BioNetFit
User Manual
Last updated October 14, 2015

http://bionetfit.nau.edu

bionetgen.help@gmail.com

Brandon R. Thomas[1],
Lily A. Chylek[2,1],
Joshua Colvin[1],
Suman Sirimulla[3],
Andrew H.A. Clayton[4],
William S. Hlavacek[5],
Richard G. Posner[1]

[1]Department of Biological Sciences, Northern Arizona University, Flagstaff, Arizona

[2]Department of Chemistry and Chemical Biology, Cornell University, Ithaca, New York

[3]Department of Basic Sciences, Saint Louis College of Pharmacy, Saint Louis, Missouri

[4]Center for Microphotonics,, Faculty of Engineering and Industrial Sciences, Cell Physics Laboratory, Swinburne University of Technology Hawthorn, Victoria, Australia.

[5]Theoretical Biology & Biophysics Group, Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico, USA

Table of Contents

I. Introduction

BioNetFit is a fitting tool compatible with two simulation packages for rule-based models: BioNetGen (http://bionetgen.org), and NFsim (http://emonet.biology.yale.edu/nfsim/). BioNetFit can be used on a standalone platform (Mac, Windows/Cygwin, and Linux) or a Linux-based cluster running SLURM, Torque/PBS, or SGE. Information about these resource management tools can be found at the following websites:

http://slurm.schedmd.com/
http://www.adaptivecomputing.com/products/open-source/torque/
http://gridscheduler.sourceforge.net/

BioNetFit installation instructions are provided below. We have successfully installed and run BioNetFit on numerous standalone and cluster platforms. Please let us know if you encounter any difficulties getting BioNetFit up and running.

BioNetFit is written in Perl. The BioNetFit website is http://bionetfit.nau.edu. This user manual includes step-by-step instructions for running example fitting jobs. The files needed to run each of these fitting jobs are included in the BioNetFit distribution in the "examples" directory. Fpir examples, "example1," "example2," "example3," and "example4" are computationally expensive. We recommend running these example fitting jobs on a cluster. The files required to run these fitting jobs are located in the "example1" through "example4" subdirectories. These files are set up for running on a cluster. The other examples, "example5" and "example6," can be run on a laptop. For more details about the examples, see the README.txt file located in the "examples" directory.

This user guide includes extensive documentation of the BioNetFit configuration (.conf) file. It also explains how to preprocess a BioNetGen/NFsim input (.bngl) file to prepare it for use in fitting and how data input (.exp) files are used and are related to the .bngl and .conf file.

BioNetFit was developed primarily by Brandon Thomas (brandon.thomas@nau.edu). It is supported by the Posner and Hlavacek laboratories. For help, write to bionetgen.help@gmail.com. Emails sent to this address will reach all BioNetFit developers as well as the larger BioNetGen developer community. If BioNetFit is useful for you, please let us know and please cite the reference below.

Thomas BR, Chylek LA, Colvin J, Sirimulla S, Clayton AHA, Hlavacek WS, Posner RG (2015) BioNetFit: a fitting tool compatible with BioNetGen, NFsim, and distributed computing environments.

Author contributions: BRT and JC wrote the code. RGP and WSH initiated the development of BioNetFit. The software was designed by BRT, JC, LAC, WSH, and RGP. LAC and SS extensively tested the software and assisted in debugging. AHAC assisted in the design of demonstration fitting runs. BRT, WSH, LAC, and RGP wrote the BioNetFit documentation.

II. Quick start: step-by-step instructions for running the example fitting jobs

BioNetFit can be used on the following stand-alone platforms: Mac OS X, Windows/Cygwin, and Linux. Perl must be installed. Additionally, BioNetFit can be used on clusters with any of the following resource management tools installed: SLURM, Torque/PBS, or SGE. Below are instructions for using BioNetFit to perform a fitting job. The examples directory of the BioNetFit distribution includes four subdirectories: example1, example2, example3, example4, example5 and example6. Each subdirectory includes a collection of files needed to perform a fitting job. For a description of these files, see the README.txt file in the examples directory. The first four jobs (example1 through example4) are computationally expensive; use of a cluster is recommended for these jobs. The last two jobs (example5 and example6) can be run on a laptop or desktop workstation. Detailed instructions for installing and configuring BioNetFit are described after this section.

II.A. Installing and running on a laptop or desktop workstation

1) Download BioNetFit from http://bionetfit.nau.edu
2) Unzip and place the BioNetFit directory in your file system at a location of your choosing.
3) BioNetFit is distributed with NFsim v1.11 and BioNetGen v2.2.6 and is automatically configured to use these simulation tools.
4) If you'd like to configure BioNetFit to use simulators different from those included in the BioNetFit directory (e.g., pre-existing copies of NFsim or BioNetGen), you must follow the following procedure (for example3) or a very similar one (for example4):
    a. Use a text editor to open the BioNetFit configuration (.conf) file example3.conf, which is located in BioNetFit/examples/example3/, and change the path option for *bng_command* to point at a copy of BNG2.pl (the Perl script called to start BioNetGen) in your file system.
    b. At this time, you may change pre-set configuration options, but there should be no need to do so.
    c. Save the .conf file.
5) Open a terminal window (e.g., on a Mac launch the Terminal application, which is located in the Utilities folder within the Applications folder).
6) Be sure that BioNetFit.pl has the necessary permissions to run. This can be accomplished with the command:

    ```
    chmod +x /path/to/BioNetFit/BioNetFit.pl
    ```

7) Start the fitting run with the following command:

```
perl /path/BioNetFit/BioNetFit.pl /path/BioNetFit/examples/example3/example3.conf
```

    In this command, each path part should be replaced with the appropriate path to BioNetFit.pl or example3.conf in your file system.

8) The fitting run will begin, and terminal output reporting job progress will appear in the terminal window.
9) When the fitting run is finished, the terminal output will indicate the directory that contains the results. This directory will contain the BioNetFit configuration (.conf) file, a BNGL model-specification (.bngl) file in which free parameters are assigned the best-fit values found by BioNetFit, a simulation output (.gdat) file containing results from a simulation based on best-fit parameter values, and a .txt file containing both these simulation results and the time-series data used in fitting.
10) These files are all plain-text files, which can be opened and viewed in a text editor or a spreadsheet program, for example.

11) RuleBender (an IDE for BioNetGen and NFsim) can be used to run the .bngl file and to visualize the simulation results. RuleBender can be downloaded from http://bionetgen.org/index.php/Download.

II.B. Installing and running on a cluster

1) Open a terminal window on your laptop or desktop workstation.
2) Login to your cluster using the instructions provided by your cluster administrator. An example login command might take the following form:
```
      ssh username@cluster_address
```
   Replace *username* with your cluster username and *cluster_address* with the address of the cluster.

3) Download BioNetFit.zip to the cluster using the following command:
```
      wget http://downloads.sourceforge.net/project/nauBioNetFit/BioNetFit.zip
```

4) If it's not possible to download BioNetFit.zip from the Internet to the cluster using the above command, it may be possible to transfer BioNetFit.zip to the cluster from your laptop or desktop workstation:
   a. Download BioNetFit.zip to your computer as described in Section II.A.
   b. Open a separate terminal window.
   c. Run the command
```
      scp /path/to/downloaded/BioNetFit.zip username@cluster.address:
```

   Replace */path/BioNetFit.zip* with the path to your downloaded copy of BioNetFit.zip, replace *username* with your cluster username, and replace *cluster_address* with the address of the cluster.

5) In the cluster terminal window, unzip BioNetFit.zip using the command
```
unzip BioNetFit.zip
```

6) Navigate to the newly created BioNetFit directory with the command:
```
cd BioNetFit
```

7) If you wish to modify BioNetFit configuration (.conf) file options, open BioNetFit/examples/example1/example1.conf or BioNetFit/examples/example2/example2.conf using a text editor such as nano, vim, or emacs:
```
      nano examples/example1/example1.conf
      vim examples/example1/example1.conf
      emacs examples/example1/example1.conf
```

8) Exit and save the file. For example, in nano, the command to save and exit is [CTRL]+x
9) Run BioNetFit with the command:
```
perl BioNetFit.pl examples/example1/example1.conf
```

10) The fitting run will begin, and terminal output regarding job progress will appear in the terminal window.
11) When the fitting run is finished, the terminal output will indicate the directory containing job results. This directory will contain the BioNetFit configuration (.conf) file, a BNGL model-specification (.bngl) file with best-fit values for free parameters, a simulation output (.gdat) file containing results from a

simulation based on the best-fit parameter values, and a whitespace-delimited .txt file containing both results from the simulation run and the time-series data used in fitting.

12) The fitting results can be copied from the cluster to your laptop or desktop workstation using the *scp* command.
    a. Open a separate terminal window on your laptop or desktop workstation
    b. Run the command

```
scp -r username@cluster.address:BioNetFit/output/example1/* /directory
```

Replace */directory* with the path to the location where you wish to create a copy of the example1 output directory, replace *username* with your cluster username , and replace *cluster_address* with the address of the cluster.

13) The result files are all plain-text files and can be opened and viewed in a text editor or a spreadsheet program, for example.
14) RuleBender (an IDE for BioNetGen and NFsim) can be used to perform simulations and to visualize the results. RuleBender can be dow**n**loaded from http://bionetgen.org/index.php/Download.

III. Installation

BioNetFit supports Windows/Cygwin 32/64 bit, Mac OS X 32/64 bit, and Linux 32/64 bit systems with Perl installed. It can also be used on clusters with SLURM, SGE, or Torque/PBS installed.

Unzip BioNetFit.zip in a directory of your choosing.

Once all files are unzipped, check to make sure your BNG2.pl, BioNetFit.pl, and the appropriate NFsim executable for your platform have permission to run. These files can be made executable, if necessary, with the following commands:

```
$ chmod +x /path/to/BioNetFit/BioNetFit.pl
$ chmod +x /path/to/NFsim/BNG2.pl
$ chmod +x /path/to/NFsim/bin/*
```

IV. Configuration

IV.A Data input (.exp) files

Data input files should each have the filename extension .exp. These files have the same format as BioNetGen's .gdat files (Faeder et al., 2009).

Multiple .exp files may be provided by a user for use in a fitting run. If multiple .exp files are provided, BioNetFit will perform a global fit, i.e., BioNetFit will attempt to align model predictions with the data in all of the .exp files simultaneously. As is typically the case with global fitting, the data to be used may require pre-processing so that one dataset is not given a disproportionate weight relative to the other datasets. Various pre-processing strategies could be used. For example, one could linearly scale time series data such that each time series has an average measurement value of 1.

IV.A.1 Formatting considerations

A data input (.exp) file should contain at least two whitespace-separated columns of numerical entries. There is no limit on the number of columns allowed. The first row of a column should be a label or header. The header row must begin with a hash (#). After the first row, additional rows should contain numerical entries.

Each such row must begin with a whitespace. For an .exp file containing time-course data, there would be at least two columns. The first column would indicate the times at which measurements were made, and the second column would indicate the measurement values at those time points. A third column is optional. In other words, a user may provide *x-y* data, or *x-y-z* data. The third column is provided if a user wishes to perform weighted non-linear least-squares fitting, which a user indicates by selecting the chi-square function as the objective function to be minimized during fitting. This objective function is selected by setting *objfunc=2* in the BioNetFit configuration (.conf) file. The column header for the third column should be the same as the header for the second column, but with the appended text "_SD." This suffix indicates that the second column consists of means from replicate experiments, and that the third column consists of sample standard deviations. As noted earlier, the number of columns is not limited to two or three. In fact, it is desirable for there to be a column or pair of columns for each readout made at the indicated *x* values. Examples of .exp files are provided below to illustrate the possible formats.

Example 1:
```
#      x      y
       0      0.8
       1      1.0
       2      1.2
```
Example 2:
```
#      time   y
       0      8.0e-1
       1      1.0e0
       2      1.2e0
```
Example 3:
```
#      x      y      y_SD
       0      0.8    0.2
       1      1.0    0.1
       2      1.2    0.2
```
Example 4:
```
#      x      y1     y2     y3
       0      0.8    0.9    0.1
       1      1.0    1.0    0.4
       2      1.2    1.1    2.5
```
Example 5:
```
#      x      y1     y1_SD  y2     y2_SD  y3     y3_SD
       0      0.8    0.2    0.9    0.05   0.1    0.1
       1      1.0    0.1    1.0    0.1    0.4    0.1
       2      1.2    0.2    1.1    0.15   2.5    0.8
```
Example 6:
```
#      x      y1     y2     y3     y1_SD  y2_SD  y3_SD
       0      0.8    0.9    0.1    0.2    0.05   0.1
       1      1.0    1.0    0.4    0.1    0.1    0.1
       2      1.2    1.1    2.5    0.2    0.15   0.8
```
Example 7:
```
#      x      y1     y2     y3
       0      0.8    0.9    0.4
       1      NaN    1.0    1.6
       2      1.2    1.1    NaN
```

In Example 2, we have used the header "time" instead of the abstract header *"x."* In the case of any time series, the first column of numerical entries must be labeled with the header "time." Any other header should be taken to be indicative of steady-state dose-response data. The reason that "time" must be used in the case of time-series data is that "time" is used as the header in .gdat files. A *simulate* action command passed to BioNetGen or NFsim produces a .gdat file in which the first column of numerical entries is labeled "time." The general principle is that headers in .exp files must also appear as headers in corresponding .gdat and .scan files.

Example 7 is intended to illustrate how missing data should be handled. The y_1 and y_3 columns each contain an instance of 'NaN' instead of a numerical entry. The 'NaN' entry indicates the absence of a measurement value. Missing data do not enter into the evaluation of the user-selected objective function.

IV.A.2 Linking experimental data to simulation output

It is desirable for each .exp file to contain as much data as possible. The reason is that there will be a simulation call made for each .exp file every time the objective function is evaluated. Thus, if a user provides *N* .exp files, every evaluation of the objective function during a fitting run will involve *N* simulation calls and *N* simulation runs. The simulation calls must appear in the *actions* block of the model-specification (.bngl) file provided by the user. (The user-supplied .bngl file will be discussed below.) These simulation calls have the standard syntax of BioNetGen language (BNGL) action commands (Faeder et al., 2009). Generally, whenever a simulation call is specified within a .bngl file, there will be an output file created that is associated with the call. If the simulation call is made using the action command *simulate*, then the output file will have the filename extension ".gdat." On the other hand, if the simulation call is made using the action command *parameter_scan*, the output file will have the filename extension ".scan." The *parameter_scan* action command is a derivative of the *simulate* action command; it performs a series of simulations to generate a steady-state dose-response curve (Faeder et al., 2009). In BNGL, one may append a suffix to a simulation output file (Faeder et al., 2009). BioNetFit expects simulation output files to have suffices that correspond to filenames of .exp files. Thus, if a user supplies time-course data in a file named "timeseries.exp" and steady-state dose-response data in a file named "doseresponse.exp," two action commands having the following forms should be included in the user-supplied .bngl file:

```
simulate({suffix=>”timeseries”,…})
parameter_scan({suffix=>”doseresponse”,…})
```

The above requirement is just one example of how .exp files provided by a user impose requirements on the content of the .bngl file provided by the user. All such requirements are delineated below in our discussion of BNGL model-specification (.bngl) files.

IV.B Model-specification (.bngl) files

IV.B.1 Compatibility

BioNetFit is compatible with BNGL, a language for specifying rule-based models (Faeder et al., 2009). There are several BNGL-compatible simulators available, including BioNetGen (Faeder et al., 2009; Harris et al., submitted "BioNetGen 2.2: advances in rule-based modeling") and NFsim (Sneddon et al., 2011). BioNetFit is designed to work with these simulators. BioNetGen provides deterministic, stochastic, and hybrid simulation capabilities. With BioNetGen, simulations must be preceded by network generation (i.e., the *generate_network* action command must be invoked before issuing a simulation command). Network generation is the process of translating rules into a list of reactions (Faeder et al., 2009). NFsim provides network-free stochastic simulation capabilities. With NFsim, network generation is not required (Sneddon et al., 2011). It should be noted that .bngl files can be viewed not only as model-specification files but also as input files for BioNetGen and/or NFsim. An NFsim input file is actually an .xml file, but this file is generated by BioNetGen from a .bngl file and NFsim can be invoked using a BNGL action command, such as *simulate({method=>"nf",…})*. A list of available action commands (and their arguments) can be found at the

BioNetGen wiki site (http://bionetgen.org). See also Chylek et al. (2015) [Modeling for (physical) biologists: an introduction to the rule-based approach. *Phys. Biol.* 12:045007].

Here, for the most part, we assume that users are familiar with rule-based modeling, BNGL, and use of BNGL-compatible simulators. Several excellent reviews are available about rule-based modeling and the methods and software tools that enable rule-based modeling (Chylek et al., 2013; 2014; 2015; Stephan et al., 2014).

Importantly, if a user is making a simulation call to NFsim, we highly recommend that the *get_final_state* argument be set to 0, which will avoid the computationally expensive (default) task of generating a list of species in the system at the end of simulation. This list is not used by BioNetFit.

IV.B.2 Consistency between .bngl and .exp files

As mentioned above, the user-supplied .exp files and .bngl file must be consistent with each other. There are three issues that require special attention.

First, as discussed above, a simulation call must be included in the .bngl file for each .exp file.

Second, a simulation call must generate outputs at each report time that is included in the corresponding .exp file. Thus, if an .exp file includes measurements at t1, t2 and t3, the action command that invokes a simulation must generate outputs at exactly t1, t2, and t3. There are several ways that this requirement may be satisfied. We recommend that report times be specified explicitly. When the BNGL action command *simulate* is used, report times may be specified using the optional *sample_times* argument (http://bionetgen.org/index.php/BioNetGen_Actions_and_Arguments). For the example .exp files presented above (Examples 1-7), report times may be specified as follows:

```
simulate({suffix=>"basename_of_exp_file",sample_times=>[0,1,2],…})
```

Alternatively, one could accomplish the same objective as follows:

```
simulate({suffix=>"basename_of_exp_file",t_start=>0,t_end=>2,n_steps=>3,…})
```

If simulations are invoked with the *parameter_scan* action command, outputs at particular parameter (*x*) values may be requested as follows:

```
parameter_scan({suffix=>"basename_of_exp_file",parameter=>"x",log_scale=>0,\
par_min=>0,par_max=>2,n_scan_points=>3,…})
```

In the above action command, note that a parameter is specified. This parameter has the name *x*. This name must correspond exactly to the header of the first column of numerical entries in the .exp file that corresponds to the action command. It must also be introduced in the *parameters* block of the .bngl file. The parameter would typically be a copy-number or concentration parameter that defines the total amount of a biomolecule considered in a model (e.g., the total amount of a ligand in a model for ligand-receptor interaction).

Third, for each measurement considered in .exp files, there must be a corresponding simulation output. The label of each of these simulation outputs must correspond exactly to a header in the .exp file. Thus, if a column is labeled "*y*" in an .exp file, the .bngl file must define an output also labeled "*y*." In BNGL, simulation outputs may be defined as either observables (Faeder et al., 2009) or as functions (Sneddon et al., 2011; Chylek et al., 2014). Note that the names of observables and functions are used as the corresponding headers in output files. If an output is defined as a function then the action command that invokes a simulation must include the *print_functions* argument (so that function evaluations will be reported in output files). This argument has a default setting of 0 (no reporting of function evaluations). It must be manually set to 1 as follows:

```
simulate({…,print_functions=>1,…})
```

or

```
parameter_scan({…,print_functions=>1,…})
```

Recall our earlier discussion about labeling of the first column of numerical entries in .exp files. This column must be labeled in accordance with the headers of simulation output files. If a simulation call generates a time course, the first column of numerical entries will be labeled "time," and the output file will have a .gdat filename extension. If a simulation call generates a steady-state dose-response curve, the first column of numerical entries will be labeled with the name of the (copy-number or concentration) parameter specified in the *parameter_scan* action command, and the output file will have a .scan filename extension.

To recap, the .bngl file supplied by the user must be consistent with the .exp file(s) supplied by the user in three ways: 1) the .bngl file must include a simulation call for each .exp file, 2) simulation outputs must be generated for each time point or (copy-number or concentration) parameter value listed in the first column of numerical entries in each .exp file, and 3) simulation outputs must be generated and labeled to match the headers of measurement values in .exp files. We note that the .bngl file is allowed to specify simulation outputs that do not correspond to any measured values considered in .exp files. Similarly, outputs can be generated for report times or parameter values not included in .exp files.

IV.B.3 Consistency between .bngl and .conf files

The .bngl file supplied by the user for a fitting job must be consistent not only with .exp files but also the BioNetFit configuration (.conf) file. As discussed below, the .conf file identifies the parameters that are allowed to vary in fitting. In the .conf file, a user may simply list the names of parameters that are free to vary. However, we recommend that the user instead provide a list of identifiers in the .conf file and then link parameter names in the .bngl file to the identifiers. For clarity, identifiers should have names that are derived from parameter names. For example, for parameter *p,* a user could link its name *p* to an identifier *p__FREE__*. Linking of parameter names to identifiers is done in the *parameters* block of the .bngl file (Faeder et al., 2009). We note that there are restrictions regarding the use of symbols for parameters in mathematical expressions in the .bngl file. Basically, parameters that will be free to vary in fitting should not be used within mathematical expressions. This restriction arises because BioNetFit currently has limited parsing capabilities. In summary, we recommend that a free parameter *p* be introduced in the *parameters* block of a .bngl file as follows:

```
begin parameters

p = p__FREE__

end parameters
```

The identifier, *p__FREE__* in the above example, should appear only once, and the linking of a parameter name to an identifier should appear before the parameter name is used anywhere else in the .bngl file. Thus, we ask users to replace parameter value assignments, such as *p = 1,* with links of parameter names to identifiers as in the above example. If one follows the recommended conventions, then the identifier of a free parameter with name *p* should appear in the .conf file as *p__FREE__*.

We expect that a user will typically need to edit a .bngl file prior to performing a fitting run. Below, we provide recommendations for editing the *parameters* block of a .bngl file.

Example 8:

Before	After
`begin parameters`	`begin parameters`
`p 0.1 # /s`	`p = p__FREE__`
`end parameters`	`end parameters`

Example 9:

Before	After
`begin parameters`	`begin parameters`
`q 0.1 # /s`	`q 0.1 # /s`
`p = q`	`p = p__FREE__`
`end parameters`	`end parameters`

Example 10:

Before	After
`begin parameters`	`begin parameters`
`NA 6.02214e23 # molecules per mol`	`NA 6.02214e23 # molecules per mol`
`V 1e-12 # L per cell`	`V 1e-12 # L per cell`
`p = 1e6/(NA*V) # /M/s converted to /molecule/cell/s`	`p = p__FREE__/(NA*V) # /molecule/cell/s`
`end parameters`	`end parameters`

In the Before situation of Example 8, parameter *p* is simply assigned a value. In the After situation, it is designated as a free parameter. In the Before situation of Example 9, the parameter *p* is assigned the same values as parameter *q*. In the After situation, *p* is designated as an independent free parameter. In the Before situation of Example 10, *p* is assigned a value in the SI unit system and a unit-conversion operation is performed. In the After situation, *p* is designated as a free parameter, which is taken to have a value in the SI unit system, and a unit-conversion operation is performed.

The conventions discussed above are recommended because a parameter is often represented symbolically at multiple places within a .bngl file. With these conventions, essential edits are minimized. Once a parameter name *p* is mapped to an identifier *p__FREE__*, for example with the line *p = p__FREE__*, subsequent uses of the symbol *p* in the starting .bngl need not be changed.

There are two minor restrictions on normal BNGL usage that a BioNetFit user must keep in mind. Normally, one is allowed to append a prefix to an output file. This feature is disallowed when using BioNetFit because BioNetFit expects the default basename and appending a prefix essentially changes the basename. Secondly, if use of NFsim requires an .rnf file (Sneddon et al., 2011), the simulation call that invokes NFsim must take a particular form. (An .rnf file is sometimes used to perform an equilibration simulation before a production simulation; it can also be used for other purposes.) The BioNetFit-compatible simulation call has the following form:

```
simulate({…method=>"nf",param=>"-rnf /path/to/rnf/file.rnf",…})
```

The following equivalent call is normally allowed, but is disallowed when using BioNetFit:

```
writeXML()
simulate({…method=>"nf",…})
```

We note that both simulation calls illustrated above require BioNetGen version 2.2.6. If an earlier version of BioNetGen is being used, it may be necessary to replace *simulate* with *simulate_nf*. We recommend that the most current version of BioNetGen always be used.

When using an .rnf file, one should note that the simulation command arguments *t_start*, *t_end*, and *n_steps* are not parsed by the simulator. However, these arguments are nonetheless required to appear in the simulation command. The simulator will ignore the argument values given in the simulation call and instead use the argument values provided in the .rnf file.

IV.C The BioNetFit configuration (.conf) file

The BioNetFit configuration (.conf) file is a plain-text file that allows a user 1) to specify parameters of the genetic algorithm, including the number of parameter value sets or permutations of parameter values (*N)* to consider at each iteration *N* of the algorithm (*N* is the "population size" in the jargon of genetic algorithms), the "recombination" rate and (Q_1), the "mutation" rate (Q_2), and the maximum number of iterations or "generations" ($G_{\max}$); 2) to select the objective function to use to evaluate the goodness of fit; 3) to identify the *K>0* parameters that are free to vary in fitting; 4) to specify initial parameter values or select a procedure for initializing parameter values; 5) to define relationships between simulation output and data input files; 6) to specify the maximum number of CPU cores to be used in parallel; 7) to specify settings related to cluster job scheduling; 8) to specify the number of replicate stochastic simulation runs to perform to obtain smoothed simulation outputs; 9) to define paths to one or more .exp files, a .bngl file, an optional .rnf file, the directory containing BioNetGen and NFsim, and the directory to which output should be sent and a job name; and 10) miscellaneous BioNetFit options, including the level of verbosity of console messages regarding job progress, plotting options, and a seed for the pseudo-random number generator used by BioNetFit (which is distinct from that used by the simulator).

The vast majority of BioNetFit options have been assigned default settings. However, many of these settings are inherently dependent on the fitting problem being solved. The BioNetFit website provides example .conf files, which are extensively annotated. The .conf files for example3 and example4 configure BioNetFit for a job to be executed on a standalone platform. The .conf files for example1 and example2 configure BioNetFit for a job to be performed on a cluster. It is intended that these files provide a helpful starting point for users, who will need to configure BioNetFit for their specific fitting problems. Each BioNetFit configuration option is explained in the table below.

IV.C.1 List of configuration options and a description of each

The table below provides a comprehensive list of BioNetFit configuration (.conf) file options. The options highlighted in italics are required. The options highlighted in bold may be required (see option description). Other options are optional.

Option Name	Type	Example	Description

PATHS			
model	String	model=/Users/username/BioNetFit/examples/example3/example3.bngl	Absolute path to your model file
exp_file	String	exp_file=/ Users/username/BioNetFit/examples/example3/example3_data1.exp	Absolute path to your .exp file. You may specify this option multiple times for multiple .exp files
output_dir	String	output_dir=/Users/username/BioNetFit/output	Absolute path where you want your output to go
bng_command	String	bng_command=/Users/username/BioNetFit/NFsim_v1.11/BNG2.pl	Absolute path to your BioNetGen executable
job_name	String	job_name=example3_Gmax100_N100_objfunc1	This is the job name, and name of the directory that will contain your results.
DISPLAY OPTIONS			
show_welcome_message	Bool	show_welcome_message=1	Whether or not to show a welcome message when running BioNetFit. Default is 1.
ask_create	Bool	ask_create=1	Whether or not to ask when creating a new output directory. Default is 1.
ask_overwrite	Bool	ask_overwrite=1	Whether or not to ask when overwriting existing job output. Default is 1. Warning: Turning this off can be dangerous!
verbosity	Int	verbosity=2	How much information to display about run progress. 0 = no information, 4 = way more information than you are likely to need. 1 or 2 is recommended. Default is 1.
GENERAL OPTIONS			
parallel_count	Int	parallel_count=2	Number of models to run simultaneously when running on a personal computer. It is recommended that this is set to the number of CPU cores present in your machine. Default is 2.
max_walltime	HH:MM:SS	max_walltime=01:00:00	If a permutation is taking longer than maximum walltime, we will move on without it. Note: Many qsub systems will give your jobs lower priority as your walltime increases. Default is 01:00:00 (1 hour).
make_plots	Bool	make_plots=1	Whether or not BioNetFit will output plots when results are ready. Default is 0.
seed	Int	seed=42	Seed for BioNetFit's random number generator. Comment this option out in order to use a different seed every time. Note: This seed only applies to BioNetFit's random number generator. It does not affect NFsim or BioNetGen. To give NFsim a seed, set seed=### in your simulation command or .rnf file. Default is no user-specified seed (i.e.,the seed is determined automatically).
max_retries	Int	max_retries=3	Sometimes a generation must be re-run because there were too many simulations which did not finish successfully. This often happens because they hit the specified maximum walltime. max_retries is the number of times a generation will be run before BioNetFit gives up. Default is 3.
delete_old_files	Bool	delete_old_files=1	BioNetFit results can use a lot of disk space. Enabling this option will delete unnecessary files (.net, .bngl, .xml, .cdat, etc) as a run progresses. We note that if this option is turned on, BioNetFit may not be able to supply you with a model or .net file containing best-fit parameters at the end of the fitting run.

			Default is 1.
FITTING OPTIONS			
max_generations	Int	max_generations=100	The maximum number of iterations or generations (G_{max}). BioNetFit will execute this number of generations unless it quits earlier due to satisfaction of a stopping condition.
permutations	Int	permutations=100	The number of parameter value sets or population size or number of permutations (N). At each iteration, N simulation jobs will be performed.
first_gen_permutations	Int	first_gen_permutations=250	Number of permutations to run in your first generation. This option allows you to start a fitting run with greater diversity. This setting will have the same value as *permutations* by default.
smoothing	Int	smoothing=1	Number of replicate simulation runs. The default value is 1. Multiple simulation runs are useful for smoothing the results of stochastic simulation. NB: Setting this parameter to 10 will increase the cost of simulation by an order of magnitude.
max_objfunc_value	Float	max_ objfunc_value=1000000	If simulation results for a given parameter value set produce a function evaluation greater than the value specified for this parameter, the parameter value set will not be used in "breeding." The default is no limit.
min_ objfunc_value	Float	min_ objfunc_value=10	If simulation results for a given parameter value set produce a function evaluation less than the value specified for this parameter, fitting will stop. The default is no stopping condition (i.e., this parameter is set to 0).
stop_when_stalled	Bool	stop_when_stalled=1	If this parameter is set to 1 (default setting) BioNetFit will stop the fitting run if new parameter value sets are identical to old parameter value sets after "breeding." This situation would only be expected to arise if the rate of "mutation" is set to 0.
objfunc	Int	objfunc=1	y(x) = measured value at condition x y'(x) = simulated value at condition x The objective function is a sum of terms over all x values. The terms can have the forms indicated below: 1: $(y(x)-y'(x))^2$ 2: $((y(x)-y'(x))/y\_SD(x))^2$ 3: $((y(x)-y'(x))/y(x))^2$ 4: $((y(x)-y'(x))/ybar)^2$ The objective function is selected by setting *objfunc* to one of the index values above. A setting of *objfunc=1* indicates that the objective function is the sum-of-squares function (i.e., nonlinear least squares fitting). A setting of *objfunc=2* indicates that the objective function is the chi-square function (i.e., weighted nonlinear least squares fitting). For this option, y values in .exp files must be accompanied by y_SD values. The default setting is 1.
extra_weight	Int	extra_weight=0	When selecting parents to breed, they're already weighted such that runs with better goodness-of-fit are more likely to be chosen. If you would like to weight the selection even more, increase this

			value. Note that more weight will make it less likely that different parents will be selected for breeding. Note: Values must be in the range 0-10. Default value is 0.
swap_rate	Float	swap_rate=0.75	This parameter sets the "recombination" rate (Q_1). It is a probability, so values must be between 0 and 1. The default setting is 0.5.
max_parents	Int	max_parents=90	This parameter sets the maximum number of "parents" to be used in "breeding." By default, this parameter is equal to the population size (N).
force_different_parents	Bool	force_different_parents=1	This parameter determines whether a "parent" is allowed to "breed" with itself. If self-breeding is allowed, some parameter value sets may be unchanged from iteration to iteration. The default setting is 1.
keep_parents	Int	keep_parents=1	This parameter determines the number of top-ranked parents (parameter value sets) that will be carried over to the next iteration. Default is 0.
divide_by_init	Bool	divide_by_init=1	This option instructs BioNetFit, before cost function evaluation, to divide each model output by the first listed value of the output in the .gdat (or .scan) file, i.e., by the top numerical value in the .gdat (or .scan) file. This feature is useful when the first value listed is, for example, the basal steady-state value and the experimental data being used in fitting consists of fold-change measurements made relative to the basal steady state.
log_transform_sim_data	Int	log_transform_sim_data=2	If this option is set to an integer value, all simulation output will be log transformed using the specified value as the transformation base. This transformation takes place prior to cost function evaluation.
standardize_sim_data	Bool	standardize_sim_data=1	This option instructs BioNetFit to standardize simulation output to a mean of 0, using the following formula: y(x) = (y(x) – ybar)/stdev(y) This standardization happens prior to cost function evaluation.
standardize_exp_data	Bool	standardize_exp_data=1	Similar to the above option, but in regards to experimental data (.exp file).
Mutate	Mix	mutate p1__FREE__ 0.2 0.1 mutate p2__FREE__ 0.2 0.1 or mutate default 0.2 0.1	This option is used to identify a free parameter. It has three arguments. The first is a parameter identifier or parameter name. We recommend that parameter identifiers such as *p__FREE__* be used rather than parameter names such as *p*. See Section IV.B.3. The second argument is the "mutation" rate (Q_2). This quantity is a probability and therefore it must take values between 0 and 1. The third argument sets a range in which a factor will fall. This factor multiplies the old value of a parameter to obtain a new "mutated" value of the parameter. The factor is chosen randomly. It is a uniform random deviate U that lies between 1 - arg3 and 1 + arg3, where arg3 is the user-specified value of the third argument. In other words, p_new = p_old*U =

			p_old*(1 + rand(0,2 * arg3) - arg3), where rand is Perl's pseudo random number generator. The first argument may be set to "default" rather than a parameter name. If this is done, only a single mutate option should be included in the .conf file. The purpose of this option is to indicate that the second and third arguments should be used for all free parameters.
INITIAL PARAMETER GENERATION The following options are different methods of generating initial parameters in the first generation. Subsequent generations use parameters from breeding/mutations. ONE OF THESE IS REQUIRED FOR EACH PARAMETER TO BE FIT.			

static_list_var	Mix	static_list_var=p1__FREE__ val_p11 val_p21 val_p31 … val_pN1
		static_list_var=p2__FREE__ val_p12 val_p22 val_p32 … val_pN2
		:
		static_list_var=pK__FREE__ val_p1K val_p2K val_p3K … val_pNK
			This option allows a user to explicitly specify initial values of a parameter for each of the N parameter permutations to be considered. Up to K lines may be used, where K is the number of free parameters. Each line includes the name or identifier of a free parameter followed by N whitespace separated numerical values.

random_var	Mix	random_var=p1__FREE__ min max
		random_var=p2__FREE__ min max
			This option allows a user to indicate that the initial value for a parameter should be randomly generated. The option takes three arguments. This first argument is the name or identifier of a free parameter. The next two arguments specify range. The second argument corresponds to the minimum allowed value, and the third argument corresponds to the maximum allowed value. The initial value for the parameter is drawn from a uniform distribution that spans the (linear) range. This option may be provided for multiple free parameters.

lognormrandom_var	Mix	lognormrandom_var=p__FREE__ mean stdev	This option allows a user to indicate that the initial value for a parameter should be randomly generated. The option takes three arguments. This first argument is the name or identifier of a free parameter. The next two arguments define the mean and standard deviation of a normal distribution. The initial value for the parameter is drawn from this distribution. This option may be provided for multiple free parameters.
loguniform_var	Mix	loguniform_var=p__FREE__ min max	This option allows a user to indicate that the initial value for a parameter should be randomly generated. The option takes three arguments. This first argument is the name or identifier of a free parameter. The next two arguments define the minimum and maximum values of a logarithmic range. The base is 10. This option may be provided for multiple free parameters.
CLUSTER OPTIONS			
use_cluster	Bool	use_cluster=1	This option should be set to 1 if you are running on a cluster. It is set to 0 by default.

cluster_command	String	cluster_command=sbatch
		cluster_command=qsub
			The user may specify the command to be used to submit a job for processing. However, in most cases, this is not necessary. BioNetFit is designed to automatically determine which command to use. Acceptable options are limited to "sbatch" and "qsub."

run_job_from_worknode	Bool	run_job_from_worknode=1	Whether or not your cluster allows you to submit cluster jobs FROM working nodes. BioNetFit is designed to automatically determine which setting to use, so in most cases the user will not need to set this option. Set to 0 if you can only submit cluster jobs from the submission node (common on SGE clusters). Set to 1 if you can submit cluster jobs from work nodes (common on Torque/PBS and SLURM clusters).

cluster_software	String	cluster_software=slurm
		
		or
		
		cluster_software=ge
		
		or
		
		cluster_software=torque
			The user may specify the cluster platform being used. However, in most cases, this is not necessary. BioNetFit is designed to automatically determine which cluster platform is being used. Acceptable options are “slurm,” “torque,” and “ge.”

pe_name	String	pe_name=orte	The user may specify the parallel environment being used on the cluster. This is generally necessary on an SGE-based cluster. In general, this setting must be obtained from the cluster administrator; however, a typical setting is “orte.”
queue_name	String	queue_name=all	The user may specify the work queue being used on the cluster. In general, this setting must be obtained from the cluster administrator; however, a typical setting is “all.”
account_name	String	account_name=my_account	The user may specify the account name being used on the cluster. This setting must be obtained from the cluster administrator. The account name is typically used to meter cluster usage.
job_sleep	Int	job_sleep=1	How many seconds to wait between sending off cluster jobs. This is useful because cluster schedulers sometimes fail to submit jobs properly when they are submitted in close succession. He default setting is 1.
cluster_parallel	Int	cluster_parallel=8	Number of CPU threads to run in parallel. Ideally, you should set this to the number of CPU cores contained on most nodes in your cluster. Thus, if most nodes on the cluster have 16 CPU cores, set this option to 16. The default setting is 8.
multisim	Int	multisim=2	Number of simulations to run with each thread. The simulations will be run in serial. If your simulations finish quickly and the *permutations* option is set to a large value, you may want to set *multisim* to a value greater than 1, which will limit the number of jobs submitted. This will avoid the idle time between job submissions. The default setting is 1.
save_cluster_output	Bool	save_cluster_output=1	This option controls message output from the cluster job scheduler. Its default setting is 0 (no output). Setting this option to 1 may be useful for debugging and/or for reporting bugs or problems.

IV.D Plotting results
BioNetFit can generate plots illustrating quality of fit at the end of a fitting run. This capability has several dependencies. The following libraries must be installed:

1) **libgd:** http://libgd.bitbucket.org/
2) **GD:** http://search.cpan.org/dist/GD/
3) **GD::Text**: http://search.cpan.org/~mverb/GDTextUtil-0.86/Text.pm
4) **GD::Graph:** http://search.cpan.org/~mverb/GDGraph/Graph.pm

Instructions for installing and configuring these libraries are platform-dependent. It is recommended that the user consult the appropriate third-party documentation for instructions on installing the libraries.

VI.E Key considerations for fitting to time-series data
Use the minimum number of .exp files to pass time-series data to BioNetFit. The header of the first column of numerical entries in each .exp file must be "time." Recall that the first row in the .exp file must begin with a hash (#). Each subsequent row must begin with a whitespace. Headers for simulation outputs should correspond to the names of observables or functions defined in the .bngl file. In other words, the .exp file has the same format as a .gdat file. In the .bngl file, include a *simulate* command for each .exp file. The "method" argument of the simulate command determines which simulator is used to generate simulation data. For example, an argument setting of *method=>"ode"* will invoke BioNetGen's ODE solver, which is CVODE in the SUNDIALS package. A parameter setting of *method=>"nf"* will invoke NFsim. Recall that the .gdat file generated by a simulation call should have a suffix that corresponds to the basename of the corresponding .exp file. A suffix is defined by using the "*suffix*" argument of the *simulate* command. Importantly, report times for the .gdat file should include the sample times in the .exp file. We recommend that sample times be listed explicitly if possible, using the "*sample_times*" argument of the *simulate* command. If stochastic simulation is being performed, we recommend that expensive .species file generation be disabled to reduce fit time when running stochastic simulations. This can be done by setting *get_final_state=>0* in your .bngl file simulation command. It may be important when running stochastic simulations to use smoothing, meaning averaging of replicate simulation runs. The number of replicate simulation runs can be specified in the BioNetFit configuration file. See the description of the "smoothing" option. If the simulator is one of the simulation engines available within BioNetGen, simulations will be preceded by network generation and a .net file will be produced. The .net file may be provided by a user to avoid network generation. BioNetFit is designed so that network generation, if required, will only be performed once. NB: Network generation is not possible for all rule-based models. If network generation is impracticable we recommend use of the NFsim simulator, which implements a stochastic simulation algorithm that does not require network generation.

VI.F Key considerations for fitting to steady-state dose-response data
Many of the important considerations to keep in mind when using steady-state dose-response data are the same as those to keep in mind when using time-series data. See the section above. An important difference is that the *parameter_scan* action is used in the .bngl file instead of the *simulate* action. The *parameter_scan* action has the same arguments as the *simulate* action, as well as additional arguments. For example, the *parameter_scan* action allows one to identify a parameter to vary. The parameter should be defined in the *parameters* block of the .bngl file. Typically, one would be interested in varying a copy number parameter or a concentration parameter. In an .exp file containing steady-state dose-response data, the header of the first column of numerical entries must correspond to the name of the parameter being varied. It should be noted that the *parameter_scan* action finds steady-state values in a brute-force manner by simulating for a sufficiently long time. It is the user's responsibility to specify the simulation time and to ensure that it is indeed sufficiently long to approach steady state. A call to the *parameter_scan* action will produce a .scan file

instead of a .gdat file. As in the case of time-series data, it is important to attach a suffix to the simulation output file, which is a .scan file. The suffix is supplied by the user using the suffix argument of *parameter_scan*. The simulation outputs sent to a .scan file should have names that match headers in the corresponding .exp file. Importantly, the parameter values considered in a scan need to include the parameter values listed in the corresponding .exp file. We recommend that the parameter values to consider in a scan be explicitly specified by a user if possible. This can be accomplished by using the "par_scan_vals" option of *parameter_scan*. Note that this argument was introduced with BioNetGen version 2.2.6-stable. It is not available in earlier versions. An example of usage is as follows: *parameter_scan({par_scan_vals[val1,val2,…],…}).* The order in which parameter values are listed is important. The order of the parameter values in the "par_scan_vals" argument should match the order of the parameter values in the corresponding .exp file. If smoothing is used with stochastic simulation, the *parameter_scan* action will be called the number of times indicated in the BioNetFit configuration file. Recall that the "smoothing" option sets the number of replicate simulations.

VI.G Key considerations for using the sum-of-squares objective function (i.e., non-linear least squares fitting)

The sum-of-squares function (F_1) is the default objective function. It is the simplest function to use in that only *x-y* experimental data are required. The function is defined as follows: $F_1 = \sum_i(y_i(x_i)-y_i'(x_i))^2$, where $y_i(x_i)$ is the measurement value at condition x_i and $y_i'(x_i)$ *is the simulated value at condition* x_i. The sum is over all measurement conditions. Importantly, if the sum-of-squares objective function is used, an .exp file should not include a column with a _SD header. Such headers are included only when using the chi-square objective function (see below).

VI.H Key considerations for using the chi-square objective function (i.e., weighted non-linear least squares fitting)

The chi-square function is defined as follows: $F_2 = \sum[_i(y_i(x_i)-y_i'(x_i))/SD_i(x_i)]^2$, where $y_i(x_i)$ is the sample mean measurement value at condition x_i, $SD_i(x_i)$ is the sample standard deviation, and $y_i'(x_i)$ is the simulated value at condition x_i. The sum is over all measurement conditions. Importantly, if the chi-square objective function is used, every measurement column must be accompanied by an _SD column (see section IV.A.1), which lists sample standard deviations. If sample standard deviations are not available for a series of measurements, an _SD column should still be provided but this column should list all 1's. We recommend using the chi-square function when possible, because with this function, noisy measurements are deemphasized in fitting, i.e., given less weight. This can speed fitting and make it easier to find a good fit.

VI.I Key considerations for using BioNetFit on a cluster

BioNetFit supports popular implementations of SLURM, SGE (SunGridEngine), and Torque/PBS. Some implementations of these resource management tools may be incompatible with BioNetFit. If you run into a compatibility problem, please notify the BioNetFit developers. We may be able to support your cluster environment. No promises.

To start a fitting run on a cluster, the *use_cluster* argument in the BioNetFit configuration (.conf) file must be set to 1. BioNetFit is capable of determining which cluster platform is installed on the cluster. If BioNetFit is not able to determine the cluster software automatically, the user will be prompted for the cluster platform at runtime. There are several options that the user may wish to change to optimize fitting on a cluster. These options are documented in section IV.C.1.

If you are using SSH to connect to a cluster, you may disconnect from the cluster at any time after the start of a fitting run. Disconnecting will not affect the progress of the fitting run, but it will terminate console messages. Job progress may be checked at a later time in one of two ways:

1) After reconnecting to the cluster using SSH, you may execute the following command:
   ```
   perl BioNetFit.pl monitor /path/to/config/file.conf
   ```

   Note that */path/to/config/file.conf* in the above command should be replaced with the correct path to the configuration file used to start the fitting run. If you are running multiple jobs at the same time, there should be multiple .conf files present in your home directory or subdirectory, one for each fitting run. Therefore, after starting a fitting run, do not edit the .conf file that was used to launch the fitting run until the fitting run is over.

2) After reconnecting to the cluster using SSH, execute one of the following commands:
   `qstat –u $USER` (in the case of Torque/PBS or SGE cluster software)
   or
   `squeue –u $USER` (in the case of SLURM cluster software)

Each of the commands given above will produce a display of your currently-running jobs. The cluster jobs launched by BioNetFit will have names (which will appear in the display of currently running jobs) that follow this convention:

[*job_name truncated to four characters]*-Xg(-Xg+1)(_Cc)

The job_name is specified in the .conf file by the user. If the job name is longer than four characters, it is truncated. The reason for truncation is that the number of characters comprising a job name in the display is limited. 'X' is one of the following four characters: A, G, S, or C. The character indicates the current execution stage of a fitting routing: ‘A’ indicates the “analyze” stage (objective function evaluation), ‘G’ indicates the “generation” stage (preparation for and launching of simulations), ‘S’ indicates the “simulation” stage (execution of simulations), and ‘C*c*’ references “chunk” run *c,* where *c* is the index of a simulation job that is part of a batch of jobs submitted for processing. The value of ‘g’ is the index of the current generation or iteration of the genetic algorithm. Parentheses indicate optional parts of a name. Examples of names and explanations of the names are provided below.

exam-G1: 1st generation of “example1” run. This job spawns and monitors simulations for the 1st generation.
exam-S1_C1: 1st chunk of “example1” simulations running in the 1st generation.
exam-S1_C2: 2st chunk of “example1” simulations running in the 1st generation.
exam-A1_G2: Analyze 1st generation results and spawn/monitor generation 2nd generation simulations.
exam-A50: Analyze 50th (and last) generation results and generate results.

V. Calling BioNetFit from the command line

To start a fitting run, you may issue the following command:

```
perl BioNetFit.pl path/to/config/file.conf
```

The BioNetFit configuration (.conf) file may be specified as an absolute path (e.g., /home/user/config_files/config_file.conf) or as a relative path (e.g., ./config_files/config_file.conf).

If a fitting run fails as the result of cluster downtime or other system failure, it’s possible to resume the fitting run. Fitting will resume from the most recent generation (iteration). To resume a fitting run, a user may issue the following command:

```
perl BioNetFit.pl resume [Gmax] path/to/config/file.conf
```

This command will cause BioNetFit to delete any simulation output from the most recent generation (iteration) and to then re-run that generation's simulations. [G_{max}] is an optional integer value that sets the *max_generations* option in the .conf file to the number specified by the argument. This argument can be used to resume a fitting run after it has reached the maximum number of generations specified in the original .conf file.

BioNetFit is capable of reporting partial progress. A progress report can be requested by issuing the following command:

```
perl BioNetFit.pl results path/to/config/file.conf
```

VI. Troubleshooting

Error messages and messages regarding job progress are displayed to the terminal in which BioNetFit is running. Additionally, simulator progress and error messages are saved to files with the filename extension ".BNG_OUT." These files can be found in the job output directory. On a cluster, BioNetFit job-progress information is piped to a file in the BioNetFit directory. This file is given the filename *job_name,* which is specified in the BioNetFit configuration file.

If a fitting run fails, it may be helpful to inspect the terminal output, .BNG_OUT files, and if running on a cluster, the *BioNetFit/job_name* file.