TreeAgePro model calibration with Scientific Python.

Why is model calibration required?

Clinical data often is limited to patient counts (proportions) by state at 2 or more points of time, such as at the beginning of the study and then at the end of 1 year, 2 years, etc.

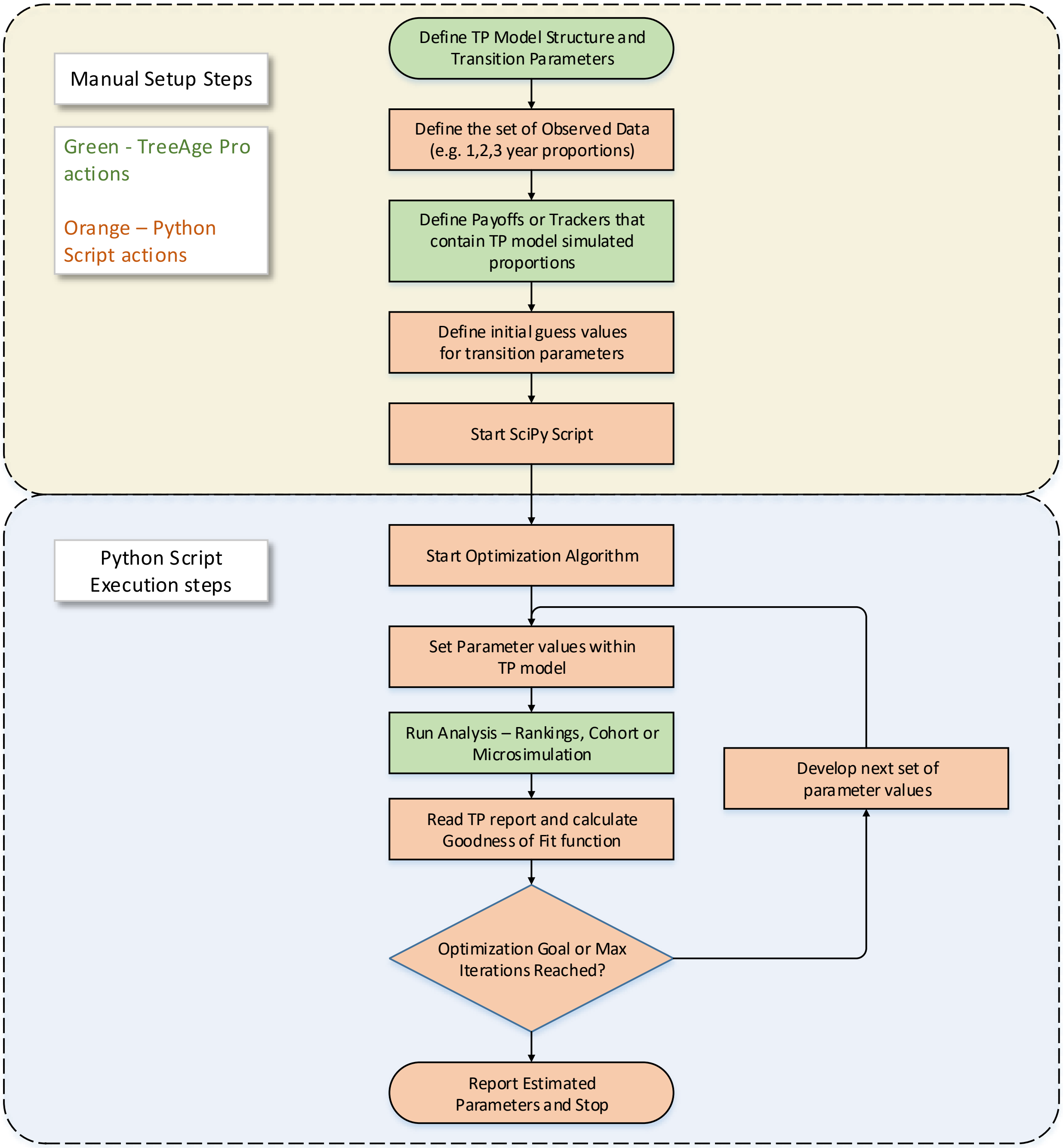
When more than 2 states are involved there is no easy way to estimate the individual state to state transitions (either exponential, Weibull, etc.), since a longitudinal state to state transition times by individual would be required. Often this level of data is not available, the only option is to assume that the transitions between states follow a particular distribution and then adjust the distribution parameters to try to generate simulated state proportions at desired times to match as closely as possible to the observed proportions.

Scientific Python library includes about 10 different optimization algorithms that can be used to perform model calibration by searching for distribution parameters that minimize the error between the simulated proportions and observed proportions.

Overview of the Calibration process

The following diagram presents the flow of the calibration process. TreeAge Pro model has to be configured with appropriate payoffs and/or trackers which can be evaluated by the Python script.

The top part of the flow chart identifies the manual steps that need to be performed with TreeAge Pro model (Green color steps) and Python script (Orange color steps). One the Optimization script is launched in the bottom part of the flow chart, the execution of steps is performed by TreeAge Pro and Python functions indicated by the corresponding colors.


Manual Setup Steps
Green - TreeAge Pro actions
Orange – Python Script actions
Define TP Model Structure and Transition Parameters
Define the set of Observed Data (e.g. 1,2,3 year proportions)
Define Payoffs or Trackers that contain TP model simulated proportions
Define initial guess values for transition parameters
Start SciPy Script
Python Script Execution steps
Start Optimization Algorithm
Set Parameter values within TP model
Run Analysis – Rankings, Cohort or Microsimulation
Read TP report and calculate Goodness of Fit function
Optimization Goal or Max Iterations Reached?
Develop next set of parameter values
Report Estimated Parameters and Stop

How good are the optimization algorithms?

There are many optimization algorithms and it is important to understand their strengths and weaknesses. All optimization algorithms suffer to a lesser or greater degree from finding a local minimum, rather than global minimum. Existence of local minima is a function of the model structure and distributions (hazard functions) used for state transition.

It is informative to create a reference model that uses particular distribution parameters and generates the cohort proportions that can be used as the observed set. This set of cohort proportions at different times is then used as the input to the optimization problem.

Different initial parameter values can be used to assess how good the optimization algorithm is at finding the reference parameters. The combination of starting initial parameter and different algorithms and their options are endless, so it is important to have a good test environment for testing optimizer performance.

Goodness of Fit functions:

There can be many different formulations of fitness functions. Python optimization algorithms rely on user defined function that will be minimized. In the case of model calibration our objective is to find transition parameters which minimize the difference between observed (given) state proportions at time periods and the proportions generated by the model with estimated parameters, starting with the initial parameter guesses.

A simple goodness of fit function is a sum of squares of differences between observed and simulated cohort proportions:

$$GoF = \sum_{i}^{n} \sum_{k}^{m} (\hat{p}_{i,k} - p_{i,k})^2$$

Where,

$\hat{p}_{i,k}$ – is the observed (given) proportion of the cohort at time *i* and at state *k*

$p_{i,k}$ – is the simulated proportion of the cohort at time *i* and at state *k*

Other goodness of fit functions could be formulated using root mean square of error or maximum-likelihood function. Selection of goodness of fit function, will also have an impact on the performance of the particular optimization algorithm.

Model calibration science or art?

Certainly there is quite a bit of experimentation involved with choosing appropriate optimization algorithm, appropriate choice of initial parameter estimates and goodness of fit objective function. It is not possible to provide an answer to what are the best choices for a given model calibration task.

However, some general observations may be helpful in preparing a model for calibration.

1. When possible use expected value calculations to establish cohort proportions rather than microsimulation. Microsimulation generate “noisy” data which works against the optimization algorithm which will likely converge on a local minimum. Rankings and Cohort analyses use deterministic cohort calculations reducing the estimate noise.
2. Using seeding with microsimulation might help reduce the level of noise. (To be tested experimentally).
3. Build a simpler cohort model and try to calibrate it first establishing a better set of parameters that could be used in the microsimulation version of the model.
4. Build a model with assumed parameters, generate observed results and then try different calibration algorithms and initial guess to see how well different algorithms can “recover” the correct parameters. (On-going experiment).