The Reference Model for Disease Progression Combines Disease Models

Jacob Barhak

Austin, TX
jacob.barhak@gmail.com

ABSTRACT

Simulations based on disease models can be used as a training tool for patient education or caregiver support to improve effectiveness of practice. Disease models provide predictions of patient outcomes, costs, and quality of life information over time. They are typically constructed by various teams around the world based on local data, and currently those do not validate well against multiple external populations. Therefore, universal understanding of disease progression is still unsolved.

The Reference Model for Disease Progression addresses this problem by implementing a league of disease models that compete for fitness towards publicly available clinical data. Currently, diabetic populations are of interest and data for the model is drawn from published clinical trials, while model building blocks are based on published risk equations and modeling hypotheses. The Reference Model creates model combinations from those building blocks, then simulates and validates them against multiple populations. High Performance Computing (HPC) techniques are used to cope with the combinatorial number of models that, until recently, took roughly a year of computation time on a single processor.

Recently new building blocks have been added to the model, and the number of combinations became too large to compute in reasonable time without access to a large cluster. To cope with this, the structure of the model was changed from a competitive discrete ensemble model, where building blocks are selected from a discrete pool of options to construct the best model, to a cooperative continuous ensemble model, where all building blocks are merged using linear combination. Moving the model toward continuous combination space allowed creation of an assumption engine that employs optimization algorithms to better deduce the most fitting model. This significantly reduces simulation time and produces a better fitting model. This paper will focus on recent modifications, and results obtained from the latest simulations.

ABOUT THE AUTHORS

Jacob Barhak specializes in chronic disease modeling with emphasis on using computational technological solutions. Dr. Barhak has diverse international background in engineering and computing science. He and received his Ph.D. in 2003 from the Technion Israel Institute of Technology. From 2003 to 2006 he worked at an NSF Engineering Research Center at the University of Michigan focusing on inspection systems and personalization. From 2006 to 2012 he worked on developing disease modeling tools for the Michigan model for diabetes at the University of Michigan. The Reference Model for disease progression was independently self-developed by Dr. Barhak in 2012. He is the developer of the Micro Simulation Tool (MIST).

The Reference Model for Disease Progression Combines Disease Models

Jacob Barhak

Austin, TX
jacob.barhak@gmail.com

INTRODUCTION

Disease models typically describe the progression of disease and its impact on a population. Disease models were typically used for economic evaluation (Lasry & Zaric, 2007), (Harvey, 2016) or for evaluation of impact of a disease (Salem & Smith?, 2016), (Asche & Kim, 2016). Much work has been done to study the outcomes of a disease to better understand factors driving the disease (Hayes & Leal 2013), (Clarke & Gray 2004). Those models typically have long term predictions counted in years ahead. Yet recent advances started the process of incorporating simple decision support models into Electronic Medical Record (EMR) systems (Sperl-Hillen & Averbeck, 2010), to make them more accessible to practitioners.

Practitioners will therefore be increasingly exposed to computational medical models in the foreseeable future. Furthermore, new virtual simulators that help medical training are becoming much more realistic (Body Interact, n.d.). Such simulations track many biomarkers and conditions that change during simulations. Although such simulations are often driven by preset scenarios, there is a recognized need for high fidelity models that properly predict changes in patient condition in shorter time spans. Yet a helpful training tool will allow long term scenarios where a doctor sees the progression of a disease in a patient in the very long term and can study the effect of their treatment in virtual environment. Such a training tool requires good longer term models to break away from preset scenarios.

The appearance of wearable medical devices, that will monitor our biomarkers such as heart rate monitors, thermometers, cholesterol meters, blood glucose meters, and blood pressure monitors, will increase the potential of predictive medical models. Medical models have the potential of being incorporated for analysis of abundant information that will be generated during constant monitoring of wearer condition. These are still in their infancy, yet technologies are being developed for monitoring (Santamaria & Serianni, 2016) and analysis towards use in clinical trials (Gore & Chandra, 2016). These simulation technologies are already slowly becoming available, yet do not broadly integrate predictive disease models, partially since those are being developed slowly. In summary, predictive disease models are already being used to study long term impact on a population, are starting to be used by practitioners, have future potential in medical training, and they will increase their potential in the near future in conjunction with medical devices.

However, there is still a large gap between desired modeling capabilities needed to support predictive medical modeling and our understanding of diseases. This gap is evident when disease modelers compare their models as done in the Mount Hood challenge (The Mount Hood Modeling Group, 2007), (Palmer, 2013) where diabetes models are compared and contrasted, typically every other year. A typical challenge provides a base population as input for modeling teams that then have to predict disease outcomes after several years of disease progression. The predicated outcomes are then compared to observed outcomes that actually took place for the modeled population. The challenge consists of comparing outcomes amongst modeling teams. Previous challenges showed than different modeling teams provide different models that led to different predictions, sometimes with considerable gaps. Since modeling teams are well trained and work hard for the challenges, then overlooking human error, it is fair to assume that different structures, data sources, and assumptions result in different predictions for similar base populations.

Those different results are reasonable considering that there are many differences between models and that new risk equations are being added constantly to the pool of available equations. For example, the United Kingdom Prospective Diabetes Study (UKPDS) (Stevens & Kothari, 2001) and QRisk (Hippisley-Cox & Coupland, 2008) modeling teams provided different models throughout the years. Considering that those equations faithfully represent phenomena observed in different populations, it is reasonable to assume those are good models.

Nevertheless, due to different structure, those will produce different results. Moreover, those will behave differently for different base populations, i.e., different initial conditions.

The Reference Model for Disease Progression (Barhak, 2014), (Barhak, 2015a), (Barhak, 2015b), (Barhak & Garrett, 2016) was built to try and explain those differences. It is composed of multiple competing risk equations and assumptions and populations. It is an ensemble model meaning that it is composed from multiple models. Other ensemble models examples include (Bell & Bennett, 2009) where viewer movie preference was predicted, and (KLusek & Dzwinel, 2016) where multiple models were merged to create a better cancer model. The Reference Model basic function was to test which equations work better for what populations, and what model constructs better explain observed data from multiple clinical studies for a given query. The Micro Simulation Tool (MIST) (Barhak, 2013) is the simulation framework upon which the model is executed. It is free software that provides object oriented population generation, Monte-Carlo micro-simulation, and reporting tools. Moreover it provides High Performance Computing (HPC) capabilities that allow coping with the large number of model/assumption/population combinations (Barhak, 2015b).

The Reference Model grew since 2012 adding more models and populations each year. In 2016 the number of combinations became unreasonable to compute all combinations, and a computational wall was reached. Therefore the model structure was changed. Instead of using model-combination competition, it was decided to allow model cooperation. So instead of calculating all combination of models/assumptions in discrete combination space, models would be allowed to cooperate by linearly combining them and creating continuous model space. This change allowed for the breaking of the computational wall and adding more equation/assumptions to the models. This paper will describe the implementation of the new technique and will explore the new results and benefits associated with this change. Let us start with discussing model structure.

THE REFERENCE MODEL STRUCTURE CHANGE

Original Structure Supporting Competing Model Combinations

The Reference Model is currently focused on diabetic populations and is composed of three main disease processes, heart disease, stroke, and competing mortality as shown in Figure 1. The Reference Model currently models and validates against 9 diabetic populations ASPEN (Atorvastatin Study for Prevention of Coronary Heart Disease Endpoints in Non-Insulin-Dependent Diabetes Mellitus), ADVANCE (Action in Diabetes and Vascular Disease: Preterax and Diamicron Modified Release Controlled Evaluation), ACCORD (Action to Control Cardiovascular Risk in Diabetes), UKPDS (United Kingdom Prospective Diabetes Study), KP (Kaiser Permanente), NDR (Swedish National Diabetes Register), Look AHEAD (Action for Health in Diabetes), ADDITION (Anglo-Danish-Dutch Study of Intensive Treatment In PeOple With screeN Detected Diabetes in Primary Care), and CARDS (Collaborative Atorvastatin Diabetes Study). Most of these populations describe published clinical trials, while NDR and KP summary data was provided by owners of data for the Mount Hood diabetes challenge participants in 2012. The Reference Model was compared and contrasted against multiple other models in two challenges in 2012 and 2014. The Reference Model uses publicly available summary data for those populations as a source of inputs and published outcomes as validation information. During each simulation year each virtual individual generated from those populations goes through a set of rules that can change their biomarkers and other parameters, then transition probabilities are set for all transitions and Monte-Carlo simulation determines for each individual if they progress to the different stages in each process. Finally outcomes and costs are determined for each person during simulation-year post processing. The process is repeated for a cohort of individuals for several years and may be repeated more times to reduce Monte-Carlo error. At the end of the process simulation results are compared to known outcomes of that cohort and a fitness function is employed to determine the fitness score. This fitness score is essentially the distance/error of simulated results from observed results in real life.

The Reference Model structure is shown in Figure 1 with some simplification for illustration purposes. Generally, each transition in the model holds at least one possible equation for the yearly transition probability. However, some transitions have more than one equation, e.g., the probability of Myocardial Infarction (MI) can be computed by equations denoted A, B, C, D and the probability of stroke can be computed using equations E, F, G, H. Therefore, there are 4x4=16 competing models combinations in this simple example to choose from when executing simulations: $AE, AF, AG, AH, BE, BF, BG, BH, CE, CF, CG, CH, DE, DF, DG, DH$. Each one of these model combinations

is different and while looking for the best model that fits multiple populations, every model should be executed for each of the populations. Availability of computing power and parallelization of execution using HPC techniques makes this possible in reasonable time. Running all models would generate a fitness matrix that is similar to a score board that shows how models behave for all populations. This can be color coded and analyzed to deduce the best model for certain populations, or the best model on average.

Since the number of combinations is a multiplication of the number of options for each component, the number of model combinations increases combinatorially, which is close to exponential growth. Notice that the example in Figure 1 can easily grow out of reasonable proportions very quickly if more equations or assumptions are added without excluding any combinations from consideration. Consider that multiple equations are used to determine multiple transitions, and that more simulations are required for additional assumptions that increase the number of simulations.

Table 1 shows both the equation options available in 2015 and those that were planned for 2016. The table lists all options that can be taken per category. The row titled: "Possible Model/Assumption Combinations" multiplies all above rows to calculate the number of model combinations, the number is multiplied by number of population cohorts to figure out the number of simulations needed to build the full fitness matrix. This number of simulations, after some model exclusions, required roughly 10-11 days in 2015 to calculate using a 16 core cluster – the equivalent of roughly half a year of computing time. Therefore the 8 fold increase in number of simulations projected for 2016 would require roughly 3 months of computation on the same cluster, equivalent to about 4 years of computing time. This increase seemed unreasonable. Moreover, future model additions would increase simulation time even further. Therefore, it was decided to change the structure of the model from competitive model combinations in discrete model space to cooperating models in continuous model space.

Table 1. Model components in 2015 and 2016. *=strictly competing components

Number of Options for:	**Year**	
	2015	**2016**
Population Correlation Assumption *	2	2
Biomarker Change Assumption*	2	2
Model Temporal Correction*	2	2
MI	10	11
MI Death	2	3
Stroke	10	12
Stroke Death	1	2
Competing Mortality	1	2
Possible Model/Assumption Combinations	1,600	12,672
Population Cohorts	47	47
Possible distinct Simulations	75,200	595,584

New Structure Supporting Cooperating Model Combinations

Figure 1 illustrates this change in modeling paradigm where the bottom describes the cooperative model combination. Instead of 4 equations for MI and stroke probabilities, there is a single equation that combines those probabilities using a linear combination with four coefficients, a, b, c, d for MI and e, f, g, h for stroke. Note that now each time a model is executed, all equations potentially contribute to the simulation results with different weights whereas with competitive combinations, only one equation for each category would contribute for each scenario. For example, it is possible to have the average of equations $A\&B$ and the average of equations $E\&F$ contribute to the model equally by setting $a = 0.5, b = 0.5, c = 0, d = 0$ and $e = 0.5, f = 0.5, g = 0, h = 0$. This combination of models can be represented as a vector in 8D space so $\mathbf{z} = [a, b, c, d, e, f, g, h]$ describes the model combination. Note that the discrete competition space is fully contained within the continuous cooperative model space, i.e., the competitive discrete model combination AE can be written in cooperative model space as $\mathbf{z} = [1,0,0,0,1,0,0,0]$.

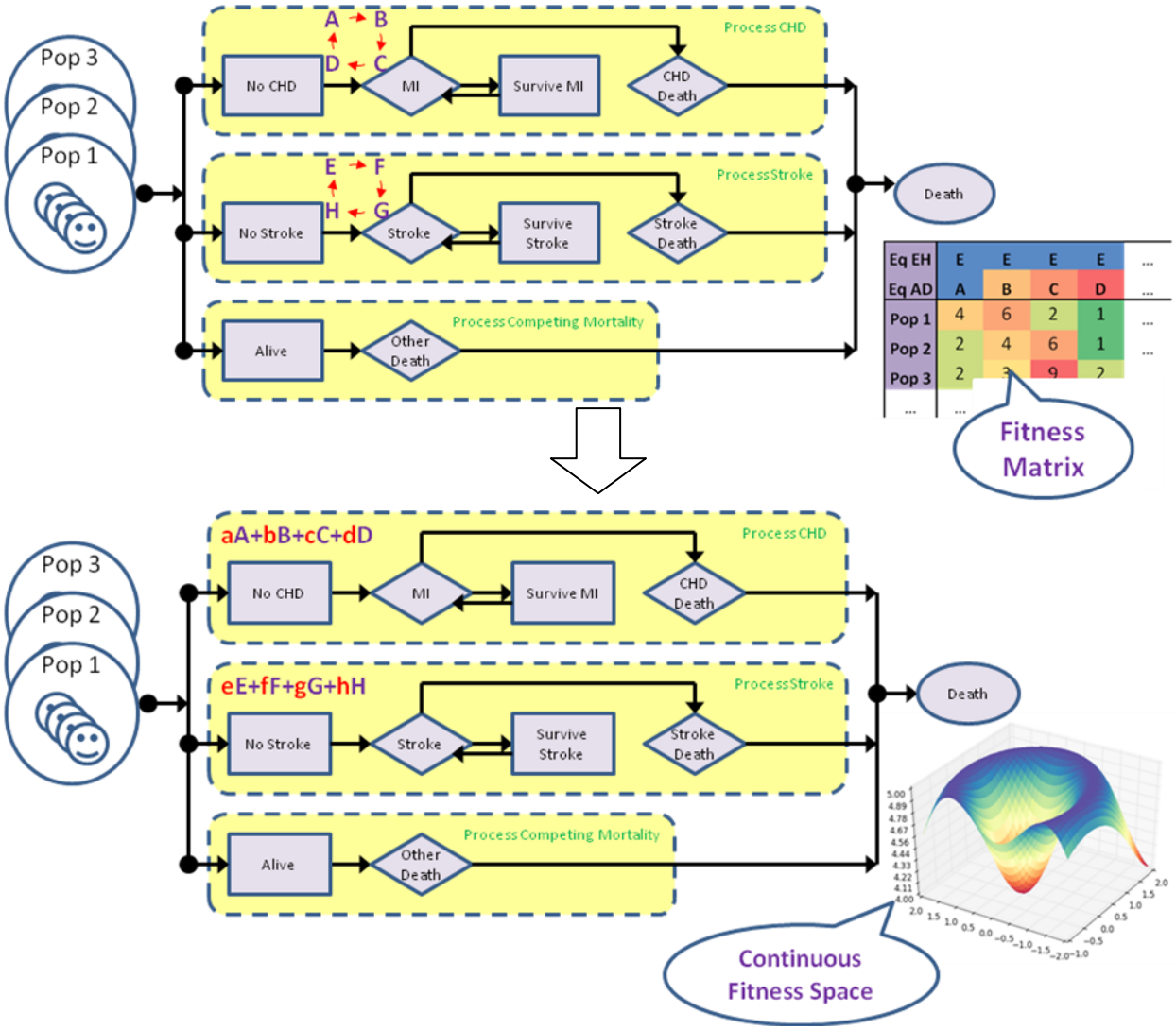

Figure 1. The Reference Model transformed from model competition (Top) to model cooperation (Bottom).

Due to containment this reconfiguration of model space is superior. However, the number of models is now infinite, since our space is now continuous in 8D. So computing the fitness matrix is no longer practical, since traversing the entire continuous model space, as previously done in the discrete competitive model space, will require infinite time. Since parameter space is now continuous it is possible to: 1) Locate better models since continuous space is higher resolution than discrete model space –higher resolution allows locating better models in between discrete model combinations. 2) Employ optimization algorithms to find best fitness – the model function is continuous considering the coefficients constructing the linear combination.

The idea now is that instead of traversing the discrete models space to build the fitness matrix to locate the best model, the system is asked to find the best fitting model. The system employs an iterative Gradient Descent (GD) algorithm to locate the best fitting model. The GD algorithm starts with a guess, then each GD iteration calculates the direction to step towards that will best improve the fitness score, then GD takes a small step in that direction and repeats until convergence. This process can take much less time than computing all model combinations to converge and can produce better models. The reason is that the complexity is no longer determined by number of combinations, instead it is determined by the number of coefficients and the step size of the GD algorithm and its convergence. Considering each step will require calculating the fitness function 8+1 times in order to calculate the gradient, 9 simulations will be conducted for each GD iteration. This is not significant when 4x4 combinations will

map the entire space. Yet looking at the amount of combinations in Table 1 considering non-strictly competing component, there are roughly 30 possible dimensions for cooperating model combination 2016. It is clear that 30+1 calculations per iteration are fewer than the thousands of calculations needed to calculate the fitness matrix in discrete space, even when repeating iterations for the GD algorithm to converge. In other words complexity of one iteration with the GD algorithm grows linearly with the addition of each equation to the model, while calculating the entire discrete space can grow combinatorially with each new equation.

HPC is still heavily employed during computation because all gradient components are calculated in parallel. Also, strictly competitive model options that have not yet converted to cooperation mode still required consideration. All those options are executed in parallel in order to figure out the best fitness there. Moreover, the parallelism can be employed to get several good model combinations in an attempt to find a global minimum rather than a local minimum. This can be accomplished by starting the GD algorithm with different initial guesses. Therefore, both model cooperation and model competition are employed by merging equations and by using multiple initial guesses and using strictly competing model components that cannot be merged. The number of parallel processes of each GD is the multiplication of 1) The number of initial guesses to starting with, 2) The of number of options for each competing component, typically the a Cartesian product that can be manipulated by the user to exclude unwanted combinations, and 3) Number of cooperating components + 1 for calculating the derivation in each dimension.

The latter set of parallel computations is required to calculate the gradient using a forward finite difference scheme. In each such parallel simulation the fitness score $s(\boldsymbol{z}_k)$ is computed, where $\boldsymbol{z}_k = \boldsymbol{z} + q_k\boldsymbol{\delta}_k$ and $\boldsymbol{\delta}_k$ is Kronecker delta in dimension k and q_k is a small number used to calculate the derivative using forward difference approximation for cooperating equation k. So the gradient vector $\nabla s(\boldsymbol{z})$ components are approximated by $(s(\boldsymbol{z}_k) - s(\boldsymbol{z}))/q_k$ and $s(\boldsymbol{z})$ and $s(\boldsymbol{z}_k)$ can be all computed in parallel. Each such computation requires full Monte-Carlo simulation of the model for a population, comparison of simulation for observed outcomes for that population, and applying the fitness function. Note that this simulation contains Monte-Carlo error, therefore the derivative is not accurate. Longer simulations and fine tuned step size q_k may improve the accuracy. Also note that multiple iterations of the GD algorithm will improve the results on average, so more computing power will eventually improve results, even with the presence of noise as demonstrated in (Barhak and Garrett, 2016) so HPC is an essential component.

To properly use HPC, the modeler has do determine what components cooperate and what components compete. This is done by separating components into groups upon definition. For example considering Figure 1 the modeler will separate the vector **z** into groups $\{a, b, c, d\}$ and $\{e, f, g, h\}$ for such group of equations, the modeler can also define additional behavior methods. Notable behavior methods are: 1) Static, 2) Scaled components. Static components do not participate in the GD optimization. Instead they can be initialized as competing components to provide multiple initial conditions for all of the cooperating components in order to improve fitness during each GD iteration. Scaled components are given a certain scaling value that their sum is scaled to match after each iteration. This is an important element in simulation since the assumption is that each equation faithfully represents the phenomenon that it observed, so their combination should not change the probability by using a linear combination that will cause scaling. For example, under the assumption that each equation $\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}$ represents the probability of heart disease as calculated in a clinical study, then if $a = 0.5, b = 0, c = 0, d = 0$ is chosen, the MI probability will be reduced by half and therefore will no longer be representing the same phenomenon. To honor our assumption that all those equations represent the same phenomenon, a constrain such as $a + b + c + d = 1$ should be added to the parameters. The Reference Model optimization algorithm accomplishes this by defining a scaling value to which each group of cooperating scaled components is scaled to. To supplement this, the optimization algorithm also allows defining bounds for parameter values, so the modeler can define $a, b, c, d \in [0,1]$ thus not allowing probability subtraction and magnification beyond original purpose. This narrows down the search space during optimization to find reasonable values that honor the equation used. The belief is that all risk equations used are reasonable and describe the phenomenon. The goal is to identify the best combination of these equations that will provide better fitness towards a certain fitness function defined by the modeler. This keeps continuous model parameter space within a convex hull where the extreme vertices define the discrete values previously used in the fitness matrix for model combination competition.

The new structure that combines 1) model competition 2) model cooperation through optimization towards a fitness function 3) parallel computing and use of HPC is called an "assumption engine". An assumption engine allows the modeler to "throw" many assumptions at it and the engine will try to figure out the best model that fits a validation

data set. In this case, the assumption engine will find the model/assumption combination that validates best against clinical study outcomes as reported in the literature. Bad assumptions such as a bad cooperating equation will be rejected by the engine by zeroing the coefficient associated with that equation. This indicates to the modeler that this equation does not work well with other equations. Note that a different combination of equations or different initial conditions may cause the equation to be more useful. This allows the modeler to explore model and assumption space while validating against known phenomena. Recall that a model is an assumption of the modeler on how phenomena observed in reality behave. Now computing power can replace some of the effort that was previously placed on the modeler for constructing a good model. The modeler's role is now converted to figuring out building blocks to place in the model and assumptions to combine them with. The assumption engine is used as a powerful tool to complete the job for the modeler. The next section will demonstrate results of employing this newly developed assumption engine.

RESULTS

The reason for the transformation to cooperative model combination was due to addition of new equations as reported in Table 1 at the 2016 column. Since calculating the best competitive model required an unreasonable amount of computing resources, the initial guess chosen was that all cooperating equations had equal contributions. Such an initial guess located in the center of the convex hull of the continuous model space allows the optimization algorithm to head in the direction of the most beneficial equation locally and to continue on this path. The idea is that equations with more beneficial effects will eventually remain while equations that perform poorly with regards to improving fitness will be eliminated. Multiple competing components can be explored and optimized in parallel. With those ideas in mind a set of new simulations followed and analyzed. Those simulations used the same model, fitness function, base populations definitions and query. The differences between simulation sets are explained hereafter per simulation set.

Simulation 1: Partial Cooperation

In this simulation set there was one active competing component from Table 1: biomarker change during simulation. Populations were generated using two population correlation assumptions and the fitness score was averaged. In this simulation the assumption is that the risk equations provided by modelers properly describe the phenomenon observed and do not need temporal correction for time passing between model inception timestamp and data timestamp. All other modeling components in Table 1 were considered cooperative, while all death probabilities were set to static, i.e., they were not changed during optimization, e.g., the probability for stroke death was the average of two equations and the weight of those equations did not change during optimization. So in this simulation the optimization algorithm was only changing the weights amongst the 11 MI probability equations and the 12 stroke probability equations, i.e., there were 23 dimensions during optimization. So during each iteration of the optimization algorithm there were 96=(23+1)x2x2 parallel simulations accounting for gradient calculation and the 4 competing initial guesses. Optimization was executed for 10 iterations. Those simulations took 18 days on a 16 core cluster which is corresponds roughly to a month of computing time per iteration.

Results from the four competing model components are shown in the left most result columns of Table 2. The first two rows in the table show the fitness after the optimization process compared to the initial guess fitness score. This clearly shows improvement since a lower fitness score = better fitting model after optimization. The four columns represent the four competing model components simulated in parallel. Double border lines separate the risk equations to contain cooperating model components in Table 2. Recall that those coefficients should sum to 1 for each such cooperating model component.

There is certainly variability in results between the two biomarker scenarios, which makes sense because some equations will behave differently and there are some equations in the mix that are very similar to each other. However, it is clear that some equations are more useful than others. For example MI equation #5 and #8 seem to be dominant in both scenarios while MI equations #1, #4, and #11 seem to be rejected by the assumption engine. This does not mean that those equations are useless; instead this suggests that those equations do not cooperate well with other model components considering the data within the system. In fact, some of the equations that did not work used to be in good positions in past simulations in 2015, when solely competing models were used. Yet recall that new death equations were added in 2016, and in this simulation those equations were static and were averaged so other equations now operate in different environment and therefore may be less efficient than in the past. Also, one

must consider that the simulation was stopped after 10 iterations and started from a certain initial guess in the center of the convex hull of model space. The fitness function is very complicated and may have been attracted to a local minimum away from other possible local minima. Finally, one should also consider the Monte-Carlo noise that may redirect solution and this is the main purpose of the next simulation set.

Simulation 2: Partial Cooperation Repeated

This simulation was totally equivalent to the previous simulation in all parameters. Furthermore the same exact populations were reused and not regenerated. So all disease simulations started from the same initial conditions. The only thing that changed during this simulation was the random numbers generated through simulation. So this simulation set serves as sensitivity analysis to Monte-Carlo noise. The results can be seen in Table 2 on the column dedicated to the second simulation. Those results should be compared to the previous results on the left side. There are some variations, yet once can clearly see dominance of MI equations #5 and #8 whereas MI equation #1,#7,#9, and #11 are rejected. So there is a nice overlap in some phenomenon yet it is evident that the gaps are wide and more computing power is required to reduce the Monte-Carlo noise. Recall that theoretically, the noise should drop by the square root of the amount of computing power invested. So to gain one digit of accuracy there is a need to spend 100 times more computing power, i.e., each iteration of the algorithm should take 100 months to calculate instead of a month. Although such computing power is available these days and not expensive, and there may be other ways to reduce computation burden, it was decided to continue simulations with current amount of computing power and explore full cooperation amongst modeling components.

Simulation 3: Full Cooperation

This simulation releases the static constrained over the death equations that were new to the system. This is a larger model space that translated to 124 parallel simulations = 30 [perturbed parameters] + 1 [unperturbed simulation] * 2 [population correction assumptions] * 2 [biomarker change assumptions]. Executing 10 optimization iterations took roughly 3 weeks on the 16 core cluster which is close to a year of computation time on one core. Again, in this simulation populations were not regenerated so initial conditions are exactly the same as previous simulations.

The results are organized in Table 2 in the column dedicated to the third simulation. The results show interesting behavior where not correcting for biomarkers provided slightly better results in one simulation. Considering Monte-Carlo noise effects this result might not be significant. Although death equations do change their influential weight on the final model, those changes are small and none of them are rejected. On the other hand, several MI equations #4, #9, and #11 are rejected while MI equations #5 and #6 are dominant. So there seem to be behaviors that repeat in most simulations. Note that all simulations so far were based on the same generated populations, this may be the reason for the similarities. The next set of simulations attempts to address this issue.

Simulation 4: Full Cooperation with Population Generation Repetition

In this simulation set simulations started with different initial conditions. The same populations generated in previous simulations used. However, populations were regenerated using the same definitions and the optimization algorithm was executed for 10 iterations where each iteration averages the results from simulations based on the both generated populations. This should also theoretically affect the mathematical solution that the algorithm is drawn towards if there are significant changes in the populations generated. Note that this should theoretically also reduce the Monte-Carlo noise by roughly 30% since the same simulation is executed twice. Otherwise the simulation is similar to the previous simulation and should be compared to it. Since more repetitions occurred in the last simulation, this simulation should be trusted more. Yet, there is room for technical error. This simulation was large enough to see some artifacts of dropped jobs that are typically seen in HPC environments when software/hardware fails. In all previous simulations such artifacts were removed by forcing recalculation of dropped jobs. In this simulation those dropped jobs were ignored since there was a backup calculation using another population set. The effect of a missing calculation from so many simulations is insignificant and is very rare. Since those artifacts occurred in early iterations, the optimization algorithm corrected for it in later iterations.
Results are shown in Table 4 on the right side. The results reject MI equations #1,#2, and #11 while MI equations #5, #6, #8 are dominant. This verifies results previously obtained and increases confidence in the trends observed.

Table 2. Simulation Results

Simulation	**Simulation 1: Partial Cooperation**				**Simulation 2: Partial Cooperation II**				**Simulation 3: Full Cooperation**				**Simulation 4: Full Cooperation with Pop. Regeneration**			
End Fitness Score	35	44	40	47	37	39	38	49	41	46	40	47	55	58	55	66
Initial Fitness Score	51	60	52	64	48	60	56	65	53	58	56	61	41	47	40	44
Population Corr.	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Biomarker Change	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
Model Temporal Correction	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
MI Eq. 1	0.000		0.000		0.000		0.000		0.000		0.034		0.000		0.000	
MI Eq. 2	0.024		0.019		0.000		0.010		0.096		0.024		0.000		0.000	
MI Eq. 3	0.014		0.017		0.005		0.000		0.070		0.011		0.128		0.000	
MI Eq. 4	0.000		0.000		0.000		0.037		0.000		0.000		0.033		0.000	
MI Eq. 5	0.318		0.249		0.391		0.351		0.281		0.263		0.203		0.249	
MI Eq. 6	0.103		0.262		0.265		0.086		0.267		0.251		0.198		0.355	
MI Eq. 7	0.000		0.105		0.000		0.000		0.048		0.071		0.011		0.030	
MI Eq. 8	0.371		0.307		0.298		0.344		0.236		0.267		0.202		0.300	
MI Eq. 9	0.051		0.000		0.000		0.000		0.000		0.000		0.025		0.000	
MI Eq. 10	0.119		0.040		0.041		0.171		0.003		0.080		0.199		0.065	
MI Eq. 11	0.000		0.000		0.000		0.000		0.000		0.000		0.000		0.000	
Stroke Eq. 1	0.031		0.237		0.097		0.128		0.100		0.135		0.082		0.124	
Stroke Eq. 2	0.124		0.137		0.331		0.096		0.165		0.039		0.162		0.049	
Stroke Eq. 3	0.382		0.090		0.010		0.151		0.196		0.094		0.167		0.183	
Stroke Eq. 4	0.231		0.144		0.238		0.075		0.102		0.144		0.128		0.000	
Stroke Eq. 5	0.011		0.009		0.000		0.003		0.131		0.055		0.026		0.000	
Stroke Eq. 6	0.034		0.010		0.000		0.084		0.009		0.060		0.049		0.080	
Stroke Eq. 7	0.038		0.068		0.000		0.212		0.024		0.158		0.062		0.172	
Stroke Eq. 8	0.000		0.055		0.046		0.000		0.000		0.068		0.042		0.000	
Stroke Eq. 9	0.059		0.053		0.000		0.100		0.156		0.055		0.106		0.121	
Stroke Eq. 10	0.000		0.013		0.047		0.000		0.032		0.000		0.000		0.000	
Stroke Eq. 11	0.000		0.045		0.000		0.007		0.038		0.023		0.002		0.000	
Stroke Eq. 12	0.091		0.140		0.231		0.145		0.047		0.168		0.174		0.271	
Death MI Eq. 1	0.333		0.333		0.333		0.333		0.299		0.302		0.276		0.308	
Death MI Eq. 2	0.333		0.333		0.333		0.333		0.349		0.412		0.371		0.422	
Death MI Eq. 3	0.333		0.333		0.333		0.333		0.351		0.286		0.353		0.269	
Death Stroke Eq. 1	0.500		0.500		0.500		0.500		0.566		0.530		0.452		0.588	
Death Stroke Eq. 2	0.500		0.500		0.500		0.500		0.434		0.470		0.548		0.412	
Death Eq. 1	0.500		0.500		0.500		0.500		0.525		0.475		0.534		0.525	
Death Eq. 2	0.500		0.500		0.500		0.500		0.475		0.525		0.466		0.475	

CONCLUSIONS AND DISCUSSION

Based on results obtained, it is clear that the assumption engine is capable of selecting a set of equations that improves fitness. The assumption engine can also be used as a mechanism to reject equations. For example MI equation #11 is a new addition to the model, yet it was rejected in all simulations. This may indicate an error, either in implementation or in design. Or it may signify that this equation is incompatible with other equations and elements of data. It is also possible that it behaves better if another query is presented to the assumption engine that emphasizes importance of certain populations.

Future simulations will address these issues by exploring more components. One simulation of interest will attempt to include the temporal correction assumption to the simulations to correct for outdating of data. Such corrections proved useful in the past when using competing components. With cooperating components it may be possible to better explore model outdating and with temporal correction older models may be better fitting newer data sets or newer models may be able to better explain progression in older populations. Yet this is only one assumption that is planned for simulation, many other models/assumptions may follow with the existence of an assumption engine to analyze them. Recall that a model is just an assumption of how reality behaves.

The assumption engine provides us important insight on our modeling assumptions and tools that were not available before. Such information can be given as feedback to modelers in order to improve their models, since comparison to other models is now being made per component. Yet, the importance of an assumption engine is as a dynamic aggregator of knowledge. Knowledge gets accumulated in two forms: 1) Models and assumptions which are active computational building blocks, 2) Input data and validation data – in this specific case clinical trial summary data for populations and outcomes. On the long run, the aggregation capability of assumption engines and the availability of computing power to handle the increasing amount of information will improve predictive models.

REPRODUCIBILITY INFORMATION

The results for this paper were calculated on a 16 core cluster with 5 nodes running Ubuntu 12.04 Linux using Sun Grid Engine and Python 2.7.8 deployed by Anaconda 2.0.1 (64-bit). The Reference Model results were generated using MIST version (0,94,1,0) with Inspyred version 1.0 and model version 34. Results are archived in: MIST_RefModel_2016_02_26_OPTIMIZE.zip, MIST_RefModel_2016_03_14_OPTIMIZE.zip, MIST_RefModel_2016_04_05_OPTIMIZE.zip, MIST_RefModel_2016_05_20_OPTIMIZE.zip. Numbers appearing in paper were rounded for display purposes.

ACKNOWLEDGEMENTS

Many thanks to Aaron Garrett and W. Andrew Pruett who helped develop the mathematical technique that helped power this work. Thanks to Philip Standiford for proof reading the paper. Special thanks to Lorie Ingraham for providing valuable feedback. This paper used the Micro Simulation Tool (MIST) that is based on IEST. The IEST GPL disease modeling framework was initially supported by the Biostatistics and Economic Modeling Core of the MDRTC (P60DK020572) and by the Methods and Measurement Core of the MCDTR (P30DK092926), both funded by the National Institute of Diabetes and Digestive and Kidney Diseases. The modeling framework was initially defined as GPL and was funded by Chronic Disease Modeling for Clinical Research Innovations grant (R21DK075077) from the same institute. The Reference Model for Disease Progression and MIST framework were developed without financial support.

REFERENCES

Asche C., Ren J., Kim M. Carmen K., Dong Y., & Hippler S. (2016) A Prediction Model to Identify Acute Myocardial Infarction (AMI) Patients at Risk for 30-Day Readmission. *SummerSim 2016*, Montreal, CA.

Barhak J., Garrett A., & Pruett W. A. (2016). Optimizing Model Combinations, MODSIM world, Virginia Beach, VA. Paper retrieved from: http://www.modsimworld.org/papers/2016/Optimizing_Model_Combinations.pdf Presentation: http://sites.google.com/site/jacobbarhak/home/MODSIM2016_Submit_2016_04_25.pptx

Barhak J. (2014). The Reference Model for Disease Progression – Data Quality Control. Monterey CA. Paper retrieved from: http://dl.acm.org/citation.cfm?id=2685666 Presentation retrieved from: http://sites.google.com/site/jacobbarhak/home/SummerSim2014_Upload_2014_07_06.pptx

Barhak J. (2015a). The Reference Model uses Object Oriented Population Generation. *SummerSim 2015*. Chicago IL, USA. Paper retrieved from: http://dl.acm.org/citation.cfm?id=2874946 Presentation retrieved from: http://sites.google.com/site/jacobbarhak/home/SummerSim2015_Upload_2015_07_26.pptx

Barhak J. (2015b). The Reference Model for Disease Progression and Latest Developments in the MIST, *PyTexas 2015*. College Station, TX. Presentation retrieved from: http://sites.google.com/site/jacobbarhak/home/PyTexas2015_Upload_2015_09_26.pptx Video retrieved from: https://www.youtube.com/watch?v=htGRRjia-QQ

Barhak J. (2013), MIST: Micro-Simulation Tool to Support Disease Modeling. *SciPy, 2013*, Bioinformatics track, https://github.com/scipy/scipy2013_talks/tree/master/talks/jacob_barhak Video retrieved from: http://www.youtube.com/watch?v=AD896WakR94

Bell R.M., Bennett J., Koren Y., & Volinsky C. (2009). The Million Dollar Programming Prize. *IEEE Spectrum*. Retrieved from http://spectrum.ieee.org/computing/software/the-million-dollar-programming-prize

Body Interact (n.d.) Retrieved May 16,,2016 from http://bodyinteract.com/

Clarke P.M., Gray A.M., Briggs A., Farmer A.J., Fenn P., Stevens R.J., . . ., &UK Prospective Diabetes Study (UKDPS) Group (2004). A model to estimate the lifetime health outcomes of patients with type 2 diabetes: the United Kingdom Prospective Diabetes Study (UKPDS) Outcomes Model (UKPDS no. 68). *Diabetologia*, 47(10),1747-59. http://dx.doi.org/10.1007/s00125-004-1527-z

Gore R. Chandra M.-S. (2015). PACT: participant-centered clinical trial framework. *SummerSim 2015*, Chicago, IL.

Harvey C. (2016) The Kidney Transplant Process Model (KTPM): Simulation Tool for the Transplant Process. *SummerSim 2016*, Montreal, CA.

Hippisley-Cox J., Coupland C., Vinogradova Y., Robson J., Minhas R., Sheikh A., Brindle P. (2008), Predicting cardiovascular risk in England and Wales: prospective derivation and validation of QRISK2, *BMJ* 336, 26, http://dx.doi.org/10.1136/bmj.39609.449676.25x

Hayes A.J., Leal J., Gray A.M., Holman R.R., & Clarke P.M. (2013). UKPDS outcomes model 2: a new version of a model to simulate lifetime health outcomes of patients with type 2 diabetes mellitus using data from the 30 year United Kingdom Prospective Diabetes Study: UKPDS 82. *Diabetologia, 56(9), 1925-33*. http://dx.doi.org/10.1007/s00125-013-2940-y

Klusek A., Dzwinel W., & Dudek A (2016). Simulation of Tumor Necrosis in Primary Melanoma. *SummerSim 2016*, Montreal, CA.

Lasry A., Zaric G.S., & Carter M.W. (2007), Multi-level resource allocation for HIV prevention: A model for developing countries. *European Journal of Operational Research* 180, 786–799 http://dx.doi.org/10.1016/j.ejor.2006.02.043

Palmer A.J., & The Mount Hood 5 Modeling Group (2013). Computer Modeling of Diabetes and Its Complications: A Report on the Fifth Mount Hood Challenge Meeting, *Value in Health*, 16(4), 670-685. http://dx.doi.org/10.1016/j.jval.2013.01.002

Salem D., & Smith? R. (2016) A Mathematical model of Ebola Virus Disease: Using Sensitivity Analysis to Determine Effective Intervention Targets. *SummerSim 2016*, Montreal, CA.

Santamaria A. F., Serianni A., Raimondo P., De Rango F. & Froio M. (2016) Smart wearable device for health monitoring in the Internet of Things domain *SummerSim 2016*, Montreal, CA.

Sperl-Hillen J. M., Averbeck B., Palattao K., Amundson J., Ekstrom H., Rush B., O'Connor P., (2010). Outpatient EHR-Based Diabetes Clinical Decision Support That Works: Lessons Learned From Implementing Diabetes Wizard. Diabetes Spectrum 23 (3) 150-154 . http://dx.doi.org/10.2337/diaspect.23.3.150

Stevens R., Kothari V., Adler A., Stratton I. (2001), The UKPDS risk engine: a model for the risk of coronary heart disease in type II diabetes UKPDS 56, *Clinical Science* 101: 671-679.

The Mount Hood 4 Modeling Group (2007). Computer Modeling of Diabetes and Its Complications, A report on the Fourth Mount Hood Challenge Meeting. *Diabetes Care*, (30), 1638–1646. http://dx.doi.org/10.2337/dc07-9919