Automatic Network Generation, Parameterization and Model Selection

## Motivation

Often, a modeler in biology must infer a network structure from noisy and incomplete data in the literature. Typically the modelling approach to this problem is to build separate models for each of your hypotheses, fit the data to each model and then calculate some statistics to discriminate between them. This is a time consuming and error prone task that is only feasible with a small number of hypotheses. This workflow was designed to remove the laborious task of manually building each model and opens the possibility of testing many more hypotheses for a given problem.

## Automatic Network Generation

The automatic generation of networks utilizes Prof. Darren Wilkinson’s SBML shorthand which is a human readable text file representation of SBML models that can be converted to SBML ([Gillespie *et al.*, 2006](#_ENREF_1)). The text files are easily manipulated using high level programming languages such as Python. As such, ‘SBconcat\_combo.py’ was written to concatenate a backbone model with an unspecified number of alternate hypotheses in all possible combinations. All information and files needed to use SBconcat\_combo are in the folder, ‘SBconcat\_combo’ along with a demonstration model that reconstructs a model of TGF-β receptor internalization dynamics ([Vilar *et al.*, 2006](#_ENREF_3)).

## Parameterization

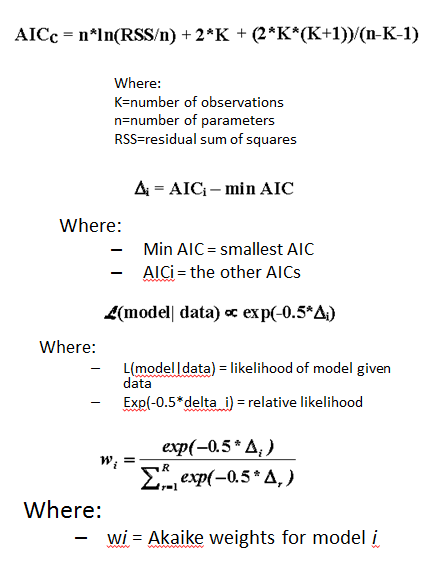
It is difficult to find a tool that is capable of easily automating multiple optimizations. Software with a graphical user interface, such as COPAS requires each variable be mapped independently within each model which is a time consuming and error prone process. Of course, COPASI does have an API for many languages but the Python interface still needs much work and is therefore not used. An alternative and recently released piece of software is a Matlab Toolbox called IQMTools. This is built on SBPOP/SBToolbox2 package ([Schmidt](#_ENREF_2)) and is better suited to automating the parameter estimation task because 1) it has the ability to use either a graphical user interface as well as Matlab scripts and 2) because building an ‘IQM project’ implicitly deals with managing experimental data so you only have to map data to variables once for all models (provided you get it right first time).

An IQM project consists of a ‘models’ and an ‘experiments’ folder. SBML representation of biological models can be placed in the models folder while a separate folder for each experiment is produced for experimental data. The format for an experiment includes a data file and an experimental description file which together describe an experiment. For more information the reader is referred to the IQM Tools documentation. The general idea to parameterize a set of models is to build an IQM project containing all models and all experiments. Subsequently we process the project with the script ‘prep\_IQM.py’ which takes a project containing all models and all data to a folder of projects (called Parent\_IQM\_Project in the example), each containing a single model and all data. Instructions on how to use ‘prep\_IQM.py’ are within the script itself but are also visible from the command line using ‘python prep\_IQM.py –h’.

The output of ‘prep\_IQM.py’ is a folder containing a list of IQM projects, called Child\_IQM\_Project in the examples. The next step is to copy and paste the two Matlab scripts ‘run1estimation.m’ and ‘run\_all\_estimations.m’ into the Child\_IQM\_Project folder and use the ‘run\_all\_estimations.m’ script. This performs a global followed by local optimization then calculates the model selection criteria which is written to a file called ‘Model\_Selection\_Criteria.csv’.

## Model Selection Criteria

The model selection criteria used in this project is the corrected version of the Akaike Information Criteria (AIC) which is related to the sum of squares objective function used in optimization. The data can be processed a little further to calculate what are known as Akaike weights *wi* which quantify the feasibility for each model in the set of models tested being the ‘true’ model (Figure 1).

H:\Newcastle University\PhD\Presentations\AIC_weights.tiffH:\Newcastle University\PhD\Presentations\AIC.tiff

C

B

A

Figure 1: A) AIC and Akaike weights calculation. B) Comparison of AIC for each model in example (lower is better). C) Akaike weights for each model

## A Note to the Reader

As of this date (10-01-2016) there are a few bugs in the parameterization section of the automated workflow that prevent it being generalizable to any model. For this reason the Matlab scripts to run the parameterizations are not yet included in the example files. Instead I have included a pre-run example of the program that works for the test case only. This folder is called ‘Child\_IM\_Project\_worked\_example’ and the results are in the Model\_Selection\_Table.xlsx file. The bugs will be fixed ASAP.

## References

Gillespie, C.S., Wilkinson, D.J., Proctor, C.J., Shanley, D.P., Boys, R.J. and Kirkwood, T.B.L. (2006) 'Tools for the SBML Community', *Bioinformatics*, 22(5), pp. 628-629.

Schmidt, H. 'The “SBPOP Package”: Efficient Support for Model Based Drug Development–From Mechanistic Models to Complex Trial Simulation Henning Schmidt Novartis Pharma AG'.

Vilar, J.M.G., Jansen, R. and Sander, C. (2006) 'Signal Processing in the TGF-β Superfamily Ligand-Receptor Network', *PLoS Comput Biol*, 2(1), p. e3.