BMC Systems Biology

SOFTWARE

Open Access

SBpipe: a collection of pipelines for automating repetitive simulation and analysis tasks

Piero Dalle Pezze* and Nicolas Le Novère

Abstract

Background: The rapid growth of the number of mathematical models in Systems Biology fostered the development of many tools to simulate and analyse them. The reliability and precision of these tasks often depend on multiple repetitions and they can be optimised if executed as pipelines. In addition, new formal analyses can be performed on these repeat sequences, revealing important insights about the accuracy of model predictions.

Results: Here we introduce SBpipe, an open source software tool for automating repetitive tasks in model building and simulation. Using basic YAML configuration files, SBpipe builds a sequence of repeated model simulations or parameter estimations, performs analyses from this generated sequence, and finally generates a LaTeX/PDF report. The parameter estimation pipeline offers analyses of parameter profile likelihood and parameter correlation using samples from the computed estimates. Specific pipelines for scanning of one or two model parameters at the same time are also provided. Pipelines can run on multicore computers, Sun Grid Engine (SGE), or Load Sharing Facility (LSF) clusters, speeding up the processes of model building and simulation. SBpipe can execute models implemented in COPASI, Python or coded in any other programming language using Python as a wrapper module. Future support for other software simulators can be dynamically added without affecting the current implementation.

Conclusions: SBpipe allows users to automatically repeat the tasks of model simulation and parameter estimation, and extract robustness information from these repeat sequences in a solid and consistent manner, facilitating model development and analysis. The source code and documentation of this project are freely available at the web site: https://pdp10.github.io/sbpipe/.

Keywords: Pipeline, Modelling, Simulation, Parameter estimation

Background

The range of software tools developed by the Systems Biology community has grown considerably in the last few years, in particular aimed at supporting mathematical modelling of biological networks. The development of a mathematical model typically comprises successive phases: design, parameterisation, simulation and testing. Model design is the phase where the core of the problem to investigate is summarised using a mathematical formalism. Once designed, the model parameters need to be calibrated, for example using some experimental data. After this stage, the model is used for generating predictions which are then tested experimentally. Depending on the outcome, a model can be refined in order to improve or correct its prediction.

Many tools already exist to generate, simulate and analyse mathematical models [1, 2]. Although these tools provide modellers with key functionalities for model parameter estimation and simulation, it has become clear that the accuracy of these tasks depends on multiple repetitions. Furthermore, the analysis of this batch of repeats can reveal important insights regarding the model itself and the data used for calibration. Therefore, it is useful to repeat tasks such as parameter estimation or stochastic simulation, collect statistics and visualise these results.

SBpipe is an open source software tool which provides modellers with a collection of pipelines for model development and simulation. A pipeline for parameter estimation allows users to repeat a model calibration

*Correspondence: piero.dallepezze@babraham.ac.uk
The Babraham Institute, Babraham Campus, Cambridge CB22 3AT, UK

many times on a multicore machine or a computer cluster. The generated fit sequence is then analysed, and information about the profile likelihood from parameter estimation samples is represented graphically and textually. Support for model simulation is also provided with pipelines for time course model simulation, as well as single and double parameter scans.

Implementation

SBpipe is an open source software package developed with the Python [3] and R [4] programming languages. Python is the main programming language connecting all the package components, whereas R is used for generating statistics and plots. The use of this statistics-dedicated programming language for analysing the results allows users to run the provided R scripts independently of SBpipe using an R environment. This can be convenient if further data analyses are needed or plots need to be annotated or edited.

Pipelines in SBpipe are configured using YAML configuration files. This allows modellers to easily edit their tasks manually or programmatically if needed. Examples of configuration files can be found within the main package in the folder

```
tests/insulin_receptor/
```

In order to maintain a flexible and extendible design, SBpipe abstracts the concepts of simulator and pipeline. The class `Simul` is a generic simulator interface used by the pipelines in SBpipe. This mechanism uncouples pipelines from simulators which can therefore be configured in each pipeline configuration file. Currently, the available simulators are `Copasi` and `Python`. These simulators process models developed in COPASI [5] and models coded in Python, respectively.

SBpipe passes the report file name as an input argument to the latter. The Python program is then responsible for generating a report file containing the simulation (or parameter estimation) results. Python can also be used as a wrapper module for running models coded in any programming language. Rather than coding a model itself, the Python file can call an external program containing the model. This Python wrapper must forward the report file name to this external program which becomes responsible of generating the report file. With this simple approach, users can run their existing models using customised command options or any program library they need. The `tests/` folder contains examples of models coded in R, Octave, or Java programming languages, and executed using basic Python module wrappers. The supplied R models depend on the packages minpack.lm, deSolve, and sde, whereas the supplied Java model requires a JVM. Dependencies for these additional models must be installed separately.

The class `Pipeline` represents a generic pipeline, which is extended by each SBpipe pipeline. The following pipelines are currently available:

- `simulate`: deterministic or stochastic time course stimulation;
- `single_param_scan`: scan a model parameter;
- `double_param_scan`: scan two model parameters;
- `param_estim`: model parameter estimation including sampling of the parameter likelihood.

An SBpipe pipeline performs three tasks: data generation, data analysis, and report generation. The first task loads and runs a simulator at runtime and organises the generated data. The second task computes statistics and plots from these data. Finally, the third task generates a LaTeX/PDF report containing the computed plots. Because of the interdependency between these tasks, their execution is sequential. However, users can select the tasks to run in the pipeline configuration file. A typical scenario requiring a task to be turned off would be the analysis of data previously generated data using different configuration thresholds. In this case, the data generation task can be disabled to prevent SBpipe from re-running the simulations.

Pipelines for parameter estimation or stochastic model simulation can be computationally intensive. SBpipe allows users to generate repeats of model simulation or parameter estimation in parallel. In a configuration file, users can select the number of repeats, and whether the jobs should be executed locally using Python multiprocessing or in a computer cluster. In this case, SBpipe supports the cluster types Sun Grid Engine (SGE) and Load Sharing Facility (LSF).

The project is available on the GitHub repository. Numerous test cases are also provided within the package. Every time the source code is updated online, these tests are automatically executed by Travis.CI, a GitHub application for continuous integration service. For standard users, these tests are useful examples of how to configure SBpipe. User and developer documentations for this project are available online and within the project folder.

Results

To demonstrate SBPIPE functions we will use a minimal model of insulin receptor (IR). This IR model is a module of a more complex Insulin/TOR model [6] (Biomodels database [7] id: BIOMD0000000581). This choice enables users to quickly reproduce the results shown in this article using the SBpipe test suite and to present the results in the most compact manner. This model describes the activation of the insulin receptor upon insulin stimulation. In the presence of insulin, the insulin receptor beta ($IR\beta$) is phosphorylated on Y1164. The phosphorylated receptor is then dephosphorylated and enters in a refractory state. This latter state is used to introduce a delay in the system succintly representing receptor internalisation, degradation and synthesis, thus reducing the number of model parameters. Finally, from this refractory state the receptor can become functional again. Details of the model are provided in Additional file 1: Table S1, Figure S1. The generic pipeline work flow is shown for the parameter estimation pipeline in Fig. 1a. To illustrate how SBpipe can reveal parameter identifiability issues from multiple parameter estimations, two fit sequences are independently generated using sufficient and insufficient data sets (Additional file 1: Tables S2–S4). For each group, SBpipe generates $N = 1000$ independent parameter estimations using Particle Swarm optimisation algorithm [8] as implemented in COPASI. These calibrations are then processed in the data analysis task. Although SBpipe does not contain a pipeline for computing identifiability analysis directly, the parameter estimation pipeline can help identify issues in parameter estimation by projecting the estimates for each parameter. This analysis uses not only the best fit of each of the N estimations, but also the sub-optimal fits. As these fits represent samples of the parameter space, they can reveal a *sampled profile likelihood estimation (PLE)* for each estimated parameter. For direct methods calculating model parameter profile likelihoods using COPASI, see [9] or https://pypi.python.org/pypi/PyCoTools.

Results of estimation tasks using data sets presented in Table S2A and Table S2B are shown in the *Identifiable* or *Non-identifiable* columns of Fig. 1, respectively. The *Identifiable* column shows how the parameter $k1$ presents clear confidence intervals at 66%, 95%, and 99% percents of confidence levels (CL). The *Non-identifiable* column shows how the same parameter is practically non-identifiable to the right of the confidence interval. Parameter distributions and correlations are also computed for the best fits, and for the fits with objective values lesser than a confidence level of 95%. For the complete results generated by this pipeline, see Additional file 1: Tables S2–S4, Figures S2–S8.

Results generated by the time-course simulation pipeline are shown in Fig. 1b. Deterministic and stochastic model simulations are illustrated for the phosphorylated state of the IR species. For deterministic simulation, time courses of model variables are simply plotted. For stochastic simulations, SBpipe can represent time courses with mean (black line), the 95% confidence intervals of the mean (cyan bars), and one standard deviation (blue bars). The second panel in Fig. 1b show this plot using a sequence of 40 independent stochastic simulations. If available, data corresponding to model variables can easily be added to the plot by specifying the data set file name in the configuration file. For the complete results, see Additional file 1: Figures S9–S10.

Figure 1c shows the results from the single parameter scan pipeline. Simulations are ran with values of the parameter `k1` within the 95% confidence interval as determined by the parameter estimation using the data with a sufficient number of data points. If needed, differential scales can also be configured in order to discriminate protein levels. This is particularly useful if a simulated protein knockdown (or overexpression) is investigated. For the complete results, see Additional file 1: Figures S11–S12.

Results generated by the double parameter scan pipeline are shown in Fig. 1d. In this analysis two model parameters are scanned simultaneously and these data are reported for each time point separately. For instance, it can be useful for revealing combinatorial effects of two drugs affecting a timecourse. For the complete results, see Additional file 1: Figures S13–S15. An example of this analysis can be found in [10], where it was applied for exploring the combination of mTOR and ROS treatments in a cellular senescence model.

Discussion

SBpipe is a software tool which allows modellers to automatically repeat certain tasks in model development and analysis, such as parameter estimation and simulation, and obtain additional information about the robustness of the model. Its use should increase productivity and the confidence in the results obtained with the model.

Parameter estimation from experimental data is a challenging task which can easily produce unreliable results due to local minima, parameter non-identifiability, or inadequate optimisation algorithm configuration. From the generation and analysis of a fit sequence, SBpipe can reveal crucial insights about a model structure, the reliability of each parameter, as well as indications about the sufficiency and quality of the experimental data used to calibrate the model. This knowledge is required for assessing whether parameters are well defined and the overall model predictions are reliable.

Several software tools exist to automate aspects model building and simulation tasks, and a comprehensive review of these packages is beyond the scope of this article. Some of these comprehensive packages such as AMIGO2 [11] and SBPOP [12] rely on proprietary software (e.g. Matlab). Condor-COPASI [13] is an example of open source alternative. This server-based software tool integrates COPASI with Condor, a high-throughput computing environment. It allows COPASI users to run and analyse models on a Condor pool. SBpipe distinguishes from Condor-COPASI for three main reasons: 1) although COPASI models are supported, users can run repeated model parameter estimations and simulations using any

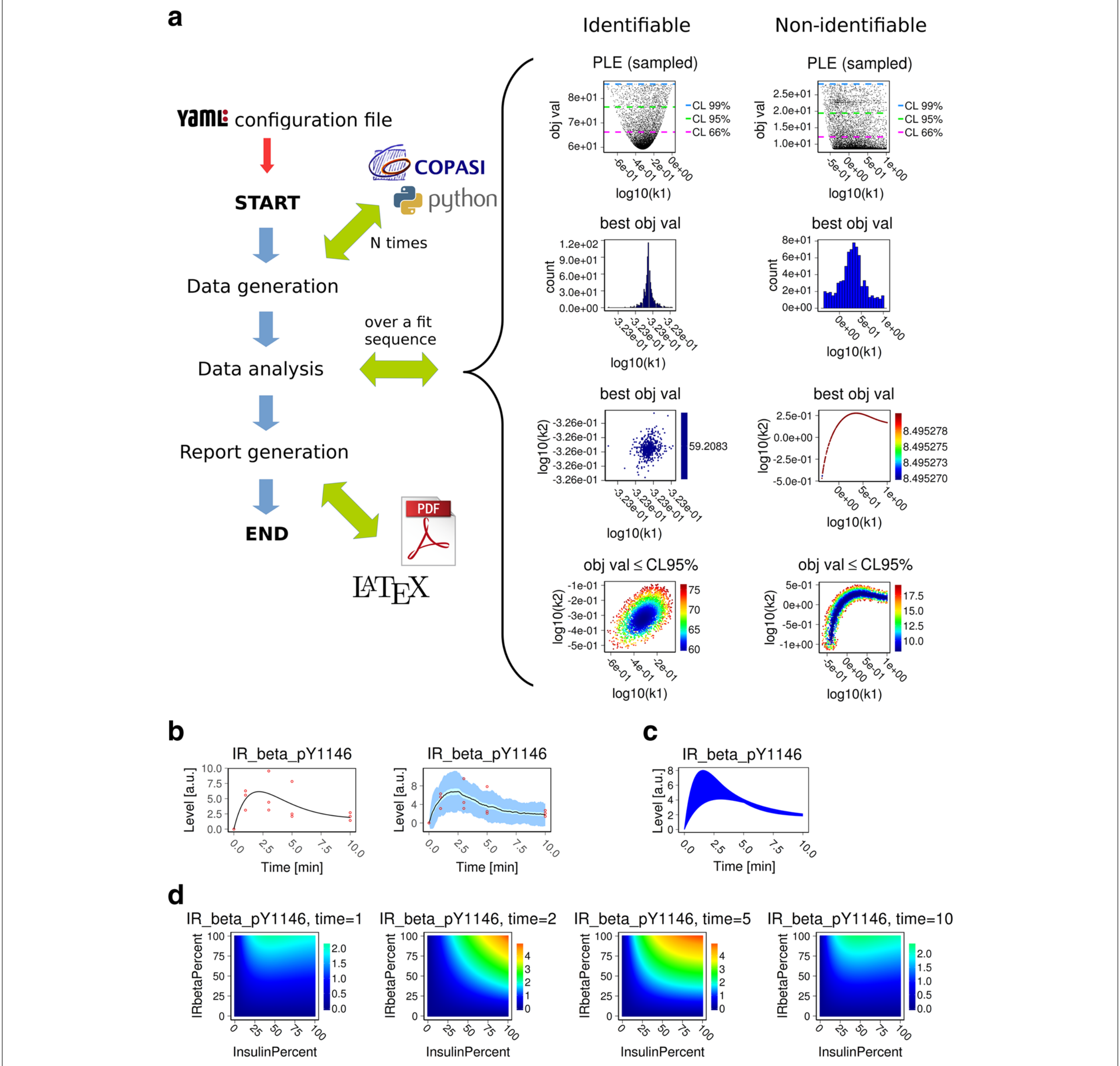

Fig. 1 Implemented pipelines in SBpipe. **a** Example of work flow using the parameter estimation pipeline. Parameter estimations were performed using data sets of different sizes. The *Identifiable* column shows the results using a data set sufficient for estimating the parameters with their confidence intervals, whereas the column *Non-identifiable* illustrates the results using the same model but a reduced data set, insufficient for identifying parameter values. Size of the fit sequence: N=1000. For the complete results generated by this pipeline, see Additional file 1: Tables S2–S4, Figures S2–S8. **b** Deterministic and stochastic model time courses for the phosphorylated IR_beta species obtained with the model simulation pipeline. For stochastic simulations, mean (*black*), 95% confidence interval for the mean (*cyan*), and 1 standard deviation (*light blue*) are reported. Experimental data are added and indicated as red circles. For the complete results, see Additional file 1: Figures S9–S10. **c** Single parameter scan pipeline. The `k1` parameter regulating the IR_beta phosphorylation was scanned within its 95% estimated confidence interval. The blue area is the results of 100 time course simulations over this interval. For the complete results, see Additional file 1: Figures S11–S12. **d** Double parameter scan pipeline. Signal intensities for the phosphorylated IR_beta receptor different levels of Insulin (*x axis*) and IR_beta receptor (*y axis*) at 1, 2, 5, and 10 minutes upon insulin stimulation. The colour representation indicates how the readout signal intensity varies upon two model parameter levels. For the complete results, see Additional file 1: Figures S13–S15. All the results can be replicated using the test files provided within the SBpipe package available online on the GitHub repository

other software or programming library; 2) it is a client-based software tools and therefore it does not require cluster administration; 3) SBpipe can also run locally via multithreading, which is ideal for preliminary testing of the most suitable algorithms for parameter estimation and simulation, before starting intensive jobs on a cluster.

SBpipe requires some familiarity with command line tools, although no programming skill is needed when COPASI models are used. Users only need to create a configuration file and run it using a simple command set. Users with a background in programming languages can also benefit from SBpipe functionalities using mathematical models coded with their preferred language if needed. In contrast to standard pipeline frameworks, SBpipe does not currently offer support for dependency management at coding level and reentrancy at execution level. The former is defined as a way to precisely define the dependency order of functions. The latter is the capacity of a program to continue from the last interrupted task. Although many pipeline frameworks are available for bioinformatics, the definition of a clear and spread standard specifying how pipelines can be configured is still limited in our opinion. In the future we hope to also use a pipeline framework as an additional way to run SBpipe tasks. Benefitting of dependency declaration and execution reentrancy would in particular be beneficial for running SBpipe on clusters or on the cloud.

From an implementation standpoint, SBpipe design is sufficiently generic to permit rapid extension of new pipelines. With this solid but flexible design, SBpipe aims to encourage the development of pipelines for systems modelling into a single community activity.

Conclusions

SBpipe is a novel open source software that enables systems biology modellers to simulate models, scan and estimate model parameters in a large scale. Novel analyses from multiple repeats are also computed via publication quality plots and tables. This project permits to increase productivity and reliability in model building and simulation.

Availability and requirements

Project name: SBpipe
Project home page: https://pdp10.github.io/sbpipe/
Operating system(s): Platform independent
Programming language: Python 2.7+ or 3.4+, R 3.3.0+
Other requirements: COPASI 4.19, TexLive 2013.
License: GNU LGPL v3

Additional file

Additional file 1: Supporting information. Additional file containing supporting Tables S1–S4 and Figures S1–S15. (PDF 29,712 kb)

Abbreviations

CL: Confidence level; IR: Insulin receptor; LSF: Load sharing facility; PLE: Profile likehood estimation; SGE: Sun grid engine

Acknowledgements

We acknowledge Dr Lu Li, Dr An Nguyen, and Dr Pınar Pir for helpful feedback.

Funding

This work was funded by British BBSRC (BBS/E/B/000C0419).

Authors' contributions

PDP and NLN conceived and designed the project; PDP implemented the software. PDP and NLN wrote the manuscript. Both authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Received: 23 February 2017 Accepted: 23 March 2017
Published online: 10 April 2017

References

1. Ghosh S, Matsuoka Y, Asai Y, Hsin KY, Kitano H. Software for systems biology: from tools to integrated platforms. Nat Rev Genet. 2011;12: 821–32.
2. Le Novère N. Quantitative and logic modelling of molecular and gene networks. Nat Rev Genet. 2015;16:146–58.
3. van Rossum G. Python tutorial, Technical Report CS-R9526. Amsterdam: Centrum voor Wiskunde en Informatica (CWI); 1995.
4. R Development Core Team. R: A Language and Environment for Statistical Computing: Vienna; 2008. ISBN 3-900051-07-0.
5. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI - a COmplex PAthway SImulator. Bioinformatics. 2006;22(24):3067–74.
6. Dalle Pezze P, Sonntag A, Thien A, Prentzell M, Gödel M, Fischer S, et al. A Dynamic Network Model of mTOR Signaling Reveals TSC-Independent mTORC2 Regulation. Sci Signal. 2012;5(217):ra25.
7. Le Novère N, Bornstein B, Broicher A, Courtot M, Donizelli M, Dharuri H, et al. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic Acids Res. 2006;34(Database issue):D689–91.
8. Kennedy J, Eberhart R. Particle Swarm Optimization. In: Proceedings of the Fourth IEEE International Conference on Neural Networks (Perth, Australia). Perth: IEEE; 1995. p. 1942–1948.
9. Schaber J. Easy parameter identifiability analysis with COPASI. Biosystems. 2012;110(3):183–5.
10. Dalle Pezze P, Nelson G, Otten E, Korolchuk V, Kirkwood T, von Zglinicki T, et al. Dynamic Modelling of Pathways to Cellular Senescence Reveals Strategies for Targeted Interventions. PLOS Comput Biol. 2014;10(8):1–20.
11. Balsa-Canto E, Henriques D, Gábor A, Banga JR. AMIGO2, a toolbox for dynamic modeling, optimization and control in systems biology. Bioinformatics. 2016;32(21):3357.
12. Schmidt H, Jirstrand M. Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. Bioinformatics. 2006;22(4):514.
13. Kent E, Hoops S, Mendes P. Condor-COPASI: high-throughput computing for biochemical networks. BMC Syst Biol. 2012;6(91). http://bmcsystbiol.biomedcentral.com/articles/10.1186/1752-0509-6-91.