# Readme: Guide to reading the code

The Matlab code uses linear algebra (i.e. a matrix-based approach) for constructing and implementing mathematical models. This approach has several different advantages: it is convenient for looping over stratifications (e.g. HIV status); it takes advantage of the computing efficiencies in programming languages that are optimised for matrix operations (e.g. Matlab or Python); and it allows easy model specification, especially in models having many compartments. The following notes outline some of the key features of the approach, in order to aid understanding of the code.

Main scripts

*Setup\_model*:

Sets up all model parameters and reference lists, as well as data to be used for the calibrations. Model parameters are encoded as structures (equivalent to ‘lists’ in R or ‘dictionaries’ in Python), such that the parameter contains all per-capita rates of transition, and the parameter contains all proportions.

‘Compartment lookups’ include *s* and *i*. The lookup is a Matlab structure recording the compartment number of each state in the model. For example, writing i.U.h1 gives a value of 2, indicating that people who are uninfected with TB, but have HIV, are represented by the 2nd compartment in the model. This approach is very helpful in keeping track of compartment numbers when constructing model equations (as conducted in ‘Make\_model’, below).

The lookup is a Matlab structure recording all states belonging to certain sets. For example, s.I gives [10 11 12], the compartment numbers of all states that correspond to active TB. It is straightforward to find overlaps between sets: for example, typing ‘intersect(s.infectious, s.h0)’ gives all compartment numbers of states that are HIV-negative, and that are infectious for TB.

*Make\_model*:

Given all model parameters and reference lists, constructs matrices to encode the model. This code outputs different matrices under the Matlab structure M, as follows:

M.lin is a matrix capturing all linear transitions (that is, those where per-capita rates of transition are not state-dependent). Generally, these include all transitions apart from the force-of-infection. The construction of M.lin loops over the different strata in the model, relating to HIV status and public/private sectors. For each transition, originating compartments are referred to as ‘source’ while receiving compartments are referred to as ‘destin’ (for ‘destination’).

The process of infection, being the only nonlinearity in the model, requires two matrices to be fully specified: one to calculate the force-of-infection (M.lam) and the other to identify the source and destination compartments for all transitions arising from infection (M.nlin): for example, from uninfected to latent ‘fast’ status. Finally, the matrix M.lambda calculates the force-of-infection by combining all infectious compartments with their infectivity, and summing.

*get\_objective2D*:

For a given parameter set , simulate the TB epidemic until 2019, and compare with data to provide the log-posterior density (‘out’). Secondary outputs (‘aux’) list some key results of the model simulation.

*Get\_calibrations*:

Performs the MCMC calibrations, using *get\_objective2D* to evaluate posterior densities. This code first draws 10^4 parameter sets, and then chooses the three best-performing parameter sets as starting points for MCMC. The actual MCMC algorithm is conducted using adaptive Bayesian MCMC.

*Disruptionvector\_get\_initial:*

Using the posterior sample from *Get\_calibrations*, simulate notifications during the period of COVID-related disruption, and compare with data. This is the code used to identify the timeseries for , which captures COVID-related disruptions to TB services.

All scripts with 'Control' in the title execute codes for a batch of countries, creating outputs that can be read into Excel files for sharing.

All scripts with 'Collate' in the title construct final tables of outputs for a batch of countries.