`anavar` - a program for estimating selection and mutation parameters from SNPs and INDELs

Kai Zeng

Department of Animal and Plant Sciences, University of Sheffield, UK
k.zeng@sheffield.ac.uk

Version 1.2

1 Introduction

To be added.

2 Citation

Barton and Zeng ....

3 Compiling the program

3.1 Installing GSL

`anavar` was developed using GSL v2.3 (using other versions should not be a problem, but this has not been tested). Compiling and installing GSL are relatively straightforward. Download the package from `www.gnu.org/software/gsl/`. To unpack, issue

```
tar -xf gsl-2.3.tar.gz
```

Enter the folder generated, and issue

```
./configure --disable-shared --prefix=$DIR
```

where `$DIR` should be replaced by the full path of the folder in which the final library files are to be stored (e.g., `--prefix=/home/gsl-2.3`). When the above is done, issue

```
make
```

This will take a little while. When done, issue

```
make check > test.log 2>&1
```

This will carry out tests and save the results to `test.log`. When the process is completed, open `test.log`, and go over it carefully. There should be no errors or failures. Finally, issue

```
make install
```

This will copy the library files to the folder specified by `$DIR`. Within this folder, there should be a folder named `lib`, which contains a file named `libgsl.a`.

3.2 Installing NLopt

Download `NLopt v2.4.2` from `http://ab-initio.mit.edu/wiki/index.php/NLopt` (again using a newer version should be fine, but has not been tested). To unpack, issue

```
tar -xf nlopt-2.4.2.tar.gz
```

Enter the folder created by this command. Issue the following commands one by one:

```
./configure --prefix=$DIR
make
make install
```

where $DIR should be replaced by the full path of the folder in which the library files are to be stored (e.g., `--prefix=/home/nlopt-2.4.2-lib`). There is a folder named `lib` in the folder specified by `$DIR`. There should be a file named `libnlopt.a`.

3.3 Compiling anavar

Unpack `anavar.zip`

```
unzip anavar.zip
```

Then enter the folder. Use a text editor to open `Makefile`. We need to change the values of two variables defined at the top of the file: `GSL` and `NLOPT`. These should be replaced by the values of the `$DIR` variable mentioned above. ***Important:*** enter the full path, but do not end them with a path separator (i.e., '/'). Save the changes and exit. Issue `make`. This should generate an executable named `anavar`.

4 Running the program

```
anavar control_file result_file log_file
```

5 Control file

This should be a plain text file. Lines that are either empty or start with "#"are ignored. Thus, we can structure the contents and add comments easily. Everything is case-sensitive. The commands should appear in exactly the same order as discussed below. Example control files can be found in the folder `example_control_files`.

A control file should look like the following.

```
[algorithm_commands]
search_algorithm:
maxeval:
maxtime:
num_searches:
nnoimp:
maximp:
optional:
$optional_commands

[model_commands]
model:
$model_specific_commands
```

Note that [algorithm_commands] and [model_commands] are section headings and should be retained. There should be no space between a command name and the trailing colon. As detailed below, $optional_commands and $model_specific_commands are to be replaced by commands determined by the values given to optional and model, respectively.

5.1 search_algorithm

The following algorithms are supported:

NLOPT_LD_SLSQP
NLOPT_LD_LBFGS
NLOPT_LD_VAR1
NLOPT_LD_VAR2
NLOPT_LN_NELDERMEAD

Details of the algorithms can be found in the following NLopt page: http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms. Except for the last algorithm (i.e., the Nelder-Mead method), the other algorithms are derivative-based, meaning that they are usually more efficient (for comparison, see Table in the supplementary materials of Barton and Zeng). The Nelder-Mead algorithm, however, has been proved to be surprisingly robust under certain conditions (e.g., when the likelihood surface is ragged). Thus, it is worth trying this algorithm when the others struggle.

Among the derivative-based algorithms, SLSQP, VAR1 and VAR2 seem to perform well, especially SLSQP.

Regardless of which algorithm, it is important to run them multiple times (anavar starts each search at a randomly selected point; see below), and check that the same maximum is returned by multiple searches. It is also advisable to use several algorithms and check the consistency with respect to the MLEs and ln-likelihood. All these should minimise the chances of getting erroneous results due to poor convergence or the existence of multiple local maxima.

5.2 maxeval, maxtime

A positive integer should be given to maxeval (e.g., 100000). It means that the search will be aborted when the likelihood function has been evaluated this number of times but the algorithm is still deemed non-converged. This

stops the algorithm from running for ever.

A positive real number should be given to `maxtime` (e.g., 600). It means that the search will be aborted when the algorithm has been running for the given amount of time (in seconds) but has still not yet converged. This stops the algorithm from running for ever.

In the result file, there is a column named `exit_code`. The returned value indicates under what condition did the algorithm stop. Details of what the returned values mean can be found here: `http://ab-initio.mit.edu/wiki/index.php/NLopt_Reference#Return_values`. In a nutshell, positive values generally indicate a successful stop, and negative values mean failures. In our case, the returned value should normally be 3 (i.e., `NLOPT_FTOL_REACHED`), which means that the algorithm has converged as defined by `rftol` (see below). If the algorithm stopped because it has reached `maxeval` or `maxtime` (`exit_code` = 5 or 6, respectively), then you should definitely increase the values of these parameters and rerun the analysis.

5.3 num_searches

This parameter specifies how many independent searches should be done. Each search starts from a randomly chosen point within the parameter space defined by the range parameters (see below). In the result file, there will be `num_searches` rows, ordered by the ln-likelihood, each showing the results obtained from an independent search (and any subsequent improvement searches (see below)). It is hard to tell how many rounds of searches are needed, because this is dependent on both the algorithm and the data in question. But using 100 for derivative-based algorithms and 500 for derivative-free algorithms are reasonable. If the best result is not found by multiple searches, then increase the value or try a different algorithm.

5.4 nnoimp, maximp

`anavar` starts a new, independent search from a randomly chosen starting point. When this search converges, we obtain the current best guess of the maximum likelihood estimates (MLEs). If we are using a derivative-free algorithm (e.g., the Nelder-Mead algorithm), it is often a good idea to *improve* this current best guess by starting a new search using the current best guess as the starting point. Iterating this process (i.e., starting the next search by using the result of the current search as a starting point) will frequently lead

to significant improvement on the result (in terms of the ln-likelihood).

A positive integer should be given to `nnoimp`. If it is set to 1, then no improvement search is attempted. Otherwise, improvement searches will be conducted until `nnoimp` number of consecutive improvement attempts lead to no significant improvement in the ln-likelihood, as determined by `rftol` (see below).

At least in the tests, setting `nnoimp` to 1 seems to be sufficient for the derivative-based search algorithms. Whether this is true for your dataset can be found out easily by looking at the log file, which records the progress of the search.

`maximp` is a positive integer (e.g., 50), and specifies the maximum number of improvement searches.

5.5 `optional`

Either `true` or `false` should be given. If `true`, then `$optional_commands` should be replaced by:

```
size:
key:
epsabs:
epsrel:
rftol:
```

If `false`, then `$optional_commands` should simply be omitted from the control file, and the default values for the commands listed above are used. Setting `optional` to `false` should suffice in most cases.

5.5.1 `size`, `key`, `epsabs`, `epsrel`

These parameters controls the QAG adaptive integration algorithm in GSL (`https://www.gnu.org/software/gsl/manual/html_node/QAG-adaptive-integration.html`). If `optional` is `false`, the following default values are used: `size` = 10000, `key` = 6, `epsabs` = 1e-50, and `epsrel` = 1e-10.

5.5.2 `rftol`

This controls when the search algorithm should stop. `rftol` should be a small value (e.g., the default is 1e-10). It means if the values of the ln-likelihood between two consecutive search steps differ by less than the given value (in

relative terms), then the algorithm is considered to have converged. We can use larger values, which will allow quicker convergence. Also, `rftol` should not be set to values smaller than the machine precision (2×10^{-16} or 2^{-52} for most machines); otherwise, time will be wasted on unnecessary iterations.

5.6 `model`

This should take one of the following values:

`SNP_1`

The models and their `$model_specific_commands` are explained in the next section.

6 The models

6.1 `SNP_1`

This model analyses exactly one set of SNPs. `$model_specific_commands` should contain the following commands:

```
n:
m:
folded:
sfs:
dfe:
$dfe_related_commands
```

These commands are explained below.

`n`
An integer should be given to `n`. This is the sample size. The minimum sample size is 4.

`m`
`m` is for setting the number of sequenced sites (i.e., both poly- and monomorphic). The estimated θ is defined as $4Nu$, where N is the effective population size and u is the mutation rate per site per generation. However, if the interest is to estimate the overall mutation rate for the region, then `m` can be set to 1. In this case the estimated θ is defined as $4Num$.

`folded`
If `true`, then the folded site-frequency spectrum (SFS) is used. If `false`, the unfolded SFS is used.

`sfs`
The SFS. If folded, then there should be $\lfloor n/2 \rfloor$ elements. If unfolded, there should be $n-1$ elements. The elements can take real values. That is, they are no necessarily integers, provided that they are all non-negative.

`dfe`
Either `discrete` or `continuous` can be given.

6.1.1 `dfe: discrete`

It is assumed that there are several different types of sites within the sequenced region. Each type of sites has a set of three parameters: θ, γ, and e, where $\gamma = 4Ns$ and e is the polarisation error rate. In the output, the parameters for the first type of sites are named `theta_1`, `gamma_1`, and `e_1`; the parameters for the other site types are specified similarly. The total mutation rate of the sequenced region is equal to $m \sum_i \theta_i$. If `m` is set to 1, and if we assume that the per-site mutation rate is uniform across sites, then the θ_i's are proportional to the number of sites belong to each of the site types. The corresponding `$dfe_related_parameters` are:

```
c:
theta_range:
gamma_range:
e_range:
constraint:
```

These commands are explained below.

`c`
An positive integer, specifying the number of site types.

`theta_range`
Two positive numbers should be given, separated by a comma. The first value should be strictly smaller than the second. They specify the range within which the search algorithm will try to look for the maximum likelihood estimate. The initial value of θ is randomly generated from this interval. This range is used in the analysis of all site types. A useful way to determine whether the range is appropriate is to calculate the per-site nucleotide diversity π from the data. If `m` is set to one, then multiple π by `m`. Setting the

range to something like [πm/500, πm $*$ 500] may be reasonable. If the search algorithm keeps returning results close to the boundaries, then increase the range and repeat the analysis to double check that the true global maximum is within the range. Increasing the size of the range and see whether the results are affected is generally a good way to check the robustness of the results.

`gamma_range`
Two numbers should be given, separated by a comma. Positive and negative values of γ signify positive and negative selection respectively. Something like [-500, 100] should generally be sufficient.

`e_range`
Two numbers should be given, separated by a comma. Because e is a probability, its range should be within the unit interval [0, 1]. In most cases, using the unit interval suffices.

`constraint`
This is for specifying whether any constraints are to be put on the parameters. One of the following values must be given:

`none`
`no_pol_error`

With `none`, no constraint is added, and the full model with θ_i, γ_i, and e_i as free parameters is to be fitted to the data.

With `no_pol_error`, all the e's are fixed to zero, meaning that there is no polarisation error. This model can be compared to the full model with a chi-squared test with `c` degrees of freedom. Note that even in this case, `e_range` should be included, although the values provided are not used.

6.1.2 `dfe: continuous`

This assumes that the distribution of fitness effects of new mutations follows a continuous distribution with parameters ζ. The parameters are ordered as follows: θ, ζ, e. If `m` is set to 1 or the number of sequenced sites, then θ is the total scaled mutation rate for the region or the per-site scaled mutation rate, respectively. ζ can contain more than one parameters. e is the polarisation error rate. The corresponding `$dfe_related_parameters` are:

```
distribution:
$distribution_related_parameters
```

The values can be supplied to `distribution` are:

`reflected_gamma`

6.1.2.1 `distribution: reflected_gamma`

It is assumed that all new mutations are deleterious, and their fitness effects (i.e., γ) follow a Gamma distribution specified by the shape and scale parameters (i.e., ζ represents the shape and scale parameters). The corresponding `$distribution_related_parameters` are:

```
theta_range:
shape_range:
scale_range:
e_range:
constraint:
optional:
$optional_commands
```

The usage of `theta_range` and `e_range` is the same as above. `shape_range` and `scale_range` specify the range of the shape and scale parameters of the Gamma distribution, respectively. Both parameters are strictly positive. So the lower bounds of their ranges should also be strictly positive. Note that the mean of a Gamma distribution is determined by the product of its shape and scale parameters (i.e., $\bar{\gamma}$ = shape * scale).

`constraint`

Either `none` or `no_pol_error` can be used, meaning that no constraint is put on any parameters of the model, or that e is fixed at zero.

`optional`

This is not to be confused with the `optional` command in the `algorithm_commands` section. Either `true` or `false` can be used. If `false`, then the default values for optional parameters are used and `$optional_commands` should be omitted. If `true`, then we should replace `$optional_commands` with the following commands:

```
fraction:
delta:
degree:
```

The first two optional commands are for controlling the step size, h, used in numerical differentiation (i.e., $f'(x) \approx [f(x+h/2) - f(x-h/2)]/h$; note that the actual algorithm is more sophisticated than this; see the GSL documentation for details of the algorithm). h is defined as $min(x * fraction, delta)$.

The default values for `fraction` and `delta` are 0.001 and 1e-7, respectively.

The integration over the Gamma distribution is approximated by an n-degree Gaussian quadrature. The default value for `degree` is 50. Using larger values tend to make the calculation less efficient, whereas using smaller values tend to give biased results. The extent of bias is usually rather small. One way to find out whether the extent of bias is acceptable is to use the Mathematica notebook provided to generate the expected SFS for given values of the shape and scale parameters, and estimate these parameters from this SFS using Gaussian quadratures with different degrees.

6.2 INDEL_1

This model analyses exactly one set of INDELs. `$model_specific_commands` should contain the following commands:

```
n:
m:
ins_sfs:
del_sfs:
dfe:
$dfe_related_commands
```

These commands are explained below.

`n`
An integer should be given to `n`. This is the sample size. The minimum sample size is 4.

`m`
`m` is for setting the number of sequenced sites (i.e., both poly- and monomorphic). The scaled mutation rates for insertions and deletions are denoted by θ_{ins} and θ_{del}, respectively. As in model `SNP_1`, these are defined on a per-site basis. However, if the interest is to estimate the overall mutation rate for the region, then `m` can be set to 1. In this case the estimated θ is defined as $4Num$.

`ins_sfs`
This is the SFS for insertions. There should be `n - 1` elements.

`del_sfs`
This is the SFS for deletions. There should be `n - 1` elements.

`dfe`
Either `discrete` or `continuous` can be given.

6.2.1 `dfe: discrete`

It is assumed that there are several different types of sites within the sequenced region. Each type of sites has a set of six parameters: θ_{ins}, γ_{ins}, e_{ins}, θ_{del}, γ_{del}, and e_{del}. In the output, the parameters for the first type of sites are named `ins_theta_1`, `ins_gamma_1`, `ins_e_1`, `del_theta_1`, `del_gamma_1`, and `del_e_1`; the parameters for the other site types are specified similarly. The total insertion mutation rate of the sequenced region is equal to $m \sum_i \theta_{ins}^{(i)}$. If `m` is set to 1, and if we assume that the per-site mutation rate is uniform across sites, then the $\theta_{ins}^{(i)}$'s are proportional to the number of sites belong to each of the site types. The corresponding `$dfe_related_parameters` are:

```
c:
ins_theta_range:
ins_gamma_range:
ins_e_range:
del_theta_range:
del_gamma_range:
del_e_range:
constraint:
```

The usage of `c` and the range commands is analogous to that in model `SNP_1`.

`constraint`
This is for specifying whether any constraints are to be put on the parameters. One of the following values must be given:

`none`

With `none`, no constraint is added, and the full model is to be fitted to the data.

6.2.2 `dfe: continuous`

This assumes that the distribution of fitness effects of new insertions and deletions follows continuous distributions specified by parameters ζ_{ins} and ζ_{del}, respectively. In the output file, the parameters are ordered as follows: `ins_theta`, `ins_`ζ, `ins_e`, `del_theta`, `del_`ζ, and `del_e`. ζ can contain more than one parameters. e is the polarisation error rate. The corresponding `$dfe_related_parameters` are:

```
distribution:
$distribution_related_parameters
```

The values can be supplied to `distribution` are:

`reflected_gamma`

6.2.2.1 `distribution: reflected_gamma`

It is assumed that all new mutations are deleterious, and their fitness effects (i.e., γ) follow a Gamma distribution specified by the shape and scale parameters (i.e., ζ represents the shape and scale parameters). In the full model, insertions and deletions are assumed to follow different Gamma distributions (i.e., ζ_{ins} and ζ_{ins} are different). The corresponding `$distribution_related_parameters` are:

```
ins_theta_range:
ins_shape_range:
ins_scale_range:
ins_e_range:
del_theta_range:
del_shape_range:
del_scale_range:
del_e_range:
constraint:
optional:
$optional_commands
```

The usage of the range commands are analogous to that described in `SNP_1`.

`constraint`

Only `none` can be used, meaning that no constraint is put on any parameters of the model.

`optional`

This is not to be confused with the `optional` command in the `algorithm_commands` section. Either `true` or `false` can be used. If `false`, then the default values for optional parameters are used and `$optional_commands` should be omitted. If `true`, then we should replace `$optional_commands` with the following commands:

```
fraction:
delta:
degree:
```

The usage of the range commands and their default values are the same as those described in `SNP_1`.