---

**Article:**
Dodd, P.J. orcid.org/0000-0001-5825-9347, Pennington, J.J., Bronner Murrison, L. et al. (1 more author) (2018) Simple inclusion of complex diagnostic algorithms in infectious disease models for economic evaluation. Medical Decision Making, 38 (8). pp. 930-941. ISSN 0272-989X

https://doi.org/10.1177/0272989X18807438

---

Journal Title
XX(X):1–??

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

# Simple inclusion of complex diagnostic algorithms in infectious disease models for economic evaluation

**P.J. Dodd[1], J.J. Pennington[2], L. Bronner Murrison[3], and D.W. Dowdy[2]**

## Abstract

**Introduction** Cost-effectiveness models for infectious disease interventions often require transmission models that capture the indirect benefits from averted subsequent infections. Compartmental models based on ordinary differential equations are commonly used in this context. Decision trees are frequently used in cost-effectiveness modelling and are well suited to describing diagnostic algorithms. However, complex decision trees are laborious to specify as compartmental models and cumbersome to adapt, limiting the detail of algorithms typically included in transmission models.

**Methods** We consider an approximation replacing a decision tree with a single holding state for systems where the time scale of the diagnostic algorithm is shorter than time scales associated with disease progression or transmission. We describe recursive algorithms for calculating the outcomes and mean costs and delays associated with decision trees, and design strategies for computational implementation. We assess the performance of the approximation in a simple model of transmission/diagnosis, and its role in simplifying a model of tuberculosis diagnostics.

**Results** When diagnostic delays were short relative to recovery rates, our approximation provided a good account of infection dynamics and the cumulative costs of diagnosis and treatment. Proportional errors were below 5% so long as the longest delay in our two-step algorithm was under 20% of the recovery time scale. Specifying new diagnostic algorithms in our tuberculosis model was reduced from several tens to just a few lines of code.

**Discussion** For conditions characterized by a diagnostic process that is neither instantaneous nor protracted (relative to transmission dynamics), this novel approach retains the advantages of decision trees while embedding them in more complex models of disease transmission. Concise specification and code re-use increases transparency and reduces potential for error.

## Keywords

cost-effectiveness, infectious diseases, decision trees, transmission models, diagnostic algorithms, computational techniques.

## Introduction

Interventions to control epidemics of infectious diseases generally aim to reduce transmission at the population level, not simply improve outcomes for individual patients who are infected. For example, vaccination campaigns often seek not just to prevent disease in those individuals vaccinated, but also to reduce the population burden of disease through herd immunity; vaccination of every individual is generally not required to eliminate the infection from the population.(Keeling and Rohani 2011) Similarly, diagnostic tests that allow detection of an infection at an earlier stage of disease could not only avert morbidity among individuals who are diagnosed, but can also avert transmission from those individuals to others in the population.(Menzies et al. 2012; Paltiel et al. 2005) As a result, unlike chronic diseases, impact and health economic evaluation of interventions for infectious diseases usually require models that incorporate mechanisms of transmission - in other words, the risk of infection at any given time should depend on the number of infectious individuals at that time. Models that fail to incorporate transmission risk underestimating the true benefit of infectious disease interventions at the population level.(Jit and Brisson 2011; Pitman et al. 2012)

The process of diagnosing infectious diseases often follows some specified algorithm. For example, one might perform a screening test that is later confirmed with a more specific (but more expensive) confirmatory test (e.g., HIV testing), or one might perform tests based on screening criteria, or even simply treat without testing depending on symptom severity (e.g., evaluation of respiratory infections in children). When applied to economic evaluations of diagnostic algorithms requiring even moderate complexity - and especially when the algorithm itself is under study - transmission models face a fundamental dilemma. Specifically, simple transmission models (often described mathematically as sets of ordinary differential equations representing rates of flow between health states) cannot easily manage the many 'holding' states (e.g., completed diagnostic test 1, awaiting results of test 2) that such algorithms require. Moreover, including a different algorithm will often require a fundamental overhaul of model structure.

Individual-based transmission models can incorporate complexity of this type flexibly, but at a substantial cost in terms of computational expense and ease of analysis.

[1] School of Health and Related Research, University of Sheffield, UK [2] Department of Epidemiology, Johns Hopkins Bloomberg School of Public Health, Baltimore, USA [3] Department of Environmental Health, University of Cincinnati College of Medicine, USA

**Corresponding author:**
P.J. Dodd, 30 Regent Street, Sheffield S1 4DA, UK
Email: p.j.dodd@sheffield.ac.uk

A method for incorporating such algorithms into transmission models, without requiring separate model states for each 'holding' transition, would therefore be an important advance.

Here, we describe a method by which algorithmic diagnostic interventions can be incorporated into transmission models of infectious diseases through use of decision trees. Decision trees are attractive and widely used representations of such interventions due to their conceptual simplicity, flexibility, and the wide availability of software for their implementation.(Brennan et al. 2006) However, decision trees do not traditionally include the time spent at each branch of the tree; this 'holding' time is essential to include in infectious disease models because individuals remain infectious during this time. We illustrate how simple approximations can incorporate diagnostic algorithms for which the 'holding' times for diagnosis are non-negligible, but still shorter than the timescale of the overall infectious disease course - arguably the most common situation in infectious disease diagnosis. We first describe the method in a highly simplified model, then we illustrate the use of this method in a more complex model of diagnostic algorithms for tuberculosis (TB).

## Methods

### *Case study 1: A simple screen/confirm diagnostic algorithm*

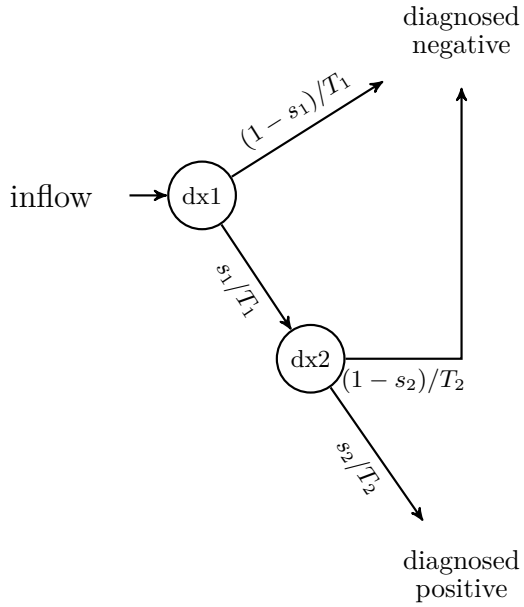
To begin with and to introduce our approximation, we consider the simple example of a screen/confirm two-step diagnostic algorithm. This algorithm is represented by the decision tree in Figure 1a. The first diagnostic (denoted dx1) is assumed to have sensitivity $s_1$ and and specificity $sp_1$. The second diagnostic (denoted dx2) is assumed to have sensitivity $s_2$ and specificity $sp_2$. It is follows that the overall sensitivity of the algorithm (in which the tests are applied sequentially, and a positive overall result requires a positive result on both tests) is $s_1 \cdot s_2$, and the overall specificity is $sp_1 + (1 - sp_1) \cdot sp_2$.

We can summarize this information in a transition matrix for the test that gives the probabilities for both true negative or true positive individuals (rows) to receive a negative or positive diagnosis (columns) overall:

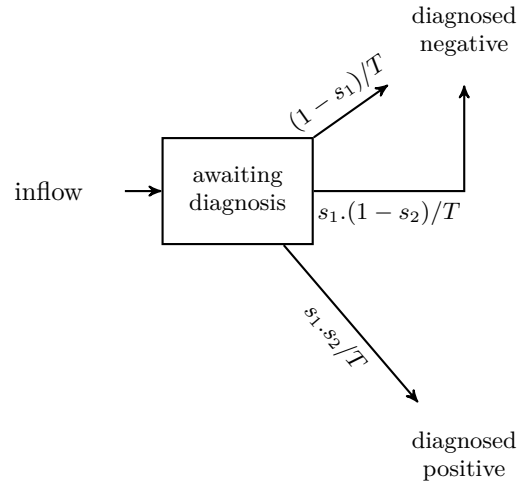
$$P = \begin{array}{c} \\ \text{true -ve} \\ \text{true +ve} \end{array} \begin{array}{c} \begin{array}{cc} \text{diagnosed -ve} & \text{diagnosed +ve} \end{array} \\ \begin{pmatrix} sp & 1-sp \\ 1-s & s \end{pmatrix} \end{array}. \quad (1)$$

Note that the rows sum to one, $\sum_j P_{ij} = 1$, as the possibilities represented by the columns are exhaustive.

Compartmental models with constant transition rates can be thought of in terms of stocks and flows, with the total outflow from a compartment equal to the inverse of the mean delay. In this context, language and intuition is often borrowed from the case of constant-rate continuous-time Markov processes, where compartment stocks model probability, and the differential equations are the associated Kolmogorov forward equations (master equations). Here, compartment sojourn times




**(a)** Before approximation




**(b)** After approximation

**Figure 1.** Approximating a decision tree with a single compartment. The rates for the arrows are shown for true positives. Figure 1a shows a simple two-stage screen/confirm diagnostic algorithm using tests dx1 and dx2. Figure 1b shows the mean time approximation representing those awaiting diagnosis by a single compartment. The outflow rates for true positive individuals label the arrows, with notation explained in the text.

are exponentially distributed, and the probability of leaving a state via one flow rather than another is given by the relative magnitude of the corresponding rates of flow. Flow rates are therefore often parametrized in terms of the mean time spent in a given compartment, multiplied by the relative probabilities of each exit flow.

Given delays $T_1$ and $T_2$ for diagnostics dx1 and dx2 respectively, one can naturally view a decision tree such as that of Figure 1a as specifying a compartmental model with each node being a compartment. The

transition rates for a true positive individual are indicated on the arrows in Figure 1a.

We will use a superscript $i = 0, 1$ to denote true negatives or true positives, respectively, so that the population count in a state is the sum of those truly negative and truly positive: $X = X^{(0)} + X^{(1)}$. For inflow $I(t)$, the differential equations governing the number in the dx1 and dx2 compartments ($D_1$ and $D_2$, respectively), and the cumulative number of negative and positive tests ($N$ and $Y$, respectively) are

$$\begin{aligned}
\frac{\mathrm{d}D_1^{(i)}}{\mathrm{d}t} &= I^{(i)}(t) - \frac{D_1^{(i)}}{T_1} \\
\frac{\mathrm{d}D_2^{(i)}}{\mathrm{d}t} &= A_{i1} \cdot \frac{D_1^{(i)}}{T_1} - \frac{D_2^{(i)}}{T_2} \\
\frac{\mathrm{d}N^{(i)}}{\mathrm{d}t} &= A_{i0} \frac{D_1^{(i)}}{T_1} + B_{i0} \cdot \frac{D_2^{(i)}}{T_2} \\
\frac{\mathrm{d}Y^{(i)}}{\mathrm{d}t} &= B_{i1} \cdot \frac{D_2^{(i)}}{T_2} \,,
\end{aligned} \tag{2}$$

where $A$ represents the transition matrix associated with dx1 (with elements $A_{ij}$), and $B$ the transition matrix associated with dx2 (with elements $B_{ij}$). Labelling those testing negative to dx1 as negative can be represented by transition matrix $C$, where $C_{i0} = 1$ and $C_{i1} = 0$.

The overall transition matrix, $P$ for a two-step decision tree as in Figure 1a can be obtained as

$$P_{ij} = \sum_k A_{ik} P_{ij}^{(k)} \,, \tag{3}$$

where $A$ is the transition matrix of the first step and $P^{(k)}$ is the transition matrix in the second step associated with the $k$-th outcome of step one. This amounts to simply summing the probabilities associated with independent ways of realising a given outcome for each possible input. For the case above, it is easy to verify that with $P^{(0)} = C$ and $P^{(1)} = B$, Equation 3 yields the expected overall sensitivity and specificity for the sequential two-step algorithm:

$$P = \begin{pmatrix} sp_1 + (1 - sp_1) \cdot sp_2 & (1 - sp_1) \cdot (1 - sp_2) \\ (1 - s_1) + s_1 \cdot (1 - s_2) & s_1 \cdot s_2 \end{pmatrix} . \tag{4}$$

Similar equations could be applied for diagnostic tests performed in parallel (i.e., where a positive test on either component test would be considered an overall positive result).

## A mean time-scale approximation

Our approximation will be to replace all compartments in a decision tree with a single holding state. We want the proportion of those exiting the holding state via each route to equal the outcome probabilities computed for the corresponding terminal nodes in the decision tree. We furthermore calculate the mean delay incurred over the entire decision tree (for a patient of given characteristics) and use that to assign a mean time spent in the holding state. This approximation is equivalent to assuming that all individuals with given characteristics undergoing a diagnostic algorithm will have the same delay from beginning of diagnosis to the end of the diagnostic process (e.g., treatment initiation), regardless of the actual diagnostic test result. We expect this approximation to work best when the time scales of the internal dynamics of the decision tree are short compared with the dynamic time scales of the overall model - a typical scenario in infectious disease diagnosis.

In our case, this means replacing the compartmental diagram with that shown in Figure 1b. With the mean delays $T^{(0)} = T_1 + (1 - sp_1).T_2$ for true negatives and $T^{(1)} = T_1 + s_1.T_2$ for true positives, and $D$ representing the population counts in the single awaiting diagnosis compartment, this approximation corresponds to the differential equations

$$\begin{aligned}
\frac{\mathrm{d}D^{(i)}}{\mathrm{d}t} &= I^{(i)}(t) - \frac{D^{(i)}}{T^{(i)}} \\
\frac{\mathrm{d}N^{(i)}}{\mathrm{d}t} &= P_{i0} \cdot \frac{D^{(i)}}{T^{(i)}} \\
\frac{\mathrm{d}Y^{(i)}}{\mathrm{d}t} &= P_{i1} \cdot \frac{D^{(i)}}{T^{(i)}} \,.
\end{aligned} \tag{5}$$

Here, $P$ is the transition matrix associated with the two-step algorithm, given above in Equation 4.

In this simple example, the saving in complexity is not great. However, in more complex algorithms with more patient characteristics, this can save tens of differential equations. Moreover, it greatly facilitates incorporation and comparison of algorithms with differing numbers of diagnostics, since under this approximation the model structure remains the same, with only the rates changing.

This same technique can be extended to handle the mean costs for patients with given characteristics by considering the costs of passing through the decision tree, and associating the cost of passing through the single holding-state with the decision tree mean, analogously to the delay.

## Case study 2: An epidemic model with treatment

In this section, we consider a more complex example of embedding our simple diagnostic algorithm into an infectious disease model. In the Results section, we evaluate the performance of our approximation in this dynamic model.

The SIR model is the archetypal infectious disease model, introduced by Kermack and McKendrick in 1927. (Kermack and McKendrick 1927) The population is divided into Susceptible, Infectious and Recovered compartments, with infectious individuals spontaneously recovering at a constant rate (which we will denote $\nu$), and infection occurring at a rate proportional to the prevalence of infectious individuals in the population with some coefficient, traditionally denoted $\beta$. While simple, this model exhibits many key features of real systems, including threshold behaviour for critical levels of population immunity.

We extend the SIR model to include diagnosis and treatment (Figure 2). Our simple two-stage screen/confirm diagnostic tree appears once for susceptible, infectious, and recovered classes, determining whether these individuals end up on treatment. Infectious individuals

are assumed to seek diagnosis at a constant rate, in addition all individuals have a diagnosis seeking rate due to symptoms unrelated to the infection of interest. We assume that infectious individuals remain infectious while under diagnosis (i.e. in states $ID_1$ and $ID_2$ in Figure 2), but are not infectious while on treatment or once treatment is completed (the 'R past treatment' box in Figure 2). We assume that those who have previously been treated are not eligible for further treatment and therefore do not re-enter the diagnostic cascade.

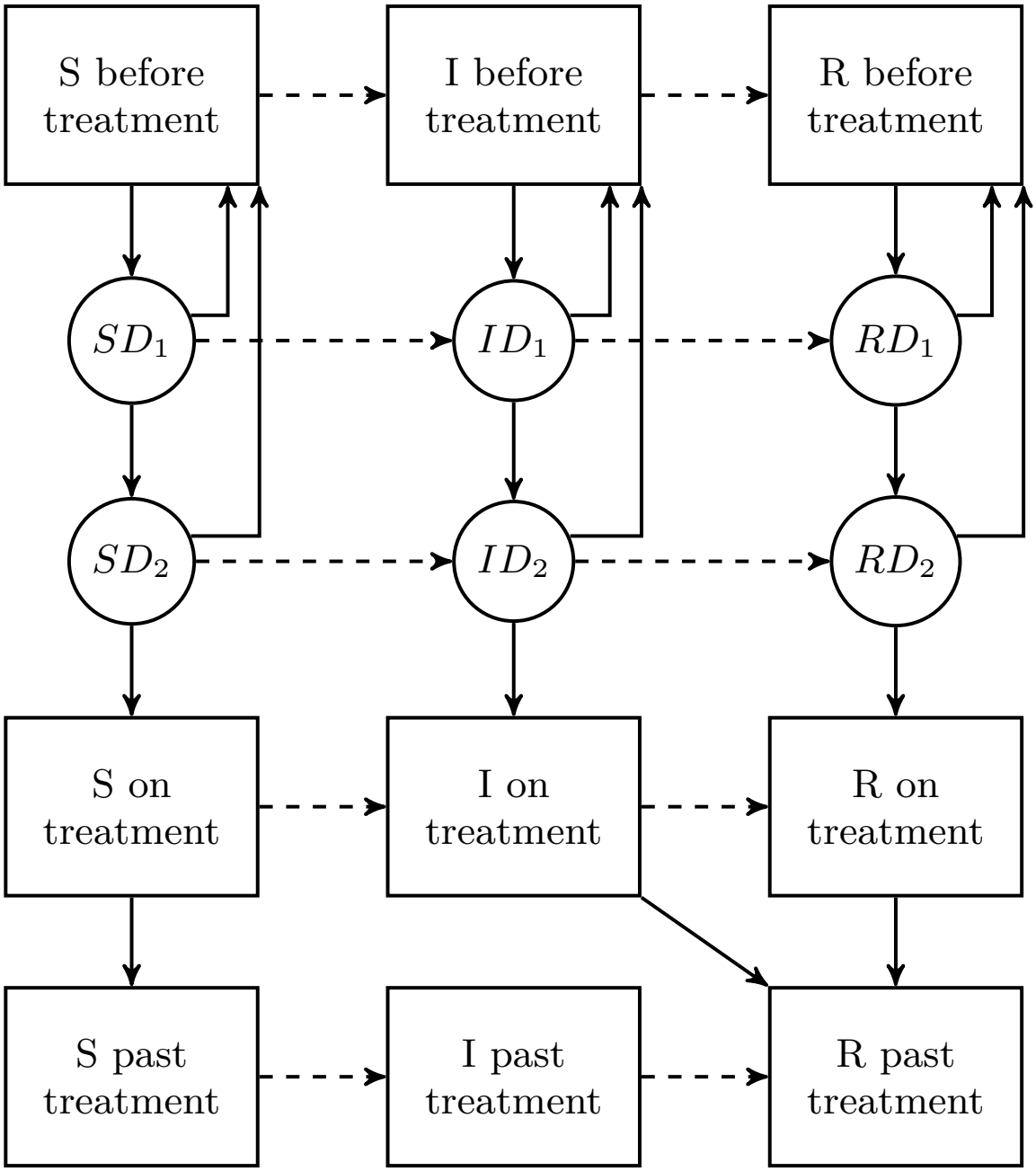



**Figure 2.** A susceptible-infected-recovered (SIR) transmission model with treatment. The diagnostic algorithm is that depicted in Figure 1a. Dashed lines represent infection and recovery processes, solid lines transitions between treatment and diagnosis states. Those previously treated are not eligible for diagnosis. Different rates of seeking diagnosis apply for true negatives (S & R) compared to true positives (I). States in circles correspond to stages in a diagnostic algorithm that will be simplified using our approximation (see text).

We will write $S_j$ for susceptibles: $j = 0$ for treatment-naive; $j = 1$ for those on treatment; and $j = 2$ for those-post treatment. We use analogous notation for infectious ($I_j$) and recovered ($R_j$). We write $SD_1$ and $SD_2$ for susceptibles awaiting diagnostic 1 (dx1) and 2 (dx2), respectively, and analogously for infectious and recovered. If $N$ is the total population, and $\beta$ the effective contact rate, the force-of-infection can be written $\lambda = \frac{\beta}{N}(ID_1 + ID_2 + I_0 + I_2)$. (Those infected on treatment are assumed non-infectious; those infected after treatment are infections.) Finally, we will denote the constant rate with which infectious individuals seek diagnosis by $b$, the diagnosis-seeking rate for reasons other than the infection among all individuals by $a$ (operating in addition to rate $b$ for infectious individuals), and the duration of treatment by $T_T$ (note the dual role of 'T'). The differential equations corresponding to Figure 2 are then:

$$
\begin{aligned}
\frac{\mathrm{d}S_j}{\mathrm{d}t} &= -\lambda \cdot S_j + \left(-a \cdot S_0 + sp_1\frac{SD_1}{T_1} + sp_2\frac{SD_2}{T_2}\right)\delta_{j0} \\
&+ (1-sp_2)\cdot\frac{SD_2}{T_2}\delta_{j1} + (\delta_{j2}-\delta_{j1})\frac{S_1}{T_T} \\
\frac{\mathrm{d}I_j}{\mathrm{d}t} &= \left((1-s_1)\frac{ID_1}{T_1} + (1-s_2)\frac{ID_2}{T_2} - (a+b)\cdot I_0\right)\delta_{j0} \\
&+ \lambda.S_j - \nu\cdot I_j + s_2\frac{ID_2}{T_2}\delta_{j1} - \delta_{j1}\frac{I_1}{T_T} \\
\frac{\mathrm{d}R_j}{\mathrm{d}t} &= +\nu\cdot I_j + \left(sp_1\frac{RD_1}{T_1} + sp_2\frac{RD_2}{T_2} - a\cdot R_0\right)\delta_{j0} \\
&+ (1-sp_2)\cdot\frac{RD_2}{T_2}\delta_{j1} + (\delta_{j2}-\delta_{j1})\frac{R_1}{T_T} + \delta_{j2}\frac{I_1}{T_T} \\
\frac{\mathrm{d}(SD_1)}{\mathrm{d}t} &= a\cdot S_0 - \frac{SD_1}{T_1} - \lambda\cdot SD_1 \\
\frac{\mathrm{d}(SD_2)}{\mathrm{d}t} &= (1-sp_1)\cdot\frac{SD_1}{T_1} - \frac{SD_2}{T_2} - \lambda\cdot SD_2 \\
\frac{\mathrm{d}(ID_1)}{\mathrm{d}t} &= (a+b)\cdot I_0 - \frac{ID_1}{T_1} + (\lambda\cdot SD_1 - \nu ID_1) \\
\frac{\mathrm{d}(ID_2)}{\mathrm{d}t} &= s_1\cdot\frac{ID_1}{T_1} - \frac{ID_2}{T_2} + (\lambda.SD_2 - \nu ID_2) \\
\frac{\mathrm{d}(RD_1)}{\mathrm{d}t} &= a\cdot R_0 - \frac{RD_1}{T_1} + \nu ID_1 \\
\frac{\mathrm{d}(RD_2)}{\mathrm{d}t} &= (1-sp_1)\cdot\frac{RD_1}{T_1} - \frac{RD_2}{T_2} + \nu ID_2\,.
\end{aligned}
\tag{6}
$$

Here, the quantity $\delta_{ij}$ is the standard Kronecker delta: equal to one when its indices are equal and zero otherwise.

Our approximation means replacing each of the three decision trees in Figure 2 with a single 'in diagnosis' compartment, which we will denote $SD$ for susceptibles, $ID$ for infectious, and $RD$ for recovered. This results in the differential equations

$$
\begin{aligned}
\frac{\mathrm{d}S_j}{\mathrm{d}t} &= -\lambda\cdot S_j + \left(-a\cdot S_0 + sp\frac{SD}{T_n}\right)\delta_{j0} \\
&+ (1-sp)\cdot\frac{SD}{T_n}\delta_{j1} + (\delta_{j2}-\delta_{j1})\frac{S_1}{T_T} \\
\frac{\mathrm{d}I_j}{\mathrm{d}t} &= +\left((1-s)\frac{ID}{T_p} - (a+b)\cdot I_0\right)\delta_{j0} \\
&+ \lambda\cdot S_j - \nu\cdot I_j + s\frac{ID}{T_p}\delta_{j1} - \delta_{j1}\frac{I_1}{T_T} \\
\frac{\mathrm{d}R_j}{\mathrm{d}t} &= +\left(sp\frac{RD}{T_n} - a\cdot R_0\right)\delta_{j0} + (1-sp)\cdot\frac{RD}{T_n}\delta_{j1} \\
&+ \nu\cdot I_j + (\delta_{j2}-\delta_{j1})\frac{R_1}{T_T} + \delta_{j2}\frac{I_1}{T_T} \\
\frac{\mathrm{d}(SD)}{\mathrm{d}t} &= a\cdot S_0 - \frac{SD}{T_n} - \lambda\cdot SD \\
\frac{\mathrm{d}(ID)}{\mathrm{d}t} &= (a+b)\cdot I_0 - \frac{ID}{T_p} + (\lambda\cdot SD - \nu ID) \\
\frac{\mathrm{d}(RD)}{\mathrm{d}t} &= a\cdot R_0 - \frac{RD}{T_n} + \nu ID\,,
\end{aligned}
\tag{7}
$$

where now $\lambda = \frac{\beta}{N}(ID + I_0 + I_2)$, and $T_p = T_1 + s_1T_2$ is the mean time spent in diagnosis for true positives, $T_n = T_1 + (1-sp_1)T_2$ is the mean time spent in diagnosis for true negatives, $s = P_{11}$ is the sensitivity of the algorithm from Equation 4, and $sp = P_{00}$ is the specificity of the algorithm.

### *Sensitivity analysis*

To quantify the effect of model parameters on approximation accuracy, we undertook a sensitivity analysis for the error in the numbers on treatment, measured as the maximum proportional error over a 30 unit time horizon. Due to the non-linear nature of the model, we calculated Sobol' total sensitivity indices (Sobol' 2001) using the SAlib Python module.(Herman and Usher 2017) We used a Saltelli sample and ran the model on 15,000 distinct parameter sets. The following 13 parameters were included with uniform distributions: timescales $T_1$, $T_2$, $a^{-1}$, $b^{-1}$, $T_T$; test characteristics $s_1$, $s_2$, $sp_1$, $sp_2$; and other parameters $C_1$, $C_2$, $\beta$, $x_0$. For timescales we used ranges from 0.01 to 1 (as a fraction of the infection recovery timescale) in line with our understanding of when this approximation approach is valid, except for the treatment duration ($T_T$) which we varied from 0.5 to 1.5. Test sensitivities and specificities ranged from 0.5 to 1.0. Other parameters ranged from 0.5 to 1.5, except for $\beta$ (1.5 to 3) and $x_0$ (1 to 10 from a total population of 10,000). In other outputs, our default parameters were 0.7 for sensitivities, 0.9 for specificities, 0.1 for timescales (except $b^{-1}$ and $T_T$, taken as 1.0). Defaults for costs were $C_1 = 1.0$ and $C_2 = 10.0$, and $\beta = 2$ for the transmission parameter.

### *Generalizable implementation using recursion*

While the approximation representing a decision tree with a single holding state for the mean time may save coding a number of differential equations, the outcome probabilities and mean sojourn times for

the decision tree still need to be calculated, which could be laborious of not handled efficiently. However, representing the data associated with a single chance node in a decision tree by a transition matrix giving the outcome probabilities (columns) for patients of a given type (rows), as in Equation 1, together with the tree structure of an algorithm allows us to use recursion to powerfully simplify the relevant calculations.

**Algorithm 1** Transition matrix for decision tree by recursion

**function** GetTransition(root of tree)
  $P_{ij} \leftarrow 0 \quad \forall i, j$
  $A_{ij} \leftarrow$ transition matrix for root of tree
  **if** root has children **then**
    **for** $k \in$ child nodes of tree **do**
      $B^{(k)}_{ij} \leftarrow$ GetTransition(root of $k$-th child)
      $P_{ij} \leftarrow P_{ij} + A_{ik} B^{(k)}_{ij}$
  **else**
    $P_{ij} \leftarrow A_{ij}$
  **return** $P_{ij}$

Algorithm 1 shows a pseudocode definition of a function that calculates the overall transition matrix for an arbitrarily complex decision tree. The function uses Equation 3 to combine each node's transition matrix with that of the attached outcome nodes that immediately follow (i.e. a node's children). The algorithm avoids anything more complicated than a for-loop over a node's children by being recursively defined in terms of itself. In this way, calling the function on the root (first node) of the decision tree will result on the function being called on all of the root's children, and so on, traversing down to the terminal nodes (leaves) of the tree, whereupon all of the relevant data is returned and gathered as all the function calls unwind back up the tree. In this way, the answer returned to the top-level function call combines the contribution from all routes through the tree and returns the overall transition matrix for the tree.

For our example, in Figure 1a, calling GetTransition on node dx2 results yields $P^{(2)}$ - the transition matrix of this node - since dx2 has no child nodes. Similarly, GetTransition called on the terminal node 'diagnosed negative' returns its transition matrix $C$ (defined below Equation 2). Calling GetTransition on dx1 (the root of this tree) will initialise $P$ to zero and set $A = P^{(1)}$. The for-loop will then find the first child: the terminal node 'diagnosed negative' (the $k$ =0-th outcome for dx1) and obtain $B^{(0)} = C$ as its transition matrix by calling GetTransition. The rows of $C$ will be multiplied by the 0-th column of $P^{(1)}$ (which we achieve through matrix multiplication by $\mathrm{diag}(P^{(1)}_{00}, P^{(1)}_{10})$ in the working below), and this added to the current $P$ (which is 0). The next node found is dx2 (the $k$ =1-st outcome of dx1) and $B^{(1)} = P^{(2)}$ is obtained by the call go GetTransition on this node. The rows of $P^{(2)}$ will be multiplied by the 1-st column of $P^{(1)}$ (which again we achieve through matrix multiplication by $\mathrm{diag}(P^{(1)}_{01}, P^{(1)}_{11})$ in the working below) and this added to the current $P$. The for-loop therefore calculates

$$
\begin{aligned}
P_{ij} &= \overbrace{0}^{\text{initial value}} + \overbrace{P^{(1)}_{i0} C_{ij}}^{\text{from 'diagnosed negative'}} + \overbrace{P^{(1)}_{i1} P^{(2)}_{ij}}^{\text{from dx2}} \\
&= \left[ \begin{pmatrix} sp_1 & 0 \\ 0 & 1-s_1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \right]_{ij} + \\
&\quad \left[ \begin{pmatrix} 1-sp_1 & 0 \\ 0 & s_1 \end{pmatrix} \begin{pmatrix} sp_2 & 1-sp_2 \\ 1-s_2 & s_2 \end{pmatrix} \right]_{ij} \\
&= \begin{pmatrix} sp_1 + (1-sp_1)\cdot sp_2 & (1-sp_1)\cdot(1-sp_2) \\ (1-s_1) + s_1 \cdot (1-s_2) & s_1 \cdot s_2 \end{pmatrix}_{ij},
\end{aligned}
$$

which indeed is then returned as the correct transition matrix associated with this tree.

**Algorithm 2** Mean costs of transitions in a decision tree by recursion

**function** GetCosts(root of tree)
  $A_{ij} \leftarrow$ transition matrix for root of tree
  $c \leftarrow$ cost associated with root of tree
  $C_{ij} \leftarrow c \times P_{ij}$
  **if** root has children **then**
    **for** $k \in$ child nodes of tree **do**
      $B^{(k)}_{ij} \leftarrow$ GetCosts(root of $k$-th child)
      $C_{ij} \leftarrow C_{ij} + A_{ik} B^{(k)}_{ij}$
  **return** $C_{ij}$

Algorithm 2 shows how this strategy can be modified to compute mean costs (or analogously, mean times spent in the decision tree) for each patient type and outcome by including the cost (or time-scale) associated with each node. We may only be interested in the mean time or cost for each patient type (row of the overall transition matrix) rather than the relative contribution from each outcome (column of the transition matrix); these can be computed by simply summing the columns.

To take advantage of this approach requires a programming language that supports recursively defined functions, and allows easy definitions of data structures or classes to represent trees. We use Python here, (Python Software Foundation 2017) which has a particularly simple class system, and provide a class definition for a diagnostic as an Appendix. This class simply stores the relevant data associated with a diagnostic node (transition matrix, cost, delay, etc.) and allows the node to point to subsequent nodes of the same class, corresponding to the outcomes of the test (the transition matrix columns, here positive or negative). If no further nodes are pointed to by the diagnostic, the positive or negative outcomes are assumed to be definitive diagnostic outcomes associated with the leaves (terminal nodes) of the decision tree. The method `getTables()`, implements algorithms 1 and 2 to compute transition probabilities, costs and delays associated of full diagnostic evaluation. The code for all numerical experiments is available as a supplementary file, and at `https://github.com/petedodd/homebrewdx`.

In this way, diagnostics can be joined together into arbitrarily complex diagnostic algorithms, and the

necessary outcome probabilities, mean delays and mean costs can be obtained by calling these methods at the root node. The class can be extended (potentially by inheritance) with simple helper functions that facilitate merging the decision tree into the specific system of differential equations defining the dynamic model.

### *Case study 3: A model of tuberculosis transmission and diagnosis*

In this section, we briefly describe a previously published dynamic model of tuberculosis (TB) transmission, diagnosis and treatment (FlexDx), which will serve as a more complex case study. (Dowdy et al. 2014)

The rationale for this model is to serve as a generic tool for projecting the epidemic and budgetary impacts of introducing new diagnostic algorithms for TB. The original model represents the natural history and transmission of TB as a series of 100 ordinary differential equations. As with the SIR model with treatment in Figure 2, every class in the population is doubled to record treatment history; HIV infection status, tuberculosis drug-resistance type also introduce additional states. TB disease becomes either smear-positive or negative, which influences infectiousness and the sensitivity of diagnostic tests.

The outcomes of a test or algorithm are: (a) TB negative, (b) TB positive and requiring first-line treatment, or (c) TB positive and requiring second-line treatment (for drug-resistant TB). We constructed separate diagnostic trees according to HIV and previous treatment status (2×2 states). Transition matrices were structured as arrays with the first index coding smear status, and the second and third indices specifying matrices where the rows correspond to the TB types in the model: no TB, drug-sensitive TB, isoniazid-resistant TB (INH-R TB), multi-drug resistant TB (MDR TB, i.e., resistant to both isoniazid and a second drug, rifampin); and the rows to diagnostic/treatment outcomes: diagnosed negative, positive for TB, and positive for multidrug-resistant TB. Note that there is no attempt, in this model, to diagnose INH resistance (as the same treatment is recommended for DS-TB and INH-R TB); this state is included only because it represents a higher risk of subsequently progressing to MDR-TB.

These matrices were parametrized as

$$P = \begin{array}{l} \\ \text{TB -ve} \\ \text{DS TB} \\ \text{INH-R TB} \\ \text{MDR TB} \end{array} \begin{array}{c} \begin{array}{ccc} \text{diagnosed -ve} & \text{diagnosed +ve (DS)} & \text{diagnosed +ve (DR)} \end{array} \\ \begin{pmatrix} sp & (1-sp) & 0 \\ 1-s & s \times sp_{DR} & s \times (1-sp_{DR}) \\ 1-s & s \times sp_{DR} & s \times (1-sp_{DR}) \\ 1-s & s \times (1-s_{DR}) & s \times s_{DR} \end{pmatrix} \end{array}, \quad (8)$$

where (as above) $s$ and $sp$ are the sensitivity and specificity (respecitvely) of the test for detecting TB; $s_{DR}$ and $sp_{DR}$ are the sensitivity and specificity (respecitvely) of the test if positive for TB for detecting multidrug resistance.

Ultimately, this model is intended to be flexible enough to calibrate to many different settings and intervention options, and to be accessible online to non-specialists. The main model allows users to select from a pre-defined set of diagnostic algorithms, but user-specified diagnostic algorithms are also possible. When shown the original model with its pre-defined diagnostic algorithms, a primary request from users was to have the ability to specify their own diagnostic algorithm, including the ability to input the performance and cost of each test in the customized algorithm. There was therefore a demand to develop a simple and re-usable application through which custom algorithms could be specified.

With classes described above and in the Appendix, a diagnostic test for patients of a given treatment history and HIV status could be defined in the model Python code by specifying the sensitivities for TB in smear-positive and smear-negative cases as:

```
Test = Diagnostic( [sens0, sens1],
                   spec, DRsens, DRspec,
                   cost, delay, ltfu )
```

where sens0 and sens1 are the test sensitivities for TB in smear-positive and smear-negative cases, respectively; spec is the test specificity for TB ($sp$); DRsens and DRspec are the test sensitivity and specificity for drug-resistant TB, respectively ($s_{DR}$ and $sp_{DR}$); and cost, delay, and ltfu are respectively the cost, delay and probability of being lost to follow-up for the test.

Tests could then be combined into a more complex algorithms by working back from the leaves of the decision tree and using a method that specifies the test that follows, e.g. after instantiating Test2, we could set:

```
Test2.setNext( 0, Test )
```

to specify use of the diagnostic Test as the outcome 0 for Test2 (the 0 referring to the outcome of the first column of the transition matrix of Test2, which is of the form Equation 8, i.e. a negative test). If the same test is to be used in multiple places in an algorithm, the defining code need appear only once with copies being used elsewhere.

In this way, a handful of lines of code can specify an algorithm, and the relevant probabilities and delays be computed by a single function call at the root node. If the decision tree is approximated by a single holding state in the dynamic compartmental model, this means that trees with entirely different structures can be handled in the same way, without re-writing the code defining the dynamics. The code for this model is open source available at `https://github.com/JJPennington/FlexDx-Xpert-Scale-Up`.

## Results

### *Case studies 1 and 2*

At equilibrium, the proportion of the population (true positive and true negative) that ultimately receives diagnosis or no diagnosis will correspond to the probabilities of these events, and the mean delay incurred during diagnosis (for true positives) will be given by

$$T = T_1 + s_1 \cdot T_2 \,. \quad (9)$$

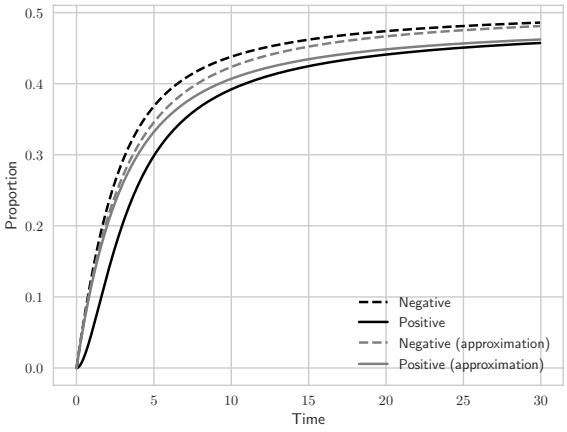



**Figure 3.** Performance of mean time approximation for constant inflow to simple screen/confirm diagnostic algorithm. True positives are solid lines; true negatives are dashed lines; approximations are in red. Here, $s_1 = s_2 = 0.7$, $sp_1 = sp_2 = 0.9$, and $T_1 = T_2 = 0.5$.

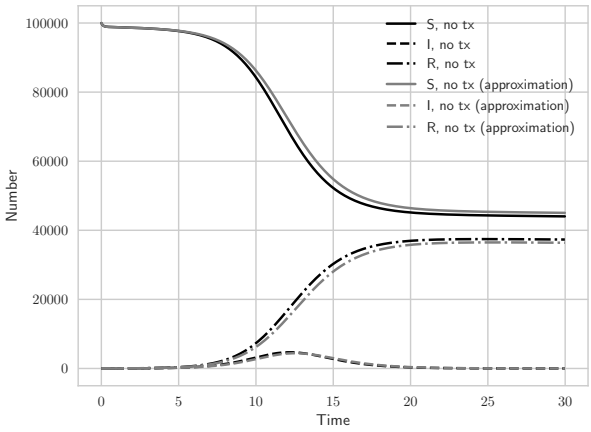


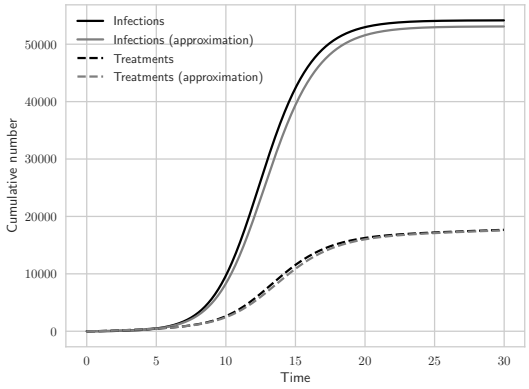

**Figure 4.** Dynamics for the SIR and treatment model of Figure 2 (black) compared with the corresponding dynamics in the mean time approximation (grey). 'tx' denotes treatment. Diagnostic time scales are taken to be 10 times shorter than the recovery time scale, $\nu = 1$ and $\beta = 2$, $s_1 = s_2 = 0.7$, and $sp_1 = sp_2 = 0.9$.

This mean delay corresponds to undergoing the first diagnostic procedure with certainty, but only reaching the second procedure with a certain probability.

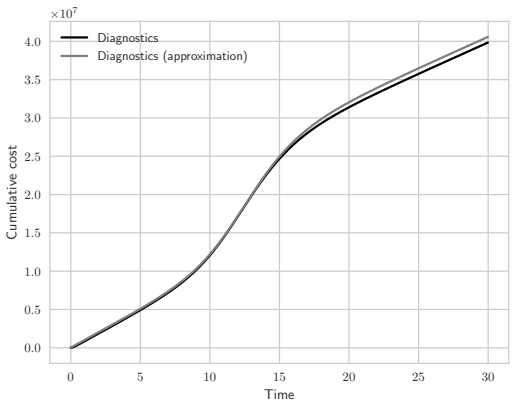
Inevitably, by allowing only one time scale, the approximation cannot represent the complex sub-dynamics of an algorithm. However, for a decision tree embedded in a dynamic model the overall numbers following each route through the tree will be correct, and the mean time spent undergoing diagnosis a close approximation. Figure 3 illustrates the faithfulness of this approximation for our simple two-step algorithm under constant inflow. The approximation is somewhat incorrect during the early dynamics as the internal tree states are populated, but arrives at the correct equilibrium.

Figure 4 shows the dynamics of the system in the full model and with the mean time approximation, and Figure 5 shows the cumulative estimates of infections, treatments and diagnostic tests (as proxies for health outcomes and costs). In this example, the mean time approximation captures all of these quantities with reasonable accuracy.

We expect this approximation will perform well when the diagnostic process is fast compared with the epidemic dynamics. In Figure 4 and Figure 5 this is the case, as the diagnostic time scales were take to be 1/10th that of the mean recovery time, which sets the time scale for the epidemic dynamics. Figure 6 shows the results of our sensitivity analysis, which confirms diagnostic timescales as the main influence on approximation accuracy. (Note the Sobol' total index measures the sensitivity to a parameter including influences through interactions with other parameters, and the fact that the sum of these indices is here greater than one indicates the existence of interactions.) As the diagnostic time scales increase relative to the epidemic time scale, the performance of the approximation for quantities of interest worsens (Figure 7). This, is




**(a)** Infections & treatments




**(b)** Diagnostics performed

**Figure 5.** Cumulative infections and treatments (a) and diagnostics performed (b) in the SIR and treatment model (black) and the mean time approximation (grey). Diagnostic time scales are taken to be 10 times shorter than the recovery time scale, $\nu = 1$ and $\beta = 2$, $s_1 = s_2 = 0.7$, and $sp_1 = sp_2 = 0.9$.

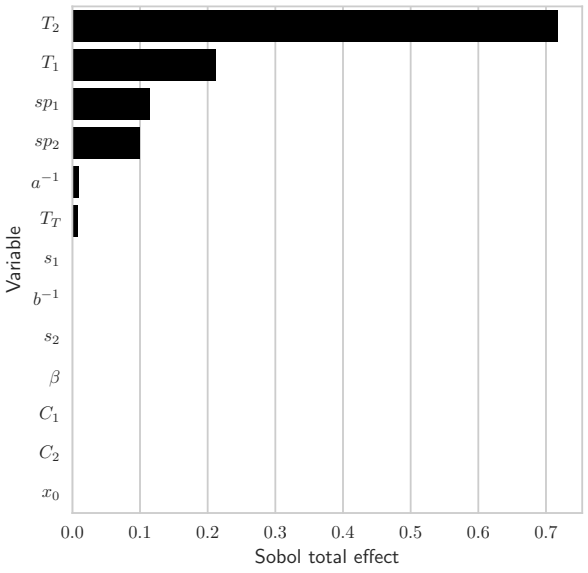



**Figure 6.** Total Sobol' sensitivity indices quantifying parameters' effects on proportional error in treatment prevalence.

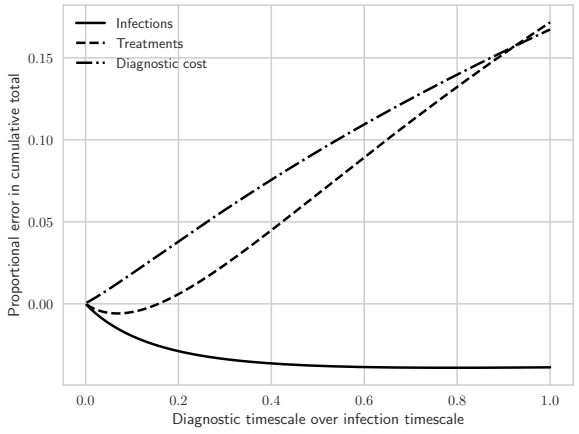



**Figure 7.** Proportional error (1-approximation/truth) for the cumulative treatments, infections and diagnostics after 30 recovery time-scales as the diagnostic time-scales increase relative to the recovery time-scale. Again, $\beta = 2$, $s_1 = s_2 = 0.7$, and $sp_1 = sp_2 = 0.9$.

particularly true for the estimated cumulative number of treatments. However, as long as proportional errors are less than around 5% while the longest delay (i.e. $T_1 + T_2$) is less than 20% of the recovery timescale - in approximate terms, as long as the time from initiation of the diagnostic algorithm to initiation of treatment is less than one-fifth of the total (untreated) infectious duration, the proportional error in estimates of all outcomes is less than 15%, which will generally be adequate for healthcare decision-making.

### *Case study 3*

We used the approach introduced above to facilitate the introduction of user-defined diagnostic algorithms for our web-based interface to the FlexDx model, in response to user demand. In this module, users can name tests and input those tests' cost, specificity and sensitivity for TB in different patient groups (defined by sputum-smear status, HIV-infection status and history of previous treatment for TB), as well as their sensitivity and specificity as a test for multidrug resistance. Once defined, these tests could then be selected alongside additional pre-defined options to specify a novel, user-defined diagnostic tree consisting of a maximum of two test options. The transmission component of the FlexDx model can then take this tree and use it to estimate the comparative cost and epidemiological impact of the user-defined algorithm, as well as additional pre-defined algorithms, in a variety of different settings. This is accomplished by calibrating the transmission model to user-specified epidemiological targets and simulating the consequences of each diagnostic algorithm.

## Discussion

In summary, we have demonstrated how to integrate decision trees with arbitrary height and width into compartmental models for purposes of evaluating the impact of algorithmic diagnostic interventions for infectious diseases. Our technique uses a mean-time approximation that allows the delay associated with an entire decision tree to be encapsulated in a single holding state, such that transmissions can still occur during the diagnostic and treatment initiation process without the need to break this process into a large number of sub-states. The decision tree is used to calculate delays and outcome probabilities, and we have demonstrated how recursive algorithms can greatly simplify this process, adding both transparency and simplicity to model code. Our technique works well in simplified systems where the mean delay associated with diagnosis is short relative to the overall dynamics of disease transmission - which is commonly the case in infectious disease diagnosis - and has also been effectively employed in a more complex model of TB dynamics. While we have demonstrated the use of our technique in evaluating diagnostic algorithms for TB, we anticipate that it will be broadly applicable to similar diagnostic interventions for a variety of infectious diseases and other processes where it may be desirable to incorporate decision trees into compartmental models.

Where applicable, this method makes specifying different algorithms and modifying them much easier. It allows users to focus attention where they want it: in describing the details of diagnostic algorithms for comparison. Algorithms and test characteristics can depend on the patient groups included in the model, and the characteristics of later tests in an algorithm can depend on the results of earlier tests. Code re-use makes model implementation more concise and easy to debug, and potentially more efficient. Use of array structure to implement model layers where

possible in defining dynamics also has these benefits, and can facilitate integrating a decision tree state into a larger model. Changing the algorithms to be compared, and comparing large numbers of algorithms, is greatly facilitated.

In contrast to compartmental (differential equation) models as described here, individual-based transmission models represent an alternative approach to handling complex diagnostic algorithms that does not rely on any mean-time approximation.(Tappenden et al. 2013; Najafzadeh et al. 2012) The primary limitation of individual-based models is their computational expense. This may limit the robustness of parameteric sensitivity analysis important to cost-effectiveness analyses and decision modelling, and is a particular problem in infectious disease modelling where models require calibration to match the unobserved infection process. For example, widely-used Monte Carlo inference approaches often require tens of thousands of likelihood evaluations for moderate dimension parameter spaces, and since individual-based models frequently exhibit substantial stochasticity, tens or hundreds of runs may be required for each parameter set to approximate a single marginal likelihood evaluation. Deterministic compartmental models of infectious disease transmission are computationally less expensive, but need to make averaging approximations. Ultimately, our approach is not meant to replace individual-based models where they are appropriate, but rather to offer an alternative to incorporating complex algorithms into compartmental models when a simpler mechanistic representation of the infectious disease transmission process is desired.

We envisage approaches such ours being of particular usefulness in the area of country-level modelling. Country-level modelling is used to evaluate the epidemiological and cost consequences of national policies, usually with a particular focus on HIV(Stover et al. 2010; Kerr et al. 2015) or TB.(Houben et al. 2016; Trauer et al. 2017) These models are typically used to support applications by low- and middle-income countries for support from the Global Fund to Fight AIDS, Tuberculosis and Malaria, and need to be sufficiently generic to encompass the wide range of interventions different countries may be considering. Modelling work is often carried out by technical assistant partners, or by country users together with technical assistance, and so models must also be friendly enough to be used by non-modellers and without substantial, or any, additional software development. Our approach, and similar techniques, may facilitate incorporation of the implementation details required for costing and specialising models to relevant contexts.

Our use of this technique in the FlexDx model simplified the coding of user-specified diagnostic tests for inclusion in a model of cost and epidemiological impact. In this example, users were restricted to a two-step algorithm, primarily because enabling specification of more complex algorithms requires a more complex user interface, and we did not have the capacity to train users in-person on the utilization of such an interface. However, adopting this approach makes it only a small step to allowing user-defined *topologies* for diagnostic algorithms, e.g. trees with different numbers of levels, linked together in different ways. Thus, although this flexibility was not used in the case of the FlexDx model, the approach described here is inherently generalizable not only to a wide array of diagnostic test specifications, but also to an unlimited number of potential diagnostic algorithm topologies.

We have focused here on the example of decision trees representing diagnostic algorithms in infectious disease models. However, our methods have wider applicability, to any problem that could benefit from embedding a decision tree into a system of differential equations to represent greater detail at a specific stage. Decision trees could, for example, add granularity to Markov models formulated in terms of differential equations governing state probabilities (e.g., if transitions from a given state in the model depended on a detailed series of decision-like steps). Moreover, system dynamic modelling(Radzicki and Taylor 1997) is often used in other contexts where shared resources or other interactions (besides infectious disease transmission) requires modelling of feedback loops and state-dependent event rates. Such models might also utilize decision trees to more precisely represent interactions or feedback loops at specific nodes.

The major limitation of this approach is its poor performance when the delay associated with the decision tree begins to approach the dynamic time scales of disease progression and transmission. In our SIR model, this meant that progression through the diagnostic algorithm should be faster than recovery from infection. Fundamentally, this positions our method as appropriate for 'meso-time-scale' processes. On one hand, if the decision tree delay is very much shorter than other time-scales (such that progression through the tree can be reasonably assumed to be instantaneous), modelling a holding state may not be necessary at all. It should be noted, however, that the recursive methodology and class structures for computing outcome probabilities would still have application in these cases. On the other hand, if the delay associated with progression through the decision tree approaches the overall time scale of disease dynamics, the mean-time approximation breaks down, and more complex approaches will likely be necessary. Importantly, many infectious disease systems do fall into the 'meso-time-scale' category where delays to diagnosis substantially influence transmission, but where they are shorter than the progression time scales. TB is a good example of this - diagnosis may take of the order months, whereas cases remain prevalent on the order of years. Other examples include diagnosis of drug resistance in 'chronic' infectious diseases such as HIV, and non-rapid (e.g., microscopic or culture-based) diagnosis of acute infections (where diagnosis may take hours to days while the generation time is measured in days to weeks). Importantly, we were unable to explore the influence of structural complexity on this approximation; however, given the results of our sensitivity analysis in Figure 6,

we expect that relative time scale will remain the key factor influencing the accuracy of this approximation under other model structures.

## Conclusion

We present a mean time-scale approximation and recursive computational techniques to greatly simplify flexible inclusion of decision trees in dynamic compartmental models of infectious disease diagnosis. These methods may have broader utility in systems where moderately complex algorithmic flows, naturally described by decision trees, need embedding in dynamic systems represented by systems of differential equations.

## References


Matt J Keeling and Pejman Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, September 2011.

Nicolas A. Menzies, Ted Cohen, Hsien-Ho Lin, Megan Murray, and Joshua A. Salomon. Population health impact and cost-effectiveness of tuberculosis diagnosis with Xpert MTB/RIF: a dynamic simulation and economic evaluation. *PLoS Medicine*, 2012.

A. David Paltiel, Milton C. Weinstein, April D. Kimmel, George R. Seage III, Elena Losina, Hong Zhang, Kenneth A. Freedberg, and Rochelle P. Walensky. Expanded screening for HIV in the United StatesâĂŤan analysis of cost-effectiveness. *New England Journal of Medicine*, 352(6):586–595, 2005.

Mark Jit and Marc Brisson. Modelling the epidemiology of infectious diseases for decision analysis. *Pharmacoeconomics*, 29(5):371–386, 2011.

Richard Pitman, David Fisman, Gregory S. Zaric, Maarten Postma, Mirjam Kretzschmar, John Edmunds, and Marc Brisson. Dynamic Transmission Modeling A Report of the ISPOR-SMDM Modeling Good Research Practices Task Force Working Group. *Medical Decision Making*, 32(5):712–721, 2012.

Alan Brennan, Stephen E. Chick, and Ruth Davies. A taxonomy of model structures for economic evaluation of health technologies. *Health economics*, 15(12):1295–1310, 2006.

William O. Kermack and Anderson G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal society of London. Series A*, 115:700–721, 1927.

I. M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Math. Comput. Simul.*, 55(1-3):271–280, February 2001.

J. Herman and W. Usher. SALib: An open-source Python library for sensitivity analysis. *Journal of Open Source Software*, 2(9), 2017.

Python Software Foundation. Python Language Reference, version 2.7., 2017. URL `http://www.python.org`.

David W. Dowdy, Jason R. Andrews, Peter J. Dodd, and Robert H. Gilman. A user-friendly, open-source tool to project impact and cost of diagnostic tests for tuberculosis. *eLife*, page e02565, June 2014. ISSN 2050-084X. doi: 10.7554/eLife.02565.

Paul Tappenden, Jim Chilcott, Alan Brennan, Hazel Squires, Rob Glynne-Jones, and Janine Tappenden. Using whole disease modeling to inform resource allocation decisions: economic evaluation of a clinical guideline for colorectal cancer using a single model. *Value in Health: The Journal of the International Society for Pharmacoeconomics and Outcomes Research*, 16(4):542–553, June 2013. ISSN 1524-4733. doi: 10.1016/j.jval.2013.02.012.

Mehdi Najafzadeh, Carlo A. Marra, Larry D. Lynd, and Sam M. Wiseman. Cost-effectiveness of using a molecular diagnostic test to improve preoperative diagnosis of thyroid cancer. *Value in Health: The Journal of the International Society for Pharmacoeconomics and Outcomes Research*, 15(8):1005–1013, December 2012. ISSN 1524-4733. doi: 10.1016/j.jval.2012.06.017.

J. Stover, P. Johnson, T. Hallett, M. Marston, R. Becquet, and I. M. Timaeus. The Spectrum projection package: improvements in estimating incidence by age and sex, mother-to-child transmission, HIV progression in children and double orphans. *Sexually Transmitted Infections*, 86 Suppl 2:ii16–21, December 2010. ISSN 1472-3263. doi: 10.1136/sti.2010.044222.

Cliff C. Kerr, Robyn M. Stuart, Richard T. Gray, Andrew J. Shattock, Nicole Fraser-Hurt, Clemens Benedikt, Markus Haacker, Maxim Berdnikov, Ahmed Mohamed Mahmood, Seham Abdalla Jaber, Marelize Gorgens, and David P. Wilson. Optima: A Model for HIV Epidemic Analysis, Program Prioritization, and Resource Optimization. *Journal of Acquired Immune Deficiency Syndromes (1999)*, 69(3):365–376, July 2015. ISSN 1944-7884. doi: 10.1097/QAI.0000000000000605.

R. M. G. J. Houben, M. Lalli, T. Sumner, M. Hamilton, D. Pedrazzoli, F. Bonsu, P. Hippner, Y. Pillay, M. Kimerling, S. Ahmedov, C. Pretorius, and R. G. White. TIME Impact âĂŞ a new user-friendly tuberculosis (TB) model to inform TB policy decisions. *BMC Medicine*, 14(1), December 2016. ISSN 1741-7015. doi: 10.1186/s12916-016-0608-4.

James McCracken Trauer, Romain Ragonnet, Tan Nhut Doan, and Emma Sue McBryde. Modular programming for tuberculosis control, the âĂIJAuTuMNâĂİ platform. *BMC Infectious Diseases*, 17(1), December 2017. ISSN 1471-2334. doi: 10.1186/s12879-017-2648-6.

Michael J. Radzicki and Robert A. Taylor. Introduction to system dynamics. *A Systems Approach to Understanding Complex Policy Issues, US Department of Energy's*, 1997.

## Diagnostic class in Python

```
import numpy as np                          #for arrays
from copy import deepcopy as dcpy           #for copying classes

class Diagnostic:
    def __init__(self,sens,spec,cost,delay,ltfu):   #initialize
        self.sens = sens; self.spec = spec;
        self.cost = cost; self.delay = delay;
        self.ltfu = ltfu;      # loss to follow-up
        self.root = True       # by default, this is a root
        self.next = [0,1]      # default as treatments, can be next diagnostics
        self.transition = np.array([[spec,(1-spec)],
                                    [1-sens,sens]])

    def setnext(self,k,newdx):        # next test
        self.next[k] = dcpy(newdx)     # add to tree, as copy
        self.next[k].root = False      # record that this is no longer a root

    def getTables(self):                    # get matrices by recursion
        txo = np.zeros((2,2)); cost = np.zeros((2,2)); delay = np.zeros((2,2))
        if self.root:
            cost += self.cost*self.transition # add on own costs if root
        for k in [0,1]:
            if isinstance(self.next[k],Diagnostic):
                txon, costn, delayn = self.next[k].getTables()
                for j in [0,1]:
                    nextbit = self.transition[:,k] * txon[:,j]
                    txo[:,j] += (1-self.ltfu) * nextbit
                    cost[:,j] += self.next[k].cost * (1-self.ltfu) * nextbit
                    delay[:,j] += self.delay * (1-self.ltfu) * nextbit
            elif self.next[k] == k:
                txo[:,k] += (1-self.ltfu) * self.transition[:,k]
                cost[:,k] += 0
                delay[:,k] += self.delay * (1-self.ltfu) * self.transition[:,k]
        return txo, cost, delay
```