFEParameterOne: (Required with beta DFE)
The alpha parameter of the beta distribution. The value given should be a decimal number bigger than zero.

DFEParameterTwo: (Required with beta DFE)
The beta parameter of the beta distribution. The value given should be a decimal number bigger than zero.

Example of how to use beta distribution of fitness effects:
DFEType: beta
DFEParameterOne: 0.5
DFEParameterTwo: 0.5

Example parameter files using log normal DFE:
ParameterFile5.txt

8.2.5. ## Discrete probability distribution of fitness effects ##

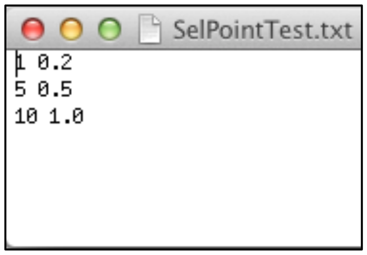
DFEType: (Required)
This option models the DFE as a probability distribution where the scaled selection coefficient 2Ns is sampled from a set of fixed discrete values with

associated probabilities. To use this option, use “DFEType: pointprob”. Two further parameters must be specified with this DFEType.

DFEPointProbSelectionFile: (Required with point probability DFE)

This file contains the particular selection coefficients (2Ns) values that will be used in the simulation. Each line in this file has two numbers (must be decimals) separated by a blank space. The first number is the 2Ns value, and the second number defines cumulative distribution function boundaries (more below).
The user can specify an arbitrary number of 2Ns values, and the values of 2Ns can be positive (modeling deleterious mutations) or negative (modeling positively selected mutations). The only three restrictions are that: 1) the numbers in the second column are between 0 and 1, 2) the numbers are listed in ascending order with respect to the second column, and that 3) the number in the second column in the last row is equal to 1.0

An example is SelPointTest.txt (also in the download folder).



```
1 0.2
5 0.5
10 1.0
```

As an example of how 2Ns values are sampled using the above file, PReFerSim generates a random number between 0 and 1 for each mutation. If the random number value generated for the mutation is between 0 and 0.2, the value of 2Ns is 1. If the random number value is between 0.2 and 0.5, the value of 2Ns is 5. If the random number value is between 0.5 and 1.0, the value of 2Ns is 10.

DFEPointProbNe: (Required with point probability DFE)
The effective population size N used in the scaled selection parameter 2Ns.
This number must be a nonnegative integer.

Example of how to use point probability distribution of fitness effects:
DFEType: pointprob
DFEPointProbSelectionFile: SelPointTest.txt
DFEPointProbNe: 10000

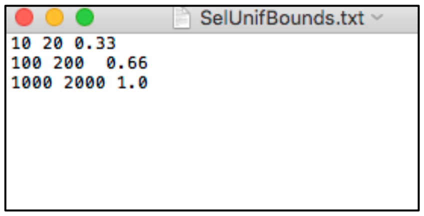
Example parameter files using point probability DFE:
ParameterFile3.txt

8.2.6. ## Distribution of fitness effects with uniform intervals ##

DFEType: (Required)
To use a distribution of fitness effects with uniformly distributed intervals, use "DFEType: unifbounds". Here the scaled selection coefficient (2Ns) of the mutations will take particular values according to the uniform intervals specified by the user, Two further parameters must be specified with this DFEType.

DFEUnifBoundsSelectionFile: (Required with unifbounds DFE)
This file contains the particular 2Ns values that will be used in the simulation. Each line in this file has three numbers (must be decimals), each separated by a blank space. The first and second numbers are a pair of 2Ns values that define a uniform interval, and the third number defines cumulative distribution function boundaries (more below). The values of 2Ns given in the first and second column can be either positive or negative.

The user can specify as many uniform intervals as needed. The four conditions that must be satisfied in the file are that: 1) the numbers in the third column have values between 0 and 1, 2) the numbers are listed in ascending order with respect to the third column, and that 3) the number in the third column in the last row is equal to 1.0. 4) In each row, the number in the second column must be bigger than the one in the first column.
An example is SelUnifBounds.txt (also in the download folder)



```
10 20 0.33
100 200  0.66
1000 2000 1.0
```

PReFerSim generates a random number between 0 and 1 for each mutation. If the random number value is between 0 and 0.33, then the value of 2Ns is a random number sampled uniformly from the interval 10-20. If the random number value is between 0.33 and 0.66, then the value of 2Ns is a random number sampled uniformly from the interval 100-200. If the random number value is between 0.66 and 1.0, then the value of 2Ns is a random number sampled uniformly from the interval 1000-2000.

DFEUnifBoundsNe: (Required with unifbounds DFE)

The effective population size, N, used in the scaled selection parameter 2Ns. This number must be a nonnegative integer.

Example of how to use this distribution of fitness effects with uniform intervals:
DFEType: unifbounds
DFEUnifBoundsSelectionFile: SelUnifBounds.txt
DFEUnifBoundsNe: 100000

Example parameter files using uniform DFE:
ParameterFile7.txt

8.3. Parameters for relaxation of selection (Optional)

Four parameters must be given if you plan to relax the selection coefficient of all the mutations. Relaxation of selection in PReFerSim means that new mutations emerging after a certain time point in the simulation will receive a different value of the selection coefficient s depending on whether their s value is smaller than or equal to a certain threshold. It also means that all mutations existing before that time point will have their s values switched at that time point depending on whether their s value is smaller than or equal to a certain threshold.

EpochOfRelaxation: (Required if relaxation of selection options used)
Starting from this epoch, the selective coefficient of all the mutations will be relaxed. The most ancestral epoch has the number 0, the second one most ancestral has the number 1, etc. For example, with the demographic history outlined in the file 'Bottleneck.txt', to relax the selection in the bottlenecked population (i.e. 2N=100), you would set this parameter to 1.

This must be a nonnegative integer number.

Example:
EpochOfRelaxation: 1

RelaxationSelectionThreshold: (Required if relaxation of selection options used)

All the mutations with a selection coefficient smaller than or equal to this threshold will have their selection coefficient relaxed. For example, to relax the selection coefficient for weakly deleterious mutations (e.g. $s \leq 0.0001$), you would set this parameter as 0.0001. To relax the selection coefficient for all mutations, you would set this parameter to 1.0.
This must be a decimal number.

Example:
RelaxationSelectionThreshold: 1.0

RelaxationSelectionType: (Required if relaxation of selection options used)

There are two ways to relax the selection coefficient. If this parameter is set to 0, then the new selection coefficient of all the mutations with selection coefficients smaller than or equal to what is specified in ‘RelaxationSelectionThreshold:’ will be set to what is specified in the parameter ‘RelaxationSelectionCoefficient:’. If the parameter is set equal to 1, then the new selection coefficient of each mutation with a selective coefficient smaller than or equal to what is specified in ‘RelaxationSelectionThreshold:’ will be equal to the multiplication of the current selective coefficient of each mutation and the value specified in the parameter ‘RelaxationSelectionCoefficient:’.

In other words:

If RelaxationSelectionType = 0, the new s for mutations with s < RelaxationSelectionThreshold will be equal to RelaxationSelectionCoefficient
If RelaxationSelectionType = 1, the new s for mutations with s < RelaxationSelectionThreshold will be equal to (RelaxationSelectionCoefficient * the existing s value of each mutation)

This must be an integer number equal to 0 or 1.

Example:
RelaxationSelectionType: 0

RelaxationSelectionCoefficient: (Required if relaxation of selection options used)

This is the new value of the selection coefficient, s, of each mutation if RelaxationSelectionType = 0 and the selective coefficient of the variant is less than or equal to what is specified in RelaxationSelectionThreshold. If RelaxationSelectionType = 1, the new selection coefficient for each mutation is equal to this value times the current selective coefficient of each mutation if the selective coefficient of the variant is less than or equal to what is specified in RelaxationSelectionThreshold.
This must be a decimal number.

Example:
RelaxationSelectionCoefficient: 0.9

Example parameter files using relaxation of selection:
ParameterFile9.txt, ParameterFile10.txt

Example of how to relax selection from ParameterFile9.txt:
EpochOfRelaxation: 1
RelaxationSelectionThreshold: 1.0

RelaxationSelectionType: 0
RelaxationSelectionCoefficient: 0

In this example, selection will be relaxed starting from the second epoch of the demographic model as specified in "*EpochOfRelaxation: 1*". All the existing mutations and mutations emerging starting from that epoch whose selection coefficient s assigned was less than or equal to 1.0 (as specified by "*RelaxationSelectionThreshold: 1.0*"), will have a new selection coefficient equal to 0. "*RelaxationSelectionType: 0"* indicates that all variants whose s value will be relaxed will have a value equal to what is specified in the parameter *"RelaxationSelectionCoefficient"*, which in this case is 0.

8.4. Parameters to specify inbreeding (Optional)

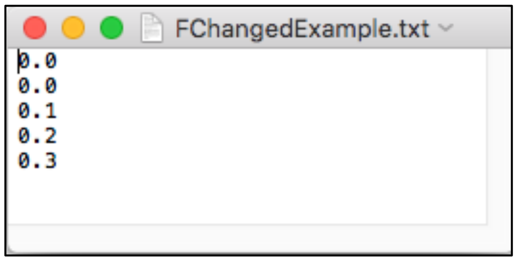
Inbreeding can be incorporated into PReFerSim using *either* the FixedFValue or ChangedFValue parameter. You cannot set both of these parameters at the same time.

FixedFValue: (Optional)
This sets the inbreeding coefficient for the population. This must be a decimal number between 0 and 1. Values > 0 ≤1 indicate an inbred population (i.e. excess homozygosity), 0.0 indicates a population at HWE (default value). If you use this parameter, then the inbreeding coefficient will have the same value across all generations.
Default value is 0.0

Example:
FixedFValue: 0.0

ChangedFValue: (Optional)
This specifies the inbreeding coefficient in each epoch for the population. You must provide a file for this parameter. In this file, the number of rows should be the same as in the file that was given to the parameter 'DemographicHistory:', where each row will contain a decimal specifying the inbreeding coefficient for the population at that particular epoch.

An example file in the downloaded folder is: FChangedExample.txt



```
0.0
0.0
0.1
0.2
0.3
```

Here in the first 2 epochs, F=0. In epoch 3, F=0.1. In epoch 4, F=0.2. In epoch 5, F=0.3.

Example:
ChangedFValue: FChangedExample.txt

Example parameter files using inbreeding parameters:
ParameterFile4.txt, ParameterFile6.txt

8.5. Parameters specifying output file options

The following output options are available

n: (Optional)
This is the sample size (in numbers of chromosomes) used for the output given in the options 'PrintSNPNumber:', 'PrintSumOfS:', 'PrintSumDAF:' , 'PrintWeightedSumOfS:', 'PrintGenLoad:' , 'PrintSFS', 'LastGenerationAFSamplingValue' (detailed below). It is important to note that, at each generation in the simulation, the same set of chromosomes sampled are used to calculate the statistics that can be printed by invoking the seven options outlined above.
This must be a nonnegative integer number. [Default value = 20]

FilePrefix: (Optional)
A prefix to use before all the output files.
Can include a path to where you want to print all the output files. [Default value = "Output"]

Example:
FilePrefix: MiniTest/Output

PrintSNPNumber: (Optional)
Prints the number of segregating sites per generation in a sample from the past going into the present. The sample size (in numbers of chromosomes) used to calculate the number of segregating sites per generation is given by the option 'n:'.
The argument to PrintSNPNumber: must be an integer number equal to 0 or 1.
[Default value = 0]. PrintSNPNumber: 1 will invoke this option.
Outfile suffix: .num_snp_out.txt

Example:
PrintSNPNumber: 1

PrintSumOfS: (Optional)

Prints the sum of the selection coefficients across all segregating sites per generation in a sample from the past going into the present. The sample size (in numbers of chromosomes) used to calculate this value is given by the option 'n:'.
The argument to PrintSumOfS: must be an integer number equal to 0 or 1. [Default value = 0]. PrintSumOfS: 1 will invoke this option.
Outfile suffix: .s_out.txt

Example:
PrintSumOfS: 1

PrintSumDAF: (Optional)
Prints the sum of the derived allele frequencies across all SNPs per generation in a sample from the past going into the present. The sample size (in numbers of chromosomes) used to calculate this value is given by the option 'n:'.
The argument to PrintSumDAF: must be an integer number equal to 0 or 1. [Default value = 0]. PrintSumDAF: 1 will invoke this option.
Outfile suffix: .sum_maf_out.txt

Example:
PrintSumDAF: 1

PrintWeightedSumOfS: (Optional)
Prints the sum of (allele frequency * selective coefficients) across all SNPs per generation in a sample from the past going into the present. The sample size (in numbers of chromosomes) used to calculate this value is given by the option 'n:'.
The argument to PrintWeightedSumOfS: must be an integer number equal to 0 or 1. [Default value = 0]. PrintWeightedSumOfS: 1 will invoke this option.
Outfile suffix: Output.11.s_weight_out.txt

Example:
PrintWeightedSumOfS: 1

PrintGenLoad: (Optional)
Prints out the multiplicative genetic load per generation from the past going into the present if set to 1. The number of alleles sampled to calculate the genetic load is given by the option 'n:'. The genetic load is estimated as:

$$L = 1 - e^{-\sum 2hs\left(p(1-p)\right)+sp^2}$$

Where p is the frequency of the derived allele, h is the dominance coefficient and s is the selective coefficient.
The argument to PrintGenLoad: must be an integer number equal to 0 or 1. [Default value = 0]. PrintGenLoad: 1 will invoke this option.
Outfile suffix: .gen_load_out.txt

Example:
PrintGenLoad: 1

PrintFixedSites: (Optional)
Prints out a file with three columns. For each generation from the past going into the present there will be one row. In each row, the first column contains the number of mutations that became extinct, the second column has the number of mutations that became fixed and the third column has the sum of the selection coefficients of the fixed SNPs.
The argument to PrintFixedSites: must be an integer number equal to 0 or 1. [Default value = 0]. PrintFixedSites: 1 will invoke this option.
Outfile suffix: fixed_sites_out.txt

Example:
PrintFixedSites: 1

PrintSFS: (Optional)
Prints out the site frequency spectrum in the present day. This is obtained by sampling n alleles per segregating mutation [as given by the option n:] and counting the number of derived alleles observed per segregating site. The output file will contain n-1 elements. The first corresponds to the number of variants with 1 derived alleles observed, the second element shows the number of variants with 2 derived allele observed, etc. The last column will contain the number of variants with n-1 derived alleles observed.
The argument to PrintSFS: must be an integer number equal to 0 or 1. [Default value = 0]. PrintSFS: 1 will invoke this option.
Outfile suffix: .sfs_out.txt

Example:
PrintSFS: 1

PrintSegSiteInfo: (Optional)
Prints out a file with information for every segregating mutation in the population. The information for each mutation is given in each row. Five elements will be shown for each mutation in five consecutive columns:
A) replicate number,
B) frequency of the allele,
C) The selection coefficient s,
D) The generation in which the allele emerged and
E) An unique ID number associated to that mutation.

The argument to PrintSegSiteInfo: must be an integer number equal to 0 or 1. [Default value = 0]. PrintSegSiteInfo: 1 will invoke this option.
Outfile suffix: .full_out.txt

Example:

PrintSegSiteInfo: 1

LastGenerationAFSamplingValue: (Optional)
If set equal to 1, the program will sample a number 'n' of alleles in the last generation of the simulation. Then, if the option '**PrintSegSiteInfo**' is set equal to 1, the output of the file created with that option will show information for every mutation that is polymorphic in the sample of n alleles. The output of 'PrintSegSiteInfo' will also show the frequency of the allele in the sample of 'n' alleles instead of the actual population allele frequency. This is useful in case you want to follow the frequency trajectory of alleles conditioning on a particular sample allele frequency using n alleles instead of the population allele frequency.

The argument to LastGenerationAFSamplingValue: must be an integer number equal to 0 or 1. [Default value = 0]. LastGenerationAFSamplingValue: 1 will invoke this option.

Example:

LastGenerationAFSamplingValue: 1

PrintSampledGenotypes: (Optional)
This option was specifically designed for our research on patterns of deleterious variation in dogs (see Marsden et al. 2015 PNAS, doi: 10.1073/pnas.1512501113) where we investigated how different demographic histories affected the ratio of deleterious heterozygous genotypes to neutral heterozygous genotypes.
The program loops through every segregating mutation in the population at the end of the simulation and counts the number of observed genotypes in two different individuals. Genotypes are sampled via multinomial sampling from the population allele frequencies. Note, inbreeding is taken into account here by computing the genotype frequencies with inbreeding.

After iterating through all mutations, the program prints eight numbers in the output file.

A) Replicate number
B) The number of homozygous ancestral genotypes in individual 1
C) The number of heterozygous genotypes in individual 1
D) The number of homozygous derived genotypes in individual 1
E) The number of homozygous ancestral genotypes in individual 2
F) The number of heterozygous genotypes in individual 2
G) The number of homozygous derived genotypes in individual 2

Then, for each mutation, we will sample one allele from individual 1 and another one from individual 2 based on their genotype and we will print:
H) The number of times that the two alleles do not match.

The argument to PrintSampledGenotypes: must be an integer number equal to 0 or 1. [Default value = 0]. PrintSampledGenotypes: 1 will invoke this option.
Outfile suffix: .sampledgenotype_out.txt

Example:
PrintSampledGenotypes: 1

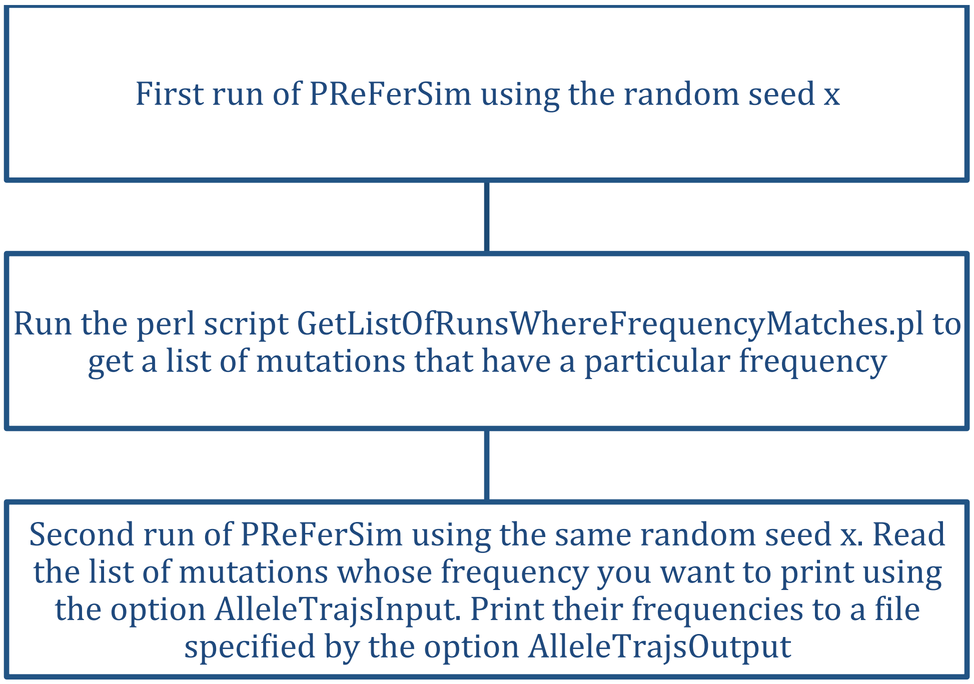
All parameter file examples provide examples specifying print options.

8.6. Printing allele frequency trajectories.

Generating frequency trajectories for alleles that have a particular frequency in the present requires running the program two times. In the first run, we output a list of mutations with their associated allele frequencies by setting the parameter option 'PrintSegSiteInfo:' as 1. Then we run a perl script (*GetListOfRunsWhereFrequencyMatches.pl*) to obtain the allele ID of those mutations whose frequency in the present is equal to a particular range of frequency values we are interested in tracking. We run the program for the second time **using the same parameters and random seed** to output the frequency trajectories of particular alleles.

The script Examples.sh gives two example of how to run all of the above steps to generate the allele frequency trajectories. ParameterFile11.txt and ParameterFile11_B.txt give an example of how to do this conditioning on the population allele frequency, i.e the actual frequency of the allele on the population. On the other hand, ParameterFile13.txt and ParameterFile13_B.txt show how to do this conditioning on the sample allele frequency, i.e. the frequency of the allele in a sample of n alleles.

A general flowchart of how to generate the allele frequency trajectories is shown below:


First run of PReFerSim using the random seed x
Run the perl script GetListOfRunsWhereFrequencyMatches.pl to get a list of mutations that have a particular frequency
Second run of PReFerSim using the same random seed x. Read the list of mutations whose frequency you want to print using the option AlleleTrajsInput. Print their frequencies to a file specified by the option AlleleTrajsOutput

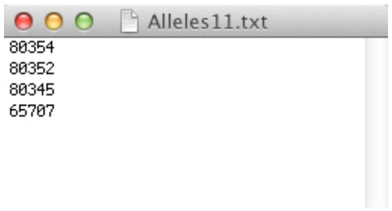
Below we outline the parameter options related to printing allele frequency trajectories

AlleleTrajsInput: (Optional but required to print allele trajectories, must be used in the second run of PReFerSim as outlined in the flowchart above)
This option requires you to add a file with the allele ID of the variants whose allele frequency trajectory you want to follow. This file is needed during the second round of running PReFerSim.

The list of alleles must be composed of nonnegative integer numbers.

Example:
AlleleTrajsInput: MiniTest/Alleles11.txt

An example file in the downloaded folder is: Alleles11.txt



```
80354
80352
80345
65707
```

The perl script GetListOfRunsWhereFrequencyMatches.pl can be used to obtain the input for 'AlleleTrajsInput:'. The perl script requires 4 parameters:

A) The lower range of the frequency values of the alleles whose trajectory you want to follow.
B) The upper range of the frequency values of the alleles whose trajectory you want to follow.
C) The input file which is the output from the first run of PReFerSim when setting the parameter option 'PrintSegSiteInfo:' as 1
D) The output file with the allele IDs of the variants whose allele frequencies you want to follow. This is the output from the first run of PReFerSim, and was generated with the option PrintSegSiteInfo.

Example:

```
perl GetListOfRunsWhereFrequencyMatches.pl 0.0 0.1
MiniTest/Output.11.full_out.txt MiniTest/Alleles11.txt
```

AlleleTrajsOutput: (Optional but required to print allele trajectories, must be used in the second run of PReFerSim as outlined in the flowchart above)
This option specifies the outfile name to print the allele frequency trajectories of the alleles whose allele ID was specified by the option 'AlleleTrajsInput:' Example.- AlleleTrajsOutput: MiniTest/Traj_11.txt

The output of 'AlleleTrajsOutput:' has the following format:

```
65707   0.001000
65707   0.001000
65707   0.003000
```

Where the first column corresponds to the allele ID and the second column corresponds to the frequency in one particular generation. The allele frequencies are displayed from the top to the bottom of the file going from the past to the present. The first appearance of a particular allele ID in the file indicates the frequency of the allele when it emerged in the population in the second column. The present-day population allele frequency for each allele ID is displayed in the second column in the last appearance of the particular allele ID in the file. The number of rows with a particular allele ID indicate the age of that allele. You could use grep to follow the frequency trajectory of one particular allele with the output of 'AlleleTrajsOutput:'. Example.-

```
grep '65707' MiniTest/Traj_11.txt
```

Example parameter files using allele trajectories:
ParameterFile11.txt, ParameterFile11_B.txt, ParameterFile13.txt, ParameterFile13_B.txt

Note how we use the same random seed to get allele frequency trajectories when we run ParameterFile11.txt and ParameterFile11_B.txt in the Examples.sh script. We also use the same random seed in our example when running ParameterFile13.txt and ParameterFile13_B.txt

9. PARALLELIZING SIMULATIONS

If you want to simulate a large number of sites, we recommend that you parallelize your simulations so that results are generated more quickly. E.g. If you want to simulate 10,000,000 sites, you could conduct ten independent simulations of 1,000,000 sites, and then concatenate, average or sum the results depending on the analysis being performed. It is important that each independent simulation has a different random seed and replicate number. As an example, you could run:

```
for i in {1..10}
do
GSL_RNG_SEED=$i GSL_RNG_TYPE=mrg ./PReFerSim ParameterFile1.txt $i
done
```

10. TROUBLESHOOTING

Below we outline some commonly encountered few errors

10.1. Errors when compiling i.e. with this command:

```
gcc -g -o PReFerSim PReFerSim.c -lm -lgsl -lgslcblas -O3
```

Example Error 1:

```
gcc: PReFerSim.c: No such file or directory
```

Error 1 probably means you are trying to compile the program in the wrong folder. You must be in folder with the file PReFerSim.c in order to compile the program.

Alternatively it may mean you have a spelling error in the command, so check that you copied the command exactly.

10.2. Errors running the program i.e. with this command

```
GSL_RNG_SEED=1 GSL_RNG_TYPE=mrg ./PReFerSim ParameterFile1.txt 1
```

Example Error 1:

```
-bash: ./PReFerSim: No such file or directory
```

If you get error 1 when trying to run the program, it means that the c program hasn't been compiled.

Example Error 2:

```
fatal error:

#include <gsl_rng.h>
```

If you get error 2 when trying to run the program, it means that the c program cannot find GSL. You need to make sure the directory GSL is installed and in your path. If using El Capitan, try compiling PReFerSim with this command:

```
gcc -g -o PReFerSim PReFerSim.c -lm -lgsl -lgslcblas -O3 -
I/usr/local/include
```

Example Error 3:

```
Parameter List

MutationRate: 10.000000

Error: Please add the option DFEType and select a particular distribution
("point", "gamma", "lognormal", "beta", "pointprob", "unifbounds")
```

If you get the above error showing the first option of the parameter file is read e.g. MutationRate, but an error stating that the next parameter is missing (DFEType) despite it being present in your parameter file, this typically means your text file is in mac or dos format, not unix format. You need to convert as specified below:

```
tr '\r' '\n' < macfile.txt > unixfile.txt
```

11. Bugs

Please report any bugs on the github website.