Author’s response to reviews

Title: Modular Programming for Tuberculosis Control, the "AuTuMN" Platform

Authors:

James Trauer (james.trauer@monash.edu)

Romain Ragonnet (romain.ragonnet@burnet.edu.au)

Tan Doan (tan.doan@unimelb.edu.au)

Emma McBryde (emma.mcbryde@jcu.edu.au)

Version: 2 Date: 13 Jul 2017

Author’s response to reviews:

RESPONSE TO EDITORIAL REQUEST TO PROVIDE ADDITIONAL PLATFORM CODE

As outlined in our previous response, although the AuTuMN modular construction is now firmly in place, several of the modules are still under development. Therefore, we have produced a codebase that presents the modular structure, with each module represented by an abbreviated version of the full module.

This codebase has been publicly released under the GitHub name “AuTuMN_framework” at https://github.com/jtrauer/AuTuMN_framework. Each module that we have coded in the full AuTuMN codebase is represented in this repository and each of the major functions of the module is demonstrated, including each module represented as a rectangle in Figure 2 (with the exception of the base module, which we have released in full). Moreover, each module is fully functional such that the codebase runs as a whole using each component module. Therefore, AuTuMN_framework could be used to undertake relatively simple country-level analyses and could rapidly be developed to undertake AuTuMN-like analyses.

We believe this codebase will actually be considerably more useful to TB modellers than releasing our full codebase in its current form. This is because, although AuTuMN is extensively coded and commented, we have not produced user documentation that would allow programmers previously unfamiliar with our platform to fully interact with it and to make further modifications. For example, the curve fitting module produces piecewise polynomial spline functions for use by the data processing module, but has several working approaches to achieve this, not all of which are currently in use or fully explained to unfamiliar users. Similarly, our current GUI is currently being replaced by a more advanced version.
The framework repository includes the following modules:

- **Interface module**
  - Presents all the data structures for the user to interact with the remaining codebase and the basis for creating more visually appealing graphical user interfaces

- **Model runner module**
  - Presents the basis for running the platform with multiple different purposes, with scenario analysis and uncertainty presented as examples (which are purposes that were fully coded by our team and have minimal dependencies)
  - Calculates consistently formatted output structures

- **Data processing module**
  - Formats both fixed and time-variant input parameters for use by the TB model module

- **Spreadsheet module**
  - Reads one external spreadsheet (the WHO’s Global TB Report), making the extracted data available to the data processing module

- **Curve fitting module**
  - Fits a sinusoidal piecewise function to input data

- **Outputs module**
  - Produces graphical outputs from the data structures created by the model runner

- **Tool kit**
  - Stores static functions for use by any other module

- **TB model**
  - Inherits the BaseModel class from the popdynamics module to create a TB model with one tier of stratification

We believe this is the most appropriate approach to ensuring that the codebase is fully transparent to end-users, useful to programmers and consistent with the editorial request and BMC policy.

In the manuscript, the web address of AuTuMN_framework is provided and, the following text:
The “General transmission dynamic model” module has been released as open-source software, including the full code, dependencies and detailed user documentation (see https://github.com/popdynamics/popdynamics). The characteristics specific to each module are presented in Table 1.

Has been highlighted in the document and revised to:

The “General transmission dynamic model” module has been released as open-source software (see https://github.com/popdynamics/popdynamics), while a repository that presents the framework of each platform module is also available (see https://github.com/jtrauer/AuTuMN_framework). These repositories include code, dependencies and detailed user documentation. The characteristics of each module are presented in Table 1, with examples provided in the AuTuMN_framework repository.

RESPONSE TO REVIEWER 2 COMMENTS

1. Example in popdynamics: from the perspective of users, there is no strong incentive to use the package because the current example seems not be more friendly than crafting a model with scipy directly. To attract potential users, however, the examples for the package should be as simple and clear as possible. My suggestions are as follows. First, add a new example of TB with complex stratification and show that the complexity of code does not largely increase. If I am correct, keeping coding complexity while expanding model is an the important features of the package and it is easy to demonstrate. Second, hide the repeated functions in order to increase readability of the code. For example, multiple lines of 'set_compartment' can be simplified as 'set_compartments' with a input of a dictionary to be iterated. In addition, the IO functions, such as 'ensure_dir' and 'def make_plots', can be specified into another file and then be reformulated as a reusable function. It will be more user-friendly to include codes only for model definition and simulation in the example file.

Response: We address the comments around illustrating the increasing model complexity through stratification through the AuTuMN framework module provided in response to the editor’s comments. This includes examples of how to provide more general output functions that deal with variation in model structure.

Initialising empty compartments has been simplified with a loop, such that only compartments that start from positive values need to be explicitly set.

The parameters for the TB model are now set through a dictionary structure, rather than individually. This was the approach used in the SEIR example as an illustration of this more
elegant approach. The individual setting was intended to be more explicit, but we accept that this approach is preferable and so have made this change.

The static functions common to multiple example models (i.e. what are now ensure_out_dir and open_out_dir) have been moved to separate module (named tool_kit.py).

2. Economic step (middle of page 14): I suggest not to mention incremental cost-effectiveness ratios (ICER) but use 'cost-effectiveness analysis' instead. Because the calculation of ICER requires significant difference between the effectiveness terms, it can not always be computed. If this condition fails, the variance of ICER will be diverged and further discussion based on ICER is meaningless. In terms of TB, there are many endemic settings so that the situation is not rare.

Response: “… to estimate incremental cost-effectiveness ratios.” has been revised to “… for cost-effectiveness analysis.” in the last paragraph of the Results section.

3. Simulation time (page 17): The incremental time for adding a scenario was highlighted in the revised example. However, it is more time-consuming in initialising and expanding strata space than adding a scenario. If possible, please adjust the example for addressing advantages and computation bottlenecks.

Response: “Run-time scales approximately multiplicatively with the number of stratifications for each type of stratification, while the time taken for initialisation remains approximately unchanged at around 7 seconds.” has been added to the end of the Discussion section.