Developing computational model-based diagnostics to analyse clinical chemistry data

Daniël B. van Schalkwijk, Kees van Bochove, Ben van Ommen, Andreas P. Freidig, Eugene P. van Someren, Jan van der Greef and Albert A. de Graaf

Submitted: 30th July 2009; Received (in revised form): 4th December 2009

Abstract

This article provides methodological and technical considerations to researchers starting to develop computational model-based diagnostics using clinical chemistry data. These models are of increasing importance, since novel metabolomics and proteomics measuring technologies are able to produce large amounts of data that are difficult to interpret at first sight, but have high diagnostic potential. Computational models aid interpretation and make the data accessible for clinical diagnosis. We discuss the issues that a modeller has to take into account during the design, construction and evaluation phases of model development. We use the example of Particle Profiler development, a model-based diagnostic tool for lipoprotein disorders, as a case study, to illustrate our considerations. The case study also offers techniques for efficient model formulation, model calculation, workflow structuring and quality control.

Keywords: computational modelling; clinical diagnostics; Particle Profiler; lipoprotein metabolism; clinical chemistry; probabilistic and deterministic modelling

Computational models have a wide range of applications in clinical setting. For instance, they are used for increasing the understanding of disease processes (see e.g. [1]), for evaluating the clinical efficacy and cost-effectiveness of existing diagnostics (see e.g. [2]), during the drug development process [3, 4], and for developing new diagnostics. In this last area, computational models are typically employed to assist clinicians in the interpretation of complex data. For example, novel imaging techniques provide a large amount of data, which are hard to interpret at first sight. Still, this data contains a wealth of relevant medical information which computational models can extract (see e.g. [5]). Also in the field of clinical chemistry, novel ‘metabolomics’ techniques are generating an increasing amount of complex data, that provide interpretational challenges [6, 7]. In this article, we focus on the development of mathematical models that interpret complex clinical chemistry data, leading to model-based diagnostics. Several mathematical models have already been developed for interpreting clinical chemistry, for instance in

Corresponding author. Daniël B. van Schalkwijk, Utrechtseweg 48, P.O. Box 360 (PP 5), 3700 AJ Zeist, The Netherlands. Tel.: +31 30 694 4553; Fax: +31 30 694 4986; E-mail: daan.vanschalkwijk@tno.nl

Daan van Schalkwijk is a PhD student at the TNO Quality of Life Research Institute and at the Leiden Amsterdam Centre for Drug Research at Leiden University.

Kees van Bochove has done his master’s thesis in bioinformatics at the TNO Quality of Life Research Institute, in collaboration with the TUFTS University in Boston.

Ben van Ommen is a senior research fellow at the TNO Quality of Life Research Institute and Director of the NuGO network of excellence funded by the European Union.

Andreas Freidig is now a project manager at Amsterdam Molecular Therapeutics after working at the TNO Quality of Life Research Institute.

Eugene van Someren is a research scientist in bioinformatics at the TNO Quality of Life Research Institute.

Jan van der Greef is a Professor of the Analytical Sciences division at the Leiden Amsterdam Centre for Drug Research at Leiden University. He also leads systems biology research at the TNO Quality of Life Research Institute.

Albert de Graaf leads the computational modelling effort at the TNO Quality of Life Research Institute.

© The Author 2010. Published by Oxford University Press. For Permissions, please email journals.permissions@oxfordjournals.org
the fields of diabetes [8] and cancer [9] diagnostics. In the context of diabetes, one mathematical model extracts an insulin resistance measure from a time series of plasma glucose levels after a standard oral glucose dose [10]. The diagnostic model for cancer calculates tumour size based on blood biomarkers [9]. In both cases, the model-based diagnostics assists clinical decision making by interpreting complex clinical chemistry data.

The importance of modelling for clinical diagnostics is increasingly recognised in the context of the emerging e-health discipline [11]. The European Union 7th Framework Programme supports research into ‘personal health systems’; this program explicitly targets ‘systems and tools which can be used for diagnostic purposes’ (http://ec.europa.eu/information_society/activities/health/research/fp7phs/). This program indicates that the development of computational models to make complex data types interpretable for clinicians will become increasingly important over the coming years. Therefore, we discuss modelling methodology and techniques to get researchers who are new to developing model-based diagnostics for clinical chemistry applications started. We first give an overview of relevant issues to consider in various stages of model development, with a focus on the early stages, and then provide a case study to illustrate some of these issues in the model development process.

CONSIDERATIONS FOR MODEL DEVELOPMENT

We will discuss three stages in model development: deciding what to model, constructing the model, and validating the model-based diagnostics. We will focus our discussion on issues related to modelling for clinical diagnosis using clinical chemistry data. In continuation, second part of the article describes a case study that illustrates the concepts discussed in this first part.

Deciding what to model

In the first stage of model development, the decision is made regarding what to model. First, the choice of a clinically relevant problem is highly important for the reception of the model once it is developed. Second, the clinical setting places demands on the data type the model uses as input. Third, data for model development are needed. Last, the diagnostic readout is important as it determines the usefulness of the model for the clinician. These issues are covered in the next sections.

Choosing an objective with clinical relevance

The model-based diagnostics should fill a need for better diagnostic capabilities in the medical community. Therefore, a careful exploration of medical necessities is an important first step, and a necessary condition for the success of the modelling project. The model-based diagnostics should also include biomedical understanding of the disease. Such understanding is necessary to interpret the large datasets the model analyses, and to extract the relevant information from the data for the clinician. Only those processes that have a direct bearing on the selected data type and the disease being studied are of interest. Therefore, the second exploration should focus on the scientific knowledge present in the subject area. Furthermore, the model-based diagnostics should aim to be a target that can be improved by therapy. Doctors want to be able to improve a patient’s health through medication, and unhealthy values for a given diagnostic should be remediable by appropriate medication. An overview of available medication for the disease state to be diagnosed is, therefore, helpful for determining what the diagnostic should indicate.

Routine clinical chemistry data

It is important to distinguish between two data types: data available in the routine clinical environment, such as blood chemistry, cytokines, and liver enzymes, and research data available from small studies that can be used during model development and validation. We will first focus on the clinically available data, which is the data that is intended to be fed into the finished model and, from which, the model extracts a diagnostic parameter. The clinical data type should be carefully selected, since the properties of the model-based diagnostics reflect the properties of the data it is based on.

First, the measurement data that the model will need to use for diagnosis should either be clinically available, or become available in the near future. For example, mass spectrometry data might be useful to build the model, but a mass spectrometer is unlikely to become a standard clinical tool in the near future. It is also important to keep in mind that the diagnostic will need to be validated in a large-scale experiment. If the data type has already been measured in a large cohort study, which is available for
analysis, this situation can considerably shorten the development time of the diagnostic and increase its chances of acceptance. In order to use clinical data for modelling, consent from the principle investigators of the clinical study is required.

Second, the selected data type should be standardised and reproducible with low error margins. Any errors in the data will propagate into the model-based diagnostics through calculations and/or data fitting. Therefore, standardisation and reproducibility of the clinical data are necessary conditions for a standardised and reproducible model-based diagnostics.

Third, the clinical data should have been biologically validated, for example, by comparing the measurement with other measurements. The diagnostic should give reliable information about biological reality, which the model-based diagnostics can further process. Therefore, biologically validated data is a necessary condition for a biologically validated model-based diagnostics.

Fourth, the clinical data should be measurable at low cost, or these costs should be expected to drop in the future. The diagnostics budget in hospitals is limited, and a diagnostic always needs to be measured repeatedly. Therefore, hospitals will use low-cost diagnostics, unless expensive diagnostics are the only available option.

Finally, the clinical data should be difficult to interpret, yet implicitly reflect the disease, so that the model analysis has added value. Easily interpretable data does not need model analysis for interpretation. On the other hand, the model cannot distil useful information from complex data when no relation with the disease process is present. Therefore, the data should contain non-obvious relevant diagnostic information.

Research data for model development
There are fewer restrictions on the research data used for model development, since these data need not be used routinely. Modellers may therefore use more expensive and elaborate measurements for development purposes. The biological accuracy of model validation depends on this data, which should, therefore, be both available and sufficiently accurate to test all relevant biological aspects of the model.

Diagnostic readout
Finally, when deciding what to model, the diagnostic readout of the model should be considered. This readout will be presented to the clinician. Therefore, it should give a good indication of the state of the biological processes associated with the disease, preferably in a single measure. It is a good idea to ask clinicians whether they would be pleased to have the proposed model’s diagnostic readout at their disposition.

Constructing the Model
Once it has been decided what output needs to be generated based on what data, the model development phase starts. In this phase, the modeller needs first to choose a suitable model type, and then implement the model.

Choosing a model type
Different model types have their different strengths and different uses for clinical diagnostics. We will start the discussion with generic model types, and proceed to models that can incorporate progressively more biological knowledge into the structure of the model. Statistical models are very generic tools that are essential in diagnostics, since they are often used to relate clinical measurements to disease outcomes. One example is a diagnostic score for Sjögren’s syndrome, a systemic autoimmune disease, based on clinical and laboratory variables [12]. Besides statistical models, other model types have been used to link clinical measurements directly to disease outcomes without biological input into the model structure. For example, the ProstAsure Index uses an artificial neural network analysis of serum measurements to indicate the presence of prostate cancer [13]. Schneider et al. [14] used a fuzzy logic-based system to diagnose lung tumours based on plasma and serum markers. However, as more potential diagnostic markers are measured, more predictors of a single disease state are possible. It then becomes more likely that some diagnostic measurements seem to be predictive for the diseased state by chance. The models discussed above cannot remedy this ‘multiple comparison’ problem. To circumvent this problem, modellers are able to ‘pre-process’ the data using models that incorporate biological background knowledge. The integrated biomarkers derived from these models can then be related to the disease state using statistical techniques (see Figure 1).

Different model types are suitable for investigating different aspects of biological understanding. For instance, agent-based models are suitable for understanding higher-level properties from lower level system properties [15]. They consist of individual
Figure 1: Schematic representation of the role of various techniques in diagnostic development. When few candidate diagnostics have been measured, statistics alone suffice to select the best diagnostic from the data. When many candidate diagnostics have been measured, an intermediate modelling step that adds biological interpretation becomes necessary.

Data

Model

Diagnostic

agents’, each of which have their own behavioural rules and together execute the behaviour of a system. In this way, for instance, the self-organisation of human keratinocytes in a petri dish has been modelled, leading to increased biological understanding of the system [16]. Another example is the class of probabilistic models, which are suitable for exploring the influence of random variation in a given biological system. The model specifies biological processes, including probabilities that the process occurs. An example of such a model is an enzymatic reaction in which the reagents have a certain probability to interact with the enzyme, instead of a fixed rate (see e.g. [17]). Bayesian networks are models that combine the use of data-driven discovery with integrating biological knowledge into the interpretation of complex data [18–20]. Bayesian networks can be used to describe approximate relations between nodes that represent biological entities such as mRNA, protein or metabolite levels. The modeller can interpret the network, and identify relevant ‘hubs’ in the network, containing compounds that contribute to the disease process. In this way, Bayesian networks can integrate biological knowledge into a relevant model-based diagnostic. The choice of model type, therefore, depends on the type of understanding needed to analyse the complex data.

Deterministic models are able to integrate relevant biological knowledge into the analysis of complex data in many ways. There are several important types that have been used in the clinic. One example is compartment modelling, in which flow rates of substances through biologically defined compartments are modelled. The computational model that predicts tumour size from serum biomarkers is of this type [9]. This model quantifies the flux of a biomarker through the plasma compartment, where the influx can come either from normal or from tumour cells. The biomarker flux from tumour cells then gives an indication of tumour size. Compartmental modelling is therefore suitable for interpreting complex clinical data in a diagnostically relevant way.

Compartmental modelling is a specific type of the broader class of Ordinary Differential Equation (ODE) models [21]. A differential equation is the relation between a function and the function’s derivative, which is, for example, used in biology as the relation between a substance’s concentration and the concentration’s rate of change. Examples of diagnostic models of this kind include a model for the blood coagulation process, which simulates a measured time course of coagulation factor activities [22], and a model to derive insulin sensitivity parameters from an oral glucose tolerance test (OGTT), which simulates a measured time course of plasma glucose values after an oral glucose dose [10]. The model structure can include relevant biological information, for instance, concerning which relevant processes are involved in the modelled dynamics. For some applications, it may be suitable to introduce a ‘steady state’ assumption, which means that all fluxes in the system have reached a balance and do not change any more with time. This assumption can considerably simplify the model, but it can only be used in stable systems. The examples illustrate that diagnostic ODE models can be an effective way of interpreting complex data for diagnostic use.

Partial differential equations (PDEs) are similar to ODEs, but with the difference that ODEs pertain to functions of one variable, usually time, and PDEs pertain to functions of two or more variables, usually time and space [21]. This property makes PDEs especially suited for the analysis of dynamic imaging data. Since our focus is here on modelling clinical chemistry, we will leave PDEs aside.

In summary, different model types have different strengths and different uses for clinical diagnostics. Statistical models are important, since they can relate clinical measures to disease outcomes.
However, in the case of large and complex data, the multiple comparison problem arises. Biological interpretation then becomes important as an intermediary step, for which differential equation modelling is an example of a suitable tool.

**Implementing the model**

The best strategy for implementing the model depends on the type of model that is being built. For statistical models, a large number of packages is available (http://en.wikipedia.org/wiki/List_of_statistical_packages). For Bayesian models, a commonly used package is Bayes Net for MATLAB, but other options exist (http://people.cs.ubc.ca/~murphyk/Bayes/bnsoft.html). For ODE models, many different modelling environments exist. At the level of biochemical pathways, which is often done with ODEs, researchers have agreed upon standard ways of formulating their models, called SBML (http://sbml.org/) and CELLML (http://www.cellml.org/). The respective websites of these standards contain a large number of packages that use the standard. More general modelling environments such as Octave, MATLAB, FreeMAT, Mathematica and others (http://en.wikipedia.org/wiki/List_of_numerical_analysis_software) allow more flexibility in model formulation, but they are also more complex to work with, and require programming skills.

If the envisioned model cannot be formulated entirely in the specified formats, more general modelling environments are necessary. For example, the Particle Profiler model we developed [23] could not be constructed using standard SBML models. In consequence, many technical issues need to be solved that have already been taken care of in standard modelling environments. We, therefore, include a case study about the Particle Profiler model development as the second part of this article, in which we discuss the technical issues we encountered.

**Validating the diagnostic**

Validating a model-based diagnostic consists of two steps: validating the model and validating the diagnostic. When validating the model, the modeller needs to show that the model reflects biological reality, also in extreme cases such as genetic variants and medicinal interventions. The model outcome needs to correspond to measured data and current biological insight. To measure the correspondence between model prediction and data, a modeller can use the ‘least squares method’, which calculates the squared deviations between measured and predicted values, appropriately weighted by, for example, the measurement error of the data. [21]

Validating a model-based diagnostic is no different from validating other diagnostics. The criteria to judge a diagnostic are: how much it improves risk prediction and whether the costs of the complete diagnostic test are acceptable. The improvement of risk prediction can be measured using sensitivity and specificity statistics, or more completely, using receiver operating characteristic (ROC) curves. The sensitivity of a test is defined as the proportion of people with disease who are predicted to have the disease. The specificity of a test is defined as the proportion of people without the disease that are predicted not to have the disease. A practical explanation of these concepts is given by Akobeng [24]. A diagnostic test always includes a ‘decision threshold’, above which patients are considered ill. The ROC curve describes how the sensitivity and specificity of the diagnostic test depend on the choice of the decision threshold. An ROC curve, therefore, allows comparison of the performance of different diagnostics in a way that is independent of the decision threshold (see [25] for an in-depth explanation of ROC curves).

**CASE STUDY—PARTICLE PROFILER DEVELOPMENT**

In this section, we present the case study of Particle Profiler, a computational model developed for clinical diagnostic purposes. We will examine the various stages of model development, especially the technical implementation of the model.

**Deciding what to model**

Clinical relevance

Particle Profiler was developed to improve cardiovascular disease risk estimation. Current well-established indicators of this risk are ‘total cholesterol’, ‘high low-density lipoprotein (LDL)’ and ‘low HDL cholesterol concentrations’. Still, there is room for improvement, especially since medication for reducing risk by raising the HDL concentration have been found to increase mortality during clinical trials [26, 27]. In addition, the standard approach of LDL–cholesterol calculation using the Friedewald formula is not always accurate [28]. Therefore, there is a need for better cardiovascular risk indicators in the clinic.
The medical–biological knowledge in the field of cholesterol and lipoproteins is very extensive. Lipoproteins are particles that transport triglycerides from the liver to other tissues. The particles contain triglycerides and cholesterol esters in their core, while the membrane around the core consists of phospholipids, free cholesterol and proteins called apolipoproteins that take care of many of the biochemical interactions of the lipoprotein (inset in Figure 2). An excess of lipoproteins in the blood, especially LDL, is one of the causal factors leading to atherosclerosis. There are three main processes in the lifecycle of a lipoprotein: lipoprotein production, lipolysis, and uptake. LDL lipoproteins are usually produced as large, triglyceride-rich VLDL particles that become smaller when losing their core triglycerides through lipolysis, and become LDL. Important biochemical entities involved in lipoprotein metabolism are apolipoprotein B, apolipoprotein E, the LDL receptor, heparan sulphate proteoglycans, lipoprotein lipase, and hepatic lipase. There is, therefore, extensive background knowledge to base a model on.

In summary, there is a need for improved diagnostic markers in cardiovascular disease; it is a developed field of research, and different medication options are available. The area meets the prerequisites for developing a useful clinical diagnostic model.

Clinically available data
Detailed lipoprotein subclass measurements constitute a promising data type for cardiovascular risk assessment (see Figure 2). Techniques for measuring lipoprotein profiles include Nuclear Magnetic Resonance [32], High Performance Liquid Chromatography [33], Asymmetric Flow-Field Flow Fractionation [34], and Vertical Auto Profile [35]. Detailed lipoprotein measurements are clinically available, but currently only as a separate measurement by specific companies (http://www.lipo-search.com/eng/index.html; http://www.liposcience.com/). One company is planning a more widespread introduction of their technology into clinical practice (http://www.liposcience.com/).
Their data has been standardised, and has proven to be reproducible, it has also been compared with other laboratory measurements [36]. The current cost of the measurement is still rather high, but the cost is expected to drop as the technology is further introduced. The importance of these data types is increasingly recognised, but it is hard to say which lipoprotein subclass contributes most to cardiovascular disease development. Therefore, in practice, subclasses are often pooled, which facilitates interpretation but in fact is unwanted since it reduces the measured information. This measured information is likely to be relevant, because several aspects of the lipoprotein phenotype are known to contribute to cardiovascular risk together, in a so-called ‘Atherogenic Lipoprotein Phenotype’ [37]. This situation is a good example in which a computational model has an opportunity to extract more relevant information by interpreting the data biologically. In summary, there exists a novel standardised, reproducible, and validated new diagnostic that is likely to become available in the clinic in the future at a reasonable cost, and for which a model approach is necessary to fully exploit the diagnostic potential of the data.

Data for model development
Data for the validation of a model on lipoprotein kinetics was already publicly available. In the field of lipoprotein kinetics, many studies have been carried out to measure fluxes of lipoproteins using stable isotope techniques (see e.g. [38]). This data was useful for model development and validation [23].

Model output
The output of the new Particle Profiler model is an indication of the balance between lipoprotein production, lipolysis and uptake processes. An imbalance in these processes is often the cause for cholesterol disorders. The output can give an indication of the functioning of specific organs, such as the liver, by comparing the rate of two of its functionalities: extracting fats from particles in the blood, or taking up the whole particle. Since the liver produces lipoproteins, lipolyses and takes up lipoproteins, liver function is an important aspect of the disease state of the patient, which is expected to help the clinician predict cardiovascular risk.

Constructing the Model
Choosing a model type
During the development of Particle Profiler, we first formulated a probabilistic model and later converted it to a deterministic model. In the probabilistic model, we could specify biological processes at the level of individual lipoprotein particles. This line of thinking led to a greater insight in the biological system, which proved crucial for later model development. To make the model suitable for developing a diagnostic, it needed to become faster and able to fit the diagnostic data, and hence we switched to a deterministic model. We chose an ODE-type model, which includes the steady state assumption. This assumption is necessary to analyse data from a single clinical measurement. Steady state in a lipoprotein measurement is only realistic if the sample is taken after an overnight fast, as is clinical practice. The switch to deterministic modelling allowed us to speed up the simulations by a factor 100.

Implementing the model
Because the lipoprotein profile data we want to analyse with the model contains lipoprotein concentrations in many particle size classes (Figure 2), Particle Profiler specifies how the rate of lipoprotein production, lipolysis and uptake varies with the size of the particle. This approach leads to a type of model that cannot be formulated using standard compartment modelling, and therefore cannot be implemented in the SBML or CELLMl standards. We, therefore, used the more flexible MATLAB and MathCAD environments for developing the model. Model development in these environments requires the use of additional modelling techniques, which we will briefly discuss.

Optimising the parameterisation
In order to derive the model-based diagnostic from clinical data, the model should be fitted to the data with a parameter fitting routine. In the case of Particle Profiler, the parameters represent the rates of metabolic processes affecting lipoproteins. A parameter fitting routine adjusts these processes in such a way that the resulting modelled lipoprotein profile matches the measured lipoprotein profile. The process parameters then contain the diagnostic information. A parameter fitting routine is like a mountaineer trying to find the lowest point in a mountainous landscape. The co-ordinates of the ‘parameter landscape’ walk along are the model’s
possible parameter values, while the height represents the error function (see Figure 3). This error function is a measure of the distance between the model prediction and the measured data, given a set of parameters. The lowest point, therefore, indicates the parameter settings that give the best model prediction.

Fitting routines have a hard time if parameter landscapes have multiple local minima or outstretched valleys. This problem frequently occurs with non-linear models. Models can be specified using different sets of parameters, and the choice of parameterisation affects the shape of the parameter landscape (see Figure 3). Carefully choosing a parameterisation that leads to a landscape with an accessible minimum helps the fitting routine.

Constructing a good parameterisation is a question of trial and error, guided by two principles. First, a small change in the parameter should give a large change in the error function (sensitivity), and second, no other parameter in the model should influence the error function in the same way (correlation). ‘Identifiability analysis’, which looks at both parameter sensitivity and correlation, can determine how well these criteria are met (see [39] for an in-depth discussion).

**Efficient and accurate parameter fitting routine**

We have seen that a fitting routine is like a heavy-duty mountaineer walking down to the lowest point in a multi-dimensional landscape. Parameter fitting is a time-consuming exercise; yet it is important, since the optimised model parameters contain much of the diagnostic information. A good diagnosis, therefore, requires a correct model fit; it requires the parameter combination that results in the best correspondence between data and model prediction. This parameter combination is called the global optimum.

The field of parameter optimization is extensive (see e.g. [40, 41]). Numerical libraries are available in several programming languages such as FORTRAN, Octave, R and MATLAB. The algorithms come in two classes: local and global. Global methods search all over the parameter space, but they are slow and do not always converge to a minimum. Local methods are faster and converge to a minimum, but they require a starting point nearby. They may find a ‘local minimum’ that is not the global optimum. The most often used algorithms are local: Gauss–Newton and Levenberg–Marquardt [42]. A comparison of different methods can be found in [39]. When using a local optimisation method, finding...
an accurate parameter estimate within an acceptable time requires well-chosen starting points in parameter space.

For the Particle Profiler model, global optimisation is too time-consuming. Therefore, we used the Levenberg–Marquardt algorithm [42], implemented in MATLAB’s nlinfit method [MATLAB version 7.5.0(R2007b), statistics toolbox version 6.1]. To select starting points for the algorithm, we evaluated all combinations of a high and low estimate for each parameter; subsequently, we evaluated middle points between the most promising combinations. From the evaluated points, we chose those with the lowest error function as starting points for the fitting routine. We applied the fitting routine to its own outcome two more times, to assure reaching a minimum. We then inspected whether different starting points converged to the same minimum. If they did, we were done; if not, we repeated the procedure within a more limited parameter space. In this way, we could effectively search a large parameter space [23].

Programming tricks
As a mountaineer needs light equipment, an efficient model needs fast code. To reach this goal, several programming tricks are helpful.

Once the model is running, the code can be analysed using functions that clock the code’s execution time. For example, MATLAB returns the time it takes to execute all the program lines in between the ‘tic’ and ‘toc’ commands. The ‘profiler’ tool is more advanced, since it shows in which subfunctions and at which lines the program spends calculation time. Improving the performance of sluggish code may then be done by replacing slow functions or decreasing matrix size. Implementing a cache is another trick to speed up code. The cache stores the outcome of an often-executed calculation as a variable, which the program can then retrieve instead of re-calculating. Using these simple tricks, one can achieve a drastic reduction of computation time. During Particle Profiler development, these tricks have increased execution speed several times.

Hardware improvements, using grid computing
If the mountaineer is slow, you can still give him a jet engine. If the program is slow, you can still run it on better hardware. In the case of parameter estimation, faster hardware may be needed if the model parameters need to be fitted to large datasets, which could be the case in diagnostic applications; grid computing is then an option. The processors in the grid can fit portions of the dataset in parallel. Parallel computation also costs time, since it requires changes in the code and communication of tasks and results between computers. This time investment is only worthwhile when fitting large datasets.

Object-oriented programming
Just as we can compare a fitting algorithm with a mountain climber, we can compare constructing a computational model to constructing a car factory. Constructing a car in a factory, like constructing a computational model, can be done in two ways. First, one can start from the raw materials, produce the car in one large process, and then test it. Second, one can produce all parts separately, test them, assemble the car, and test again. Many modellers use a method similar to the first, they write one large ‘script’ file with all model commands. Although such files are easy to write, they just as easily lead to hidden mistakes. The alternative is called object-oriented programming, which is like the second car production method. The programmer thinks of a well-arranged program structure, based on program units called objects. He then writes each of the objects, tests them, assembles them into a working model program, and tests again. Each object consists of variables and functions to handle its variables; it contains all information necessary for its functioning. With such a clear and easy-to-use structure, mistakes become less easy to make and easier to track down.

All objects in the program can be tested using a ‘test object’, written for this purpose. The programmer can choose to write a test even before the tested object exists, the so-called Test Driven Development paradigm [43]. This method forces the programmer to think carefully about the object’s functionality before writing it. For example, a test might check whether a single function returns the correct value in some well-known cases. A more complicated test might check whether the entire model gives the correct outcome under several known conditions. Writing tests helps to identify accidental errors by running the tests again after each major modification. The initial time investment, therefore, quickly pays off through the abilities to spot errors quickly and to ensure correct functionality.
Functional tests are the most difficult to write. For a mathematical model, a functional test can be achieved by implementing the model in two calculation environments; the outcomes should correspond. We implemented Particle Profiler in both MATLAB and MathCAD. Other software packages such as Maple, Mathematica, Octave and SAGE are also suitable. It is advisable to choose a software package capable of algebraic manipulation next to the object-oriented program, since algebraic formulation is more easily surveyable, which helps to spot mathematical errors. Algebraic manipulation can also considerably speed up model formulation and help to find an efficient algorithm.

Object-oriented programming is not supported by all programming languages. Some programming languages, like Java, have been designed for it; others, like MATLAB, Octave and R, support it.

For Particle Profiler, a program structure was designed and implemented in MATLAB as shown in Figure 4. Each module can be tested separately. The function of each module is clear, and a high-level command such as fit (fitterFactory) allows a whole dataset to be fitted at once. The effort it costs to devise such a structure is more than compensated by the advantages it brings afterwards, like ease-of-use, transparency of the code and testability.

Using cell-structured files to construct a workflow

Once all the car parts can be produced, they need to be assembled and inspected in a coordinated way. A computational model can be ‘assembled and inspected’ using a workflow; a workflow is a sequence of high-level commands to the model objects. The workflow flexibly allows inspection tasks to be done in any order the user prefers.

For Particle Profiler, a program structure was designed and implemented in MATLAB as shown in Figure 4. Each module can be tested separately. The function of each module is clear, and a high-level command such as fit (fitterFactory) allows a whole dataset to be fitted at once. The effort it costs to devise such a structure is more than compensated by the advantages it brings afterwards, like ease-of-use, transparency of the code and testability.

Using cell-structured files to construct a workflow

Once all the car parts can be produced, they need to be assembled and inspected in a coordinated way. A computational model can be ‘assembled and inspected’ using a workflow; a workflow is a sequence of high-level commands to the model objects. The workflow flexibly allows inspection tasks to be done in any order the user prefers.

In MATLAB, modellers can construct a simple workflow by using cell-structured files. These are simple text files containing different sections called cells (see Figure 5). MATLAB allows the execution of both the whole file at once and of individual cells as the user requests. This method allows redoing different parts of the analysis at any time. More advanced workflows can be constructed by dedicated programs like Taverna (http://taverna.sourceforge.net/) [44] and E-Bioflow (http://sourceforge.net/projects/e-bio-flow/).

Quick investigation of parameter changes using GUIs

Once a factory has built a car, the constructors need to test whether it drives well. In the same way, once you have a constructed a model to derive model-based parameters from data, you need to see whether it fits the data well. A common method is plotting the data next to the model simulation result. At times, the model will not reproduce the data well, which raises the question why. For such cases, it is useful to have a graphical user interface (GUI) which allows the user to vary parameter values and immediately see the effect on the model prediction. Loading an erroneous fit into the GUI, and evaluating how the model responds to changing the parameter values, quickly shows what is wrong.

The GUI can be extended with any features that come in useful. For example, it can include parameters from different model parameterisations. It can also include various types of graphs and a function to re-fit the model using the current setting in the GUI as initial condition. GUIs can be constructed in any programming language; MATLAB includes a convenient tool named GUIDE for constructing them easily. Figure 6 shows a GUI for the Particle
Profiler model that facilitates the evaluation of the parameter fits.

Using version control to structure programming and quality control

A car factory can be improved bit-by-bit, until it works well. If one keeps track of previous designs, malfunctioning changes can be reverted to working designs. Likewise, if one keeps track of the programming processes, older program versions can be reused if necessary. Version control systems, such as the Subversion software package (subversion.tigris.org), are designed for this purpose. Such an application allows the programmer to upload (or commit) his program versions to a server using a client application such as TortoiseSVN (tortoisesvn.tigris.org). The server saves all uploaded versions and allows the programmer to download any version at any time. The Subversion server also allows the construction of ‘branches’; these contain versions of the program that are, for example, released to the public, or used to generate results for a publication. The version control system can be accessed by various users; therefore, various scientists are able to collaborate on the same code and to reproduce each other’s results.

Figure 5: Particle profiler workflow in cell-structured file. Data loading, model analysis and model outputs are specified in different cells. They can be run all at once, or separately.
Reproducibility and transparency are important quality control criteria that version control helps to meet.

Validating the diagnostic

The Particle Profiler model leads to a model-based diagnostic, which indicates liver function based on a detailed measured lipoprotein profile. A first biological validation of the model has been presented [23], but further tests with genetic variants and medicinal interventions are required. For validation of the diagnostic, we are working on applying the model to data from a large cohort study, in our case the Framingham Heart Study [45]. From the data in this study, we will be able to calculate the sensitivity, specificity and ROC curves to compare the new diagnostic to existing diagnostics.

Key Points

- Computational modelling helps to make complex data interpretable for clinicians.
- We guide starting researchers through the subsequent phases of deciding what to model, modelling and model evaluation for a model-based diagnostic using clinical chemistry data.
- We present a case study of Particle Profiler development, a model-based diagnostic for lipoprotein disorders.

Acknowledgements

The authors gratefully acknowledge the helpful comments of Dr Jildau Bouwman.

FUNDING

The Netherlands Bioinformatics Centre [Biorange project SP 3.3.1 to D.v.S.].
References


