Chapter #

Using BioNetFit to estimate parameters of rule-based models

Authors

Summary

Rule-based models consist of a collection of formalized rules that can provide concise representations of large biomolecular interaction networks. These models can be analyzed to obtain better understanding of how system behaviors emerge from biomolecular interactions. Several simulation packages have been developed to analyze rule-based models. In some scenarios, we have experimental results for a given set of known reactions and the goal is to specify the reactions rates that replicates the experimental results in a simulation environment. Since the search space (possible combination of free parameter values) and consequently the search time grows exponentially and the number of unknown parameters, this task is performed manually can be extremely time consuming and practically infeasible. BioNetFit is a fitting tool compatible with two simulation packages for rule-based models that automates the parameter tuning task by utilizing genetic algorithm and parallel processing technologies ***(1)***. This chapter includes step-by-step instructions for software installation and running example fitting jobs. We present an extensive documentation on these file formats, explaining how to preprocess the input files for running BioNetFit. After preparing the input files, this guide provides detailed instructions on how to run BioNetFit and how to access the simulation and fitting results.

BioNetFit can be used to optimize a set of simulation parameters for molecular interaction networks, it is a general fitting tool and can be integrated (after some minor modifications) with other simulator used to model interactions in other discipline such as engineering, economics, business and social networking.

Key words: Model fitting tool, Automated model checking, BioNetFit, NFsim, BioNetGen, rule-based models, molecular interactions, regulatory networks.

1. Introduction

Molecular interactions usually are stated in terms of a set of rules that specify the input molecules and species, the set of changes and reactions, the rate of reaction, and the output molecular entities. Rule-based models consist of a collection of formalized rules that can provide concise representations of large intermolecular interaction networks. These abstract models can be analyzed for obtaining better understanding on how intermolecular interactions form complex biological systems.

A popular standard language that is recently developed in order to represent molecular reactions is called BNGL (BioNetGen language), which specifies a complex molecular process through a set of user-friendly instructions. A typical BNGL file includes input molecules and species[1], initial concentrations,

[1] Species in this context is a molecule or molecular complex with specific states for its components. For instance, it can be a protein complex whose specific binding site is phosphorylated.

reactions, reaction rates, observables and simulation parameters (start time, number of steps, stop time, …). Figure 1 shows an example of BNGL file for binding of bivalent ligand to bivalent receptor, available in the user manual of BioNetGen (http://bionetgen.org/index.php/BNGManual:Model_3).
Multiple software packages including BioNetGen ***(2, 3)***, and NFsim ***(4)*** are developed in order to simulates bngl files and provide the outputs.
BioNetGen is a software designed for the specification and simulation of rule-based models of biochemical systems, including signal transduction, metabolic, and genetic regulatory networks ***(2, 3)***. The BioNetGen language has recently been extended to include explicit representation of compartments. A recent review of methods for rule-based modeling is available in ***(5)***. NFsim is a free, open-source, biochemical reaction simulator designed to handle systems that have a large or even infinite number of possible molecular interactions or states ***(4)***. NFsim also has advanced and flexible options for simulating coarse-grained representations of complex nonlinear reaction mechanisms.
These tools can be used to simulate BNGL files and predict the outcome of experiments with application in many fields including chemistry, biology, and oncology. In some applications, the objective is different, where experimental results are already available and the goal is to fine tune the set of reaction rates under which the predefined set of complex reactions specified by a model yield the same results. In other words, we try to discover the reactions by observing the experiment results. One may manually and repeatedly change the reaction rates (or other free parameters) such that the simulations outcome matches the experimental results. This process can be a tedious and extremely time consuming job. BeNeFit is an automated fitting tool that serves for this purpose and accelerates the process of finding optimal model parameters by utilizing several elegant methods and technologies ***(1)***. Firstly, this software uses genetic algorithm for as a heuristic search method in order to find the optimal set of parameters which is much faster than bind exhaustive search methods. Secondly, this software utilize parallel processing capabilities of high performance clustering and drastically reduces the search time by running simulations with different sets of parameters in parallel. A block-diagram of the model tuning process using BioNetFit is depicted in Figure 2.

BioNetFit can be used on a standalone platform (Mac, Windows/Cygwin, and Linux) or a Linux-based cluster running SLURM or, Torque/PBS, or standalone MPI, SLURM and Torque/PBS must be configured to use MPI. Information on these resource management tools can be found at the following websites:

1. SLURM Workload Manager: http://slurm.schedmd.com/
2. Torque Resource Manager: http://www.adaptivecomputing.com/products/open-source/torque/
3. Massage Passing Interface library: https://www.open-mpi.org/

This chapter 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. For 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.

```
begin model

begin parameters
NA  6.02e23    # Avogadro's number (molecues/mol)
f   0.001      # Fraction of the cell to simulate
Vo  f*1e-9     # Extracellular volume=1/cell_density (L)
V   f*3e-12    # Cytoplasmic volume (L)
L0  1e-9*NA*Vo # Conc. in Molar -> copies per cell
R0  f*3e5
kp1 3.3e6/(NA*Vo)
km1 0.1
kp2 1e6/(NA*V)
km2 0.1
kp3 30
km3 0.1
end parameters

begin molecule types
R(r,r)
L(l,l)
end molecule types

begin seed species
R(r,r) R0
L(l,l) L0
end seed species

begin observables
Species   FreeL     L(l,l)
Molecules Dimers    R==2
Molecules Trimers   R==3
Molecules FOURmers  R==4
Molecules FIVEmers  R==5
Molecules SIXmers   R==6
Molecules SEVENmers R==7
Molecules EIGHTmers R==8
Molecules NINEmers  R==9
Molecules TENmers   R==10
Molecules gt10mers  R>10
end observables

begin reaction rules
# Ligand addition
R1: R(r) + L(l,l) <-> R(r!1).L(l!1,l) kp1,km1
# Chain elongation
R2: R(r) + L(l,l!+) <-> R(r!1).L(l!1,l!+) kp2,km2
# Ring closure
R3: R(r).L(l) <-> R(r!1).L(l!1) kp3,km3
end reaction rules

end model

# Simulation of a truncated network
 generate_network({overwrite=>1,max_stoich=>{R=>10,L=>10}})
 simulate_ode({t_end=>50,n_steps=>20})

# Simulation on-the-fly
 generate_network({overwrite=>1,max_iter=>1})
 simulate_ssa({t_end=>50,n_steps=>20})
```

Figure 1. An example of BNGL file for binding of bivalent ligand to bivalent receptor, available in the user manual of BioNetGen (http://bionetgen.org/index.php/BNGManual:Model_3).

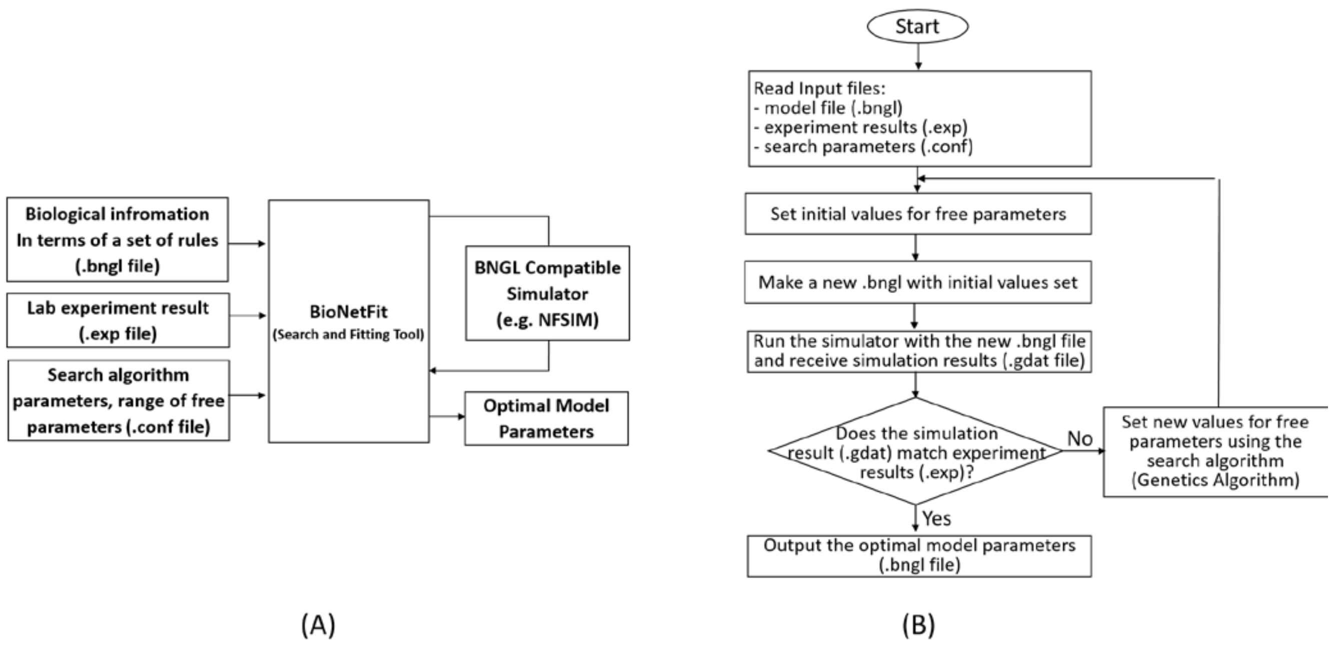

Biological infromation In terms of a set of rules (.bngl file)
Lab experiment result (.exp file)
Search algorithm parameters, range of free parameters (.conf file)
BioNetFit (Search and Fitting Tool)
BNGL Compatible Simulator (e.g. NFSIM)
Optimal Model Parameters
(A)
Start
Read Input files: - model file (.bngl) - experiment results (.exp) - search parameters (.conf)
Set initial values for free parameters
Make a new .bngl with initial values set
Run the simulator with the new .bngl file and receive simulation results (.gdat file)
Does the simulation result (.gdat) match experiment results (.exp)?
No
Set new values for free parameters using the search algorithm (Genetics Algorithm)
Yes
Output the optimal model parameters (.bngl file)
(B)

Figure 2. (a) Block diagram of model fitting using BioNetFit package. (b) BioNetFit flow-chart.

2. Methods

There are three main input files required for running BioNetFit: a configuration file (.conf), a model-specification file (.bngl), and a data file (.exp). This guide includes an extensive documentation on these file formats and their role in the whole process of parameter tuning. After preparing the input files, this guide provides detailed instructions on how to run BioNetFit and how to access the simulation and fitting results.

2.1. Installation instructions

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 or Torque/PBS. SLURM and Torque/PBS must be configured to use MPI. BioNetFit can also run on clusters running only MPI. BioNetFit relies heavily on the C++ Boost libraries and requires that the libraries be compiled and installed prior to compilation. The installation of BioNetFit can be performed trough the following steps:

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. BioNetFit is distributed with NFsim v1.11 and BioNetGen v2.2.6 and is automatically configured to use these simulation tools.

3) Open a terminal window (e.g., on a Mac launch the Terminal application, which is located in the Utilities folder within the Applications folder). 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
```

4) 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.

5) The fitting run will begin, and terminal output reporting job progress will appear in the terminal window. 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.

6) These files are all plain-text files, which can be opened and viewed in a text editor or a spreadsheet program, for example. 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.

For additional details on the installation of BioNetFit, refer to BioNetFit user manual (http://bionetfit.nau.edu/support.html).

2.2. Preparing the model-specification file

BioNetFit is compatible with BNGL, a language for specifying rule-based models ***(6)***. There are several BNGL-compatible simulators available, including BioNetGen ***(3, 6)*** and NFsim ***(4)***. BioNetFit is designed to work with these simulators. Several reviews are available about rule-based modeling and the methods and software tools that enable rule-based modeling ***(5, 7, 8, 9, 10, 11)***.
The model-specification file consists of five blocks, each beginning with a line containing a *begin blockname* command and ending with a line containing an *end blockname* command. The main block names are parameters, molecule types, seed species, reaction rules and observables:

1. Parameters are used to provide numerical values for variables that may appear in the species and reaction rules blocks. Parameters are generally used to specify initial concentrations of species and rate constants for reaction rules:

```
begin parameters
    km1  0.2
    kp1  4e-3
    km2  0.3
    kp2  4e-2
end parameters
```

The number in the first column is an index, the second and third columns are the name and the value of the given parameters. We note that an equal sign can be included between the index and the values. For

instance the firs line can be represented as km1=0.2. As clear from the examples, the number can be in exponential format xey, which is equivalent to $x \times 10^{y}$.
Also note that a bngl file used by BioNetFit should include at least one free parameter. In this case, the numerical value is not set for the free parameter. The following is an example

```
      KD1 KD1__FREE__
```

In this case, the parameter KD1 can take different values based on a range defined for KD1__FREE__ in the cfg file. The name format for free parameters is arbitrary but we encourage the convention of including __FREE__ in the name.

2. Molecules are structured objects with a set of components that can have states. Molecules can bind to each other through their binding sites and generate more complex structures. Each molecule type declaration begins with the name of the molecule followed by a list of components in parentheses. The tilde character ('~') precedes the allowed state values. Here is a typical example that illustrates the syntax:

```
begin molecule types
    A(c)
    B(d,e,F~U~P)
end molecule types
```

This molecule type block specifies that molecule A consists of a single component (c), whereas molecule B has three allowed states (d, e,and F), but only component F is allowed to have two different states (U and P).

3. Seed species are the molecules and complexes that are present at the start of network generation. Each declaration of a seed species is followed by a number or parameter that gives its initial concentration (i.e. A0 and B0). In addition, when a molecule is listed in the species block its state must be defined (F~U).

```
begin species
    A(c)          A0
    B(d,e,F~U) B0
end species
```

4. The observables block is used to define a set of molecules and species whose concentration variations are measured and reported during the curse of simulations. The simulators sums over the concentrations of species sharing similar attributes as defined an observable object, which correspond to the quantities that are measured in typical biological experiments. The first column of an observable declaration indicates the type of observable: Molecules or Species. Molecules indicates a weighted sum over the species selected by the pattern(s) defining a group, with the weight given by the number of matches found for each species. For example, a dimer containing two phosphorylated receptors R would have a weight of 2 in the observable R_phos. On the other hand, Species would count the dimer once. The second column specifies the name of the observable, while the remaining entries (separated by spaces) are patterns (see below) that select species contributing to the observable:

```
begin observables
    Molecules B_phos B(F~P!?)
    Molecules A_B      A(c!1).B(F~P!1)
end observables
```

Patterns are similar to Species, except for the fact that they do not have to be fully specified. Wildcarding is allowed. For instance, the edge label '?', has the special meaning that a bond may or may not be present.

Components that form a bond will have the same binding index (bond 1 is formed between component c of molecule A and component F at state P of molecule B as depicted below). Figure 3 shows a schematic representation of these molecules and their components.

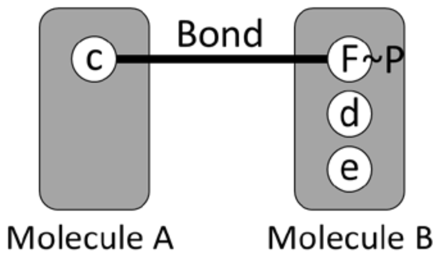


Figure 3. Schematic representation of two molecules binding through specific components.

5. Reaction rules are the generators of species and reactions. Each rule has four basic elements: reactant patterns, an arrow, product patterns, and a rate law. Reactant patterns are used to select a set of reactant species to which the rule will be applied. The arrow indicates whether the rule is applied in the forward direction only or in the forward and reverse directions. The product patterns define how the selected species are transformed by the rule and also act as the species selectors when the rule is applied in reverse (if applicable). Rules may add or delete molecules and edges and may change component state labels. Components may not be added or deleted:

```
begin reaction rules
    A(c) + B(d,e) <-> A(c!1).R(d!1,e) kp1, km1
end reaction rules
```

The reactant pattern A(c) selects ligand molecules that have an unbound c component. When multiple reactant or product patterns appear in a rule, they may be separated by the plus character ('+'), which is used to specify molecularity. For more detailed description of BioNetGen language properties refer to the BioNetGen user manual (http://bionetgen.org/).

2.3. Preparing Data Input Files

Data input files (.exp) include the experimental results (propensities during the curse of an experiment) for a set of molecules and species and has the filename extension .exp. These files have the same format as BioNetGen's .gdat files ***(6)***. 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.

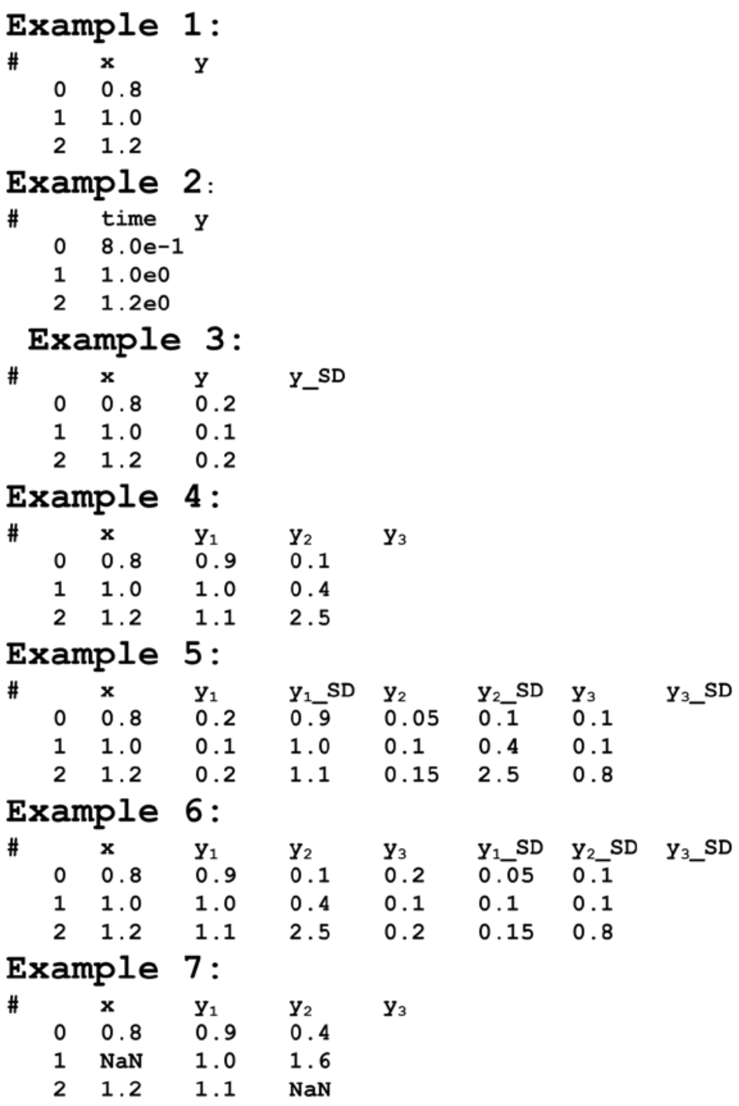
The data input files must be generated following a set of formatting instructions:

1. 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.

2. The first row of a column should be a label or header. The header row must begin with a hash (#).
3. After the first row, additional rows should contain numerical entries. Each such row must begin with a whitespace.
4. 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.
5. 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.
6. 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 in Figure 4 to illustrate the possible formats.
7. 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.

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.



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
```

Figure 4. Examples of .exp files.

2.4. Preparing the BioNetFit Configuration File

The BioNetFit configuration (.conf) file is a plain-text file that allows a user:

1. to specify variation range for the free parameters defined in bngl file.

2. 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" ($G_{\max}$);

3. to select the objective function to use to evaluate the goodness of fit;

4. to identify the *K>0* parameters that are free to vary in fitting;

5. to specify initial parameter values or select a procedure for initializing parameter values;
6. to define relationships between simulation output and data input files;
7. to specify the maximum number of CPU cores to be used in parallel;
8. to specify settings related to cluster job scheduling;
9. to specify the number of replicate stochastic simulation runs to perform to obtain smoothed simulation outputs;
10. 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
11. 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. For detailed examples of configuration options refer to the BioNetFit user manual (http://bionetfit.nau.edu/support.html).

2.5. Consistency among the input files

Before running BioNetFit it is necessary to check the consistency between the input files:

1. The user-supplied .exp files and .bngl file must be consistent with each other. A simulation call must be included in the .bngl file for each .exp file.

2. A simulation call must be included in the .bngl file for each .exp file. 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({prefix=>”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,…})
```

3. 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).

4. 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. 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.

5. 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.

2.6. Running BioNetFit

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

```
~$ perl bin/BioNetFit.pl path/to/config/file.conf
```

For example, the following command performs the fitting run for example 5:

```
~$ perl bin/BioNetFit.pl  examples/example5.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 bin/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. [Gmax] 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 bin/BioNetFit.pl results path/to/config/file.conf
```

References

1. Thomas BR, Chylek LA, Colvin J et al. (2016) BioNetFit: a fitting tool compatible with BioNetGen, NFsim and distributed computing environments. Bioinformatics 32:798-800
2. Blinov ML, Faeder JR, Goldstein B et al. (2004) BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. Bioinformatics 20:3289-3291
3. Harris LA, Hogg JS, Tapia JJ et al. (2016) BioNetGen 2.2: advances in rule-based modeling. Bioinformatics 32:3366-3368
4. Sneddon MW, Faeder JR, Emonet T (2011) Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. Nat Methods 8:177-183
5. Chylek LA, Harris LA, Tung CS et al. (2014) Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. Wiley Interdiscip Rev Syst Biol Med 6:13-36
6. Faeder JR, Blinov ML, Hlavacek WS (2009) Rule-based modeling of biochemical systems with BioNetGen. Methods Mol Biol 500:113-167
7. Chylek LA, Harris LA, Faeder JR et al. (2015) Modeling for (physical) biologists: an introduction to the rule-based approach. Phys Biol 12:045007
8. Sekar JA, Faeder JR (2012) Rule-based modeling of signal transduction: a primer. Methods Mol Biol 880:139-218
9. Maus C, Rybacki S, Uhrmacher AM (2011) Rule-based multi-level modeling of cell biological systems. BMC Syst Biol 5:166
10. Lemons NW, Hu B, Hlavacek WS (2011) Hierarchical graphs for rule-based modeling of biochemical systems. BMC Bioinformatics 12:45
11. Hwang M, Garbey M, Berceli SA et al. (2009) Rule-Based Simulation of Multi-Cellular Biological Systems-A Review of Modeling Techniques. Cell Mol Bioeng 2:285-294