J Pharmacokinet Pharmacodyn (2014) 41:141–151
DOI 10.1007/s10928-014-9352-6

ORIGINAL PAPER

Ubiquity: a framework for physiological/mechanism-based pharmacokinetic/pharmacodynamic model development and deployment

John M. Harrold · Anson K. Abraham

Received: 16 October 2013 / Accepted: 24 February 2014 / Published online: 12 March 2014

Abstract Practitioners of pharmacokinetic/pharmacodynamic modeling routinely employ various software packages that enable them to fit differential equation based mechanistic or empirical models to biological/pharmacological data. The availability and choice of different analytical tools, while enabling, can also pose a significant challenge in terms of both, implementation and transferability. A package has been developed that addresses these issues by creating a simple text-based format, which provides methods to reduce coding complexity and enables the modeler to describe the components of the model based on the underlying physiochemical processes. A Perl script builds the system for multiple formats (ADAPT, MATLAB, Berkeley Madonna, etc.), enabling analysis across several software packages and reducing the chance for transcription error. Workflows can then be built around this package, which can increase efficiency and model availability. As a proof of concept, tools are included that allow models constructed in this format to be run with MATLAB both at the scripting level and through a generic graphical application that can be compiled and run as a stand-alone application.

Keywords MATLAB · ADAPT · PBPK · Berkeley Madonna · Graphical user interface

Introduction

Pharmacokinetic (PK) modeling of clinical population data is a well defined field in terms of the overall process and scope of analysis. Standard PK models are often used to describe drug disposition and the modeling workflow has been systematized around common data file formats, control streams that define the model structure, and modeling software (e.g. NONMEM). Clinical pharmacodynamic (PD) modeling, while more complicated, fits well into this paradigm of a modeling workflow with well defined inputs and expected outputs. This is partly a product of the regulatory environment, which requires a well documented and reproducible process that can be scrutinized.

In the pre-clinical discovery space, mathematical modeling and simulation is being applied to a wide range of problems including: understanding the impact of tissue distribution on efficacy and toxicity [1–3]; multi-scale models of in vivo efficacy [4]; identifying the ideal properties of molecules [5]. While these modeling activities can result in systems ranging from simple fit-for-purpose compartmental models to complex descriptions of pharmacological systems, they are being used increasingly to aid and influence project teams earlier in the development cycle [6, 7]. Compared to population-based pharmacostatistical modeling approaches that are typically used to analyze clinical data, preclinical PK/PD and systems modeling efforts may not be as structured and well defined.

Electronic supplementary material The online version of this article (doi:10.1007/s10928-014-9352-6) contains supplementary material, which is available to authorized users.

J. M. Harrold (✉) · A. K. Abraham
Translational Modeling & Simulation, Department of Pharmacokinetics, Dynamics, and Metabolism, Pfizer Worldwide R&D, 200 Cambridgepark Drive, Cambridge, MA 02140, USA
e-mail: jharrold@amgen.com

A. K. Abraham
e-mail: anson.k.abraham@pfizer.com

Present Address:
J. M. Harrold
Pharmacokinetics & Drug Metabolism, One Amgen Center Drive, Thousand Oaks, CA 91320, USA

By pulling heavily from diverse subject matter areas such as systems biology, engineering, and physics, systems modelers have access to many computational tools. This creates an accommodating environment for individuals with the expertise, as many tools can be utilized to perform an analysis and deliver outputs to the project team. However, because of the resources required for model generation and the desire to apply models more broadly, the model itself is a significant portion of the final product. A key drawback is having a model ensconced in a single software package that limits (by user expertise) the ability of others to use the model. Translating each individual model between software packages is cumbersome, time consuming and error prone.

These issues can be addressed directly by defining the model development workflow around this objective of increasing access to models. There are many facets to this problem, which center on the development of a way to abstractly describe the relevant components of a system. In this context, there are several languages currently available and in development, each with their own set of tradeoffs. The General Algebraic Modeling Language (GAMS), provides a convenient way to describe sets of information [8], and is typically used to characterize systems of algebraic equations for constrained optimization. While these tools have been implemented in analyzing PKPD systems [9], they remain on the periphery. At the preclinical and translational level, a significant portion of the analysis tools focus on systems described by coupled sets of ordinary differential equations (ODEs). For defining sets of ODEs there are several tools ranging from very general options such as Modelica [10], with implementations across several fields, to the more specific systems biology markup language (SBML) [11], originally designed to describe biochemical networks. This is also currently an active area of interest within the PK/PD community, as can be seen by the recently released PharmML specification [12] and the accompanying draft specifications for a Model Coding Language [13] from the Drug Disease Model Resource. PharmML is a comprehensive effort aimed at integrating population PK/PD software. The approach pursued in the work presented here is to strike a parsimonious balance between comprehensiveness and utility: a human readable text file that simplifies both describing the elements of a system and dissemination of the model.

Hence, the "Ubiquity" framework presented here attempts to accomplish three specific tasks: (i) simplifying the programming complexities associated with implementing systems models (multiple parameterizations, intricate descriptions of physiology, etc.); (ii) automate implementation of models across multiple software platforms in order to reduce obstacles to interoperability; and (iii) provide individuals with little or no modeling experience with the means to interrogate and explore the response of models to changes in both system parameters and dosing regimens. This was accomplished by defining a file format for describing a system that is software independent, and once compiled can produce human-readable input files for multiple software platforms. A deployable, standalone graphical user interface (GUI) was built on top of this framework that would allow anyone to operate the model.

Materials and methods

Workflow overview

The basic methodology presented here is outlined in Fig. 1. A system file with a well-defined set of descriptors is created. This file, intended to be a canonical source of information about the system describing aspects such as basic parameter information, default dosing details etc., is then converted using a supplied Perl script into many different output formats that currently include MATLAB (m-file and C) (2012b, The Mathworks, Natick, MA), ADAPT (version 5) [14], Berkeley Madonna [15], PottersWheel (version 3) [16], and Monolix [17, 18]—portions of generated output are shown in the included listings. The MATLAB files that are generated can then be read with the supplied GUI, which can be run either from within MATLAB or compiled as a stand-alone executable.

Model description: system.txt

Central to the methodology here is the creation of the "system.txt" file that is intended to describe different aspects of a system that can be characterized by a coupled set of ODEs. Each line in the file is interpreted by the presence of a model descriptor. This includes the ability to specify parameters (primary system and static secondary parameter), default inputs (bolus and infusion rates), compartmental or state information, and model outputs. While it is possible to simply define differential equations, it is also possible to break a system down into component pieces and characterize it according to these components which include: equilibrium relationships, reaction rates, inter-compartmental transport, and turnover processes. The rates specified by these elements are combined additively by the build script, which creates the input files for multiple software platforms.

Implementation details

The supplemental material contains a 'template' folder that can be used as a starting point to develop a system. Included in this folder is a system_help.txt file that contains all of the

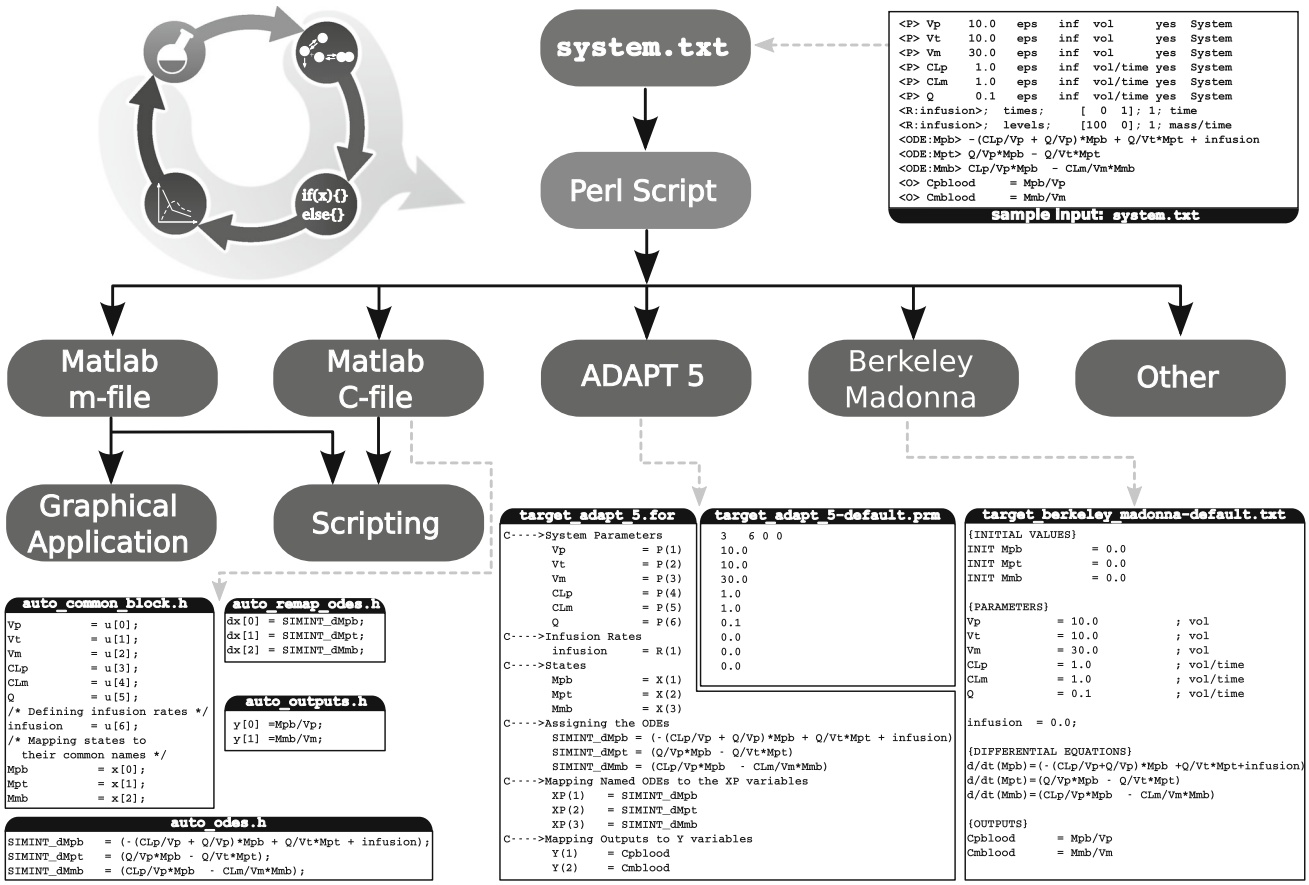


Fig. 1 Model development workflow: A single file describing the system, processed by an included Perl script, produces the same model for several different modeling applications (currently including MATLAB, ADAPT, Berkeley Madonna, MONOLIX, and PottersWheel). The MATLAB files are used by the included GUI allowing non-modelers to examine the model. Annotations indicate sample input (system from Fig. 2a) and generated output for the MATLAB C (auto_common_block.h, auto_odes.h, auto_remap_odes.h, auto_outputs.h), ADAPT (target_adapt_5.for, target_adapt_5-default.prm), and Berkeley Madonna (target_berkeley_madonna-default.txt)

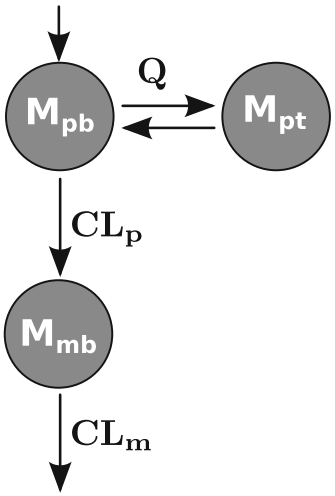
descriptors used in the system file format as well as details of how they are used. The system.txt file can be created directly (starting with system_template.txt) or one of the system.txt files from the case studies discussed below can be copied into this directory.

To build the system, the accompanying Perl script can be run directly (build_system.pl). Upon execution, the Perl script will look for the file system.txt in the current directory and generate several files in the transient subdirectory. These files are intended to serve as input and intermediate files for different analysis software. For example many files with the prefix 'auto' are created, and these are intended to be used within MATLAB to facilitate analysis within that software. Most of these are used internally, but a few are intended to be used directly by the user.

Specifically, the file **auto_fetch_system_information.m** produces a data structure that contains descriptive information about different aspects of the system. This includes default parameter values, mapping information between names and variable indices (e.g. the parameter Vp is the fourth parameter in the parameter vector), and default dosing information. To facilitate running the model at the scripting level, the file **auto_simulation_driver.m** is created with all the components a user needs to execute the model at the command line. Lastly, MATLAB produces simulation output in vector form. This requires the user to remember, for example, that the second output is the metabolite concentration in the serum. To assist in accessing this information and reduce the chance for user error, **auto_map_simulation_outputs.m** can be used to convert the normal simulation output into a data structure with named fields corresponding to the names (states, outputs, and timescales) used in the model. Alternatively, the same files with the 'auto' prefix are called by the GUI to run the model interactively.

While the Perl script can be run directly, it can also be run from analysis software which allows system calls. For example it can be run from within MATLAB using the Perl interpreter that ships with MATLAB (see build_system.m).

a



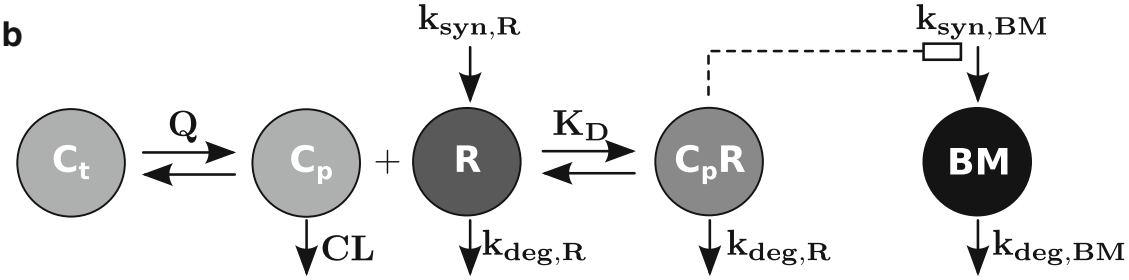
Contents of system.txt

```
#
# parameters
#   name  value  lb    ub   units    editable  type
<P> Vp    10.0   eps   inf  vol      yes       System
<P> Vt    10.0   eps   inf  vol      yes       System
<P> Vm    30.0   eps   inf  vol      yes       System
<P> CLp    1.0   eps   inf  vol/time yes       System
<P> CLm    1.0   eps   inf  vol/time yes       System
<P> Q      0.1   eps   inf  vol/time yes       System

#<R:name>    time/levels  values     scaling   units
#                                    factor
<R:infusion>;  times;     [  0  1];      1;     time
<R:infusion>;  levels;    [100  0];      1;     mass/time

#odes
<ODE:Mpb> -(CLp/Vp + Q/Vp)*Mpb + Q/Vt*Mpt + infusion
<ODE:Mpt> Q/Vp*Mpb - Q/Vt*Mpt
<ODE:Mmb> CLp/Vp*Mpb  - CLm/Vm*Mmb

#outputs
<O> Cpblood      = Mpb/Vp
<O> Cmblood      = Mmb/Vm
```

$$\frac{dM_{pb}}{dt} = -\left(\frac{CL_p}{V_p} + \frac{Q}{V_p}\right) M_{pb} + \frac{Q}{V_t} M_{pt} + \text{infusion} \qquad \frac{dM_{mb}}{dt} = \frac{CL_p}{V_p} M_{pb} - \frac{CL_m}{V_m} M_{mb}$$

$$\frac{dM_{pt}}{dt} = \frac{Q}{V_p} M_{pb} - \frac{Q}{V_t} M_{pt} \qquad C_{pb} = \frac{M_{pb}}{V_p}, \quad C_{mb} = \frac{M_{mb}}{V_m}$$

b



Contents of system.txt

```
# System Parameters
#   name    value    LB     UB      units edit  grouping
<P> CL      0.0129   eps    inf     L/hr  yes   System
<P> Q       0.0329   eps    inf     L/hr  no    System
<P> Vp      3.1      eps    inf     L     yes   System
<P> Vt      2.8      eps    inf     L     yes   System
<P> koff    0.1      eps    inf     1/hr  yes   Drug
<P> KD      0.042    eps    inf     nM    yes   Drug
<P> R_IC   10.0      eps    inf     nM    yes   Target
<P> kdegR  0.01      eps    inf     1/hr  yes   Target
<P> BM_IC   1.0      eps    inf     pct   yes   Biomarker
<P> kdegBM  0.1      eps    inf     1/hr  yes   Biomarker
<P> Imax    4.0      eps    inf     1/hr  yes   Biomarker
<P> IC50    0.2      eps    inf     1/hr  yes   Biomarker
# Secondary Parameters (Static)
<As> ktp     = Q/Vt
<As> kpt     = Q/Vp
<As> kel     = CL/Vp
<As> kon     = koff/KD
<As> ksynR   = kdegR*R_IC*Vp
<As> ksynBM  = kdegBM*BM_IC
```

```
# Secondary Parameters (Dynamic)
<Ad> STIM  =1.0+Imax*Cp_R/(IC50+Cp_R)
# nonzero initial conditions
<I> R   = R_IC
<I> BM  = BM_IC
# bolus inputs
# type      state values     scale  units
<B:times> ;        [ 0];      24*7;  weeks
<B:events>;   Cp; [50];  7.143/Vp;     mg
# Inter-compartmental movement
Cp;  Vp;  kpt  <C> Ct;  Vt;  ktp
# Turnover
  ksynR/Vp    <S:R>   kdegR*R
  ksynBM*STIM <S:BM>  kdegBM*BM
# Equilibrium
Cp + R  <=kon:koff=> Cp_R
<ODE:Cp>    -kel*Cp
<ODE:Cp_R> -kdegR*Cp_R
# outputs
<O> Occupancy         = 1.0 - R/(R+Cp_R)
<O> BM_Stim           = BM
```

◀**Fig. 2** **a** Parent-metabolite model demonstrating the creation of system parameters, infusion inputs, ordinary differential equations describing the evolution of the system in time and model outputs; **b** Target-mediated drug disposition system with complex internalization where the drug/target complex stimulates biomarker synthesis—providing examples of how to define a system in terms of the underlying process (inter-compartmental movement, turnover processes, and equilibrium relationships) as well as how to specify secondary parameters, nonzero initial conditions, and bolus inputs

The purpose here is to streamline the workflow where the system is described completely (including documentation and references) in a single file that automatically generates the components necessary to perform an analysis in the software appropriate for the task at hand. The Perl script also generates multiple files which have a target prefix followed by a text description of the software meant to read that file (e.g. target_monolix-default.txt). These are intended to form the basis for developing workflows around these tools.

The utility of this methodology will be explored through case studies of increasing complexity, each intended to demonstrate different advantages of this framework. For each case study, the indicated folder will contain the system.txt file describing the system, an analysis.m file to run the model within MATLAB, and the components of the template directory need to support execution of the script.

Results

Simple systems

The simplest way of representing a given PK/PD system is through a series of ODEs. In Fig. 2a. A description of a parent drug with two-compartment disposition and metabolite formation is shown (**supplement folder: case_study_odes**). This example, adapted from the ADAPT Users manual [14], contains six system parameters, three states described by ODEs, two observable outputs (serum parent and metabolite levels) and one infusion input. The descriptors needed to characterize this system are shown to the right of the model diagram with comments denoted with the hash (#) symbol.

System parameters are denoted by <*P*> followed by seven columns of information: name, value, lower bound, upper bound, units and a grouping. Next, the rate of infusion is specified by the <R:?> descriptor with information separated by semicolons. The word 'infusion' denotes the name this infusion will have elsewhere in the simulation. Each infusion needs a line indicating the times in which the rate of infusion changes and the level in which it will change. This is referred to as the 'type' of rate information being specified. In this example, the rate of infusion begins at 100 at time zero and the rate switches to zero at time one. This is the default rate but it can be modified through the application or at the scripting level. If the dosing units are different from those in which the system has been parameterized, the 'scale' (set to 1 here) is multiplied by the values supplied by the user in order to obtain the system units. The last column of information contains the units in which the dose is applied. Like bolus inputs described later, this allows dosing information to be specified in units the end-user is likely to be familiar with (e.g., infusing a drug in mg/min) and have them automatically converted into units convenient for modeling (e.g., nM/hr when modeling reaction equilibria). The <ODE:?> descriptor defines the differential equation for a given state. Here the states are Mpb, Mpt, and Mmb, the mass of parent drug in the blood, the mass of the parent drug in the tissue, and the mass of the metabolite in the blood (by default the initial value for all ODEs is zero). Lastly we specify the observable outputs using the <O> descriptor.

Characterizing a system with process-centric descriptions

The system shown in Fig. 2b describes an intravenously injected antibody interacting with a membrane-bound serum target [19] which drives an indirect biomarker response by stimulating the synthesis of a biomarker [20] (**supplement folder: case_study_processes**). This example expands on the previous parent metabolite system by introducing some new concepts. This system begins as that in Fig. 2a, with a listing of the system parameters. The population estimates of parameters for mAb disposition taken from Dirks and Meibohm [21] were used in this example. Next clearances and volumes are converted to first-order rate constants (k_{pt}, k_{tp}, etc.), the second-order rate constant of association (k_{on}) is defined and the steady-state synthesis rate constants ($k_{syn,R}$ and $k_{syn,M}$) are calculated. These are referred to as static secondary parameters (defined with <As>) because they do not change during the course of an individual simulation. Similarly, secondary parameters can also be used to perform variable transformations (i.e., exponentiation). While static secondary parameters do not change during the course of the simulation, it is sometimes necessary to create intermediate values that evolve over time. In this example STIM is defined as a dynamic secondary parameter (<Ad>), and can be defined in terms of states, infusion rates, and previously defined secondary parameters. The <I> descriptor is used to define the nonzero initial conditions, which can be any combination of numeric values, system parameters, or static secondary parameters.

This model also contains dosing information specified by the <B:?> descriptor. The first line specifies the bolus

times in weeks. While the units of time used in the simulation are hours, it may be more convenient from an end user perspective to provide dosing in terms of weeks. The 'scale', allows the end user to convert the time in input units to simulation units. Each compartment or state to receive a dose requires an additional entry with the 'events' keyword. The state name is specified as well as the amount to be injected. Similar to the time entry, the amount can be specified in units different than that of the system. In this case the amount is specified in mg while the state (C_p) is modeled in concentration units. The scale converts the dose in mg into nanomoles and divides by the volume (V_p) resulting in a dose in nM. As with initial conditions, scaling can be done using both system and static secondary parameters.

Model construction is more commonly accomplished by writing ODEs that represent the system. However, it can be advantageous to simply describe the underlying processes which govern the system. The movement between compartments C_p and C_t is characterized using the <C> descriptor. On the left side the concentration in plasma (C_p), the volume of that compartment (V_p) and the rate constant of distribution from plasma (k_{pt}) is specified (delimited by semicolons). To the right of the <C> descriptor the concentration in tissue (C_t), volume of the tissue (V_t) and rate constant of distribution to plasma (k_{tp}) are specified. This entry is then used to construct the relevant components of the ODEs for C_p and C_t.

The turnover of the receptor R is characterized using the <S:?> descriptor. This allows the user to simply list rates of production to the left of the descriptor and rates of loss to the right (multiple values separated by a semicolon). In this case the rate of receptor synthesis ($k_{syn,R}/V_p$) is balanced out by the degradation rate ($k_{deg,R}$•R) at steady-state. Similarly the turnover of the biomarker (BM) was incorporated using the <S:?> descriptor with the synthesis modulated using the previously defined dynamic secondary parameter STIM. Next the equilibrium relationship between the drug and the receptor is described using the <=?:?=> descriptor. This is intended to be written like an equilibrium reaction with the reactants specified to the left and the products on the right separated by plus signs. The forward and reverse rates are then placed in the descriptor. Lastly the systemic elimination of the drug and the target mediated elimination of the complex are added using the <ODE:?> descriptor mentioned previously. When the Perl script builds the system, each of the statements are applied additively.

Model parameterization

Typically models are parameterized for a single situation. However, it can be useful to consider the same system in a different context (different species, healthy versus diseased, etc.). In Fig. 3a, a compartmental system for antibody disposition is shown. In the left portion, a two compartment system describing antibody disposition in humans [21], and to the right the system is parameterized as a one compartment system for mice [22]. This can be accomplished in the same system.txt file using parameter sets (**supplement folder: case_study_parameterizing**).

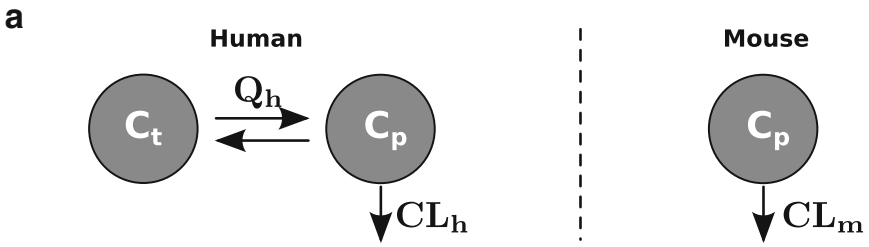
Information about parameter sets is specified using the <PSET:?> descriptor. When system parameters are created using the <*P*> descriptor, these parameters define the default parameter set. To give the default parameter set a more verbose description, the descriptor <PSET:default> is used. Alternatively a second parameter set can be created by simply using a short name to label that parameter set (e.g. <PSET:mouse>). This makes a copy of the default set and then individual parameters can then be overwritten as needed (e.g., <PSET:mouse:Q>).

Parameter set information will manifest in different ways. As will be seen later, it can be selected by the user in the stand-alone application. These parameter sets can be selected explicitly at the scripting level in MATLAB. For the different output formats, a separate file is generated for each parameter set. In this example, the ADAPT output is stored in three files. First there is a Fortran file that contains the ODEs and output definitions (target_adapt_5.for). Next, there are two parameter files that are created (target_adapt_5-default.prm and target_adapt_5-mouse.prm). These contain the parameterizations for the human and mouse with similar files generated for the other output types.

Reducing repetitive tasks with mathematical sets

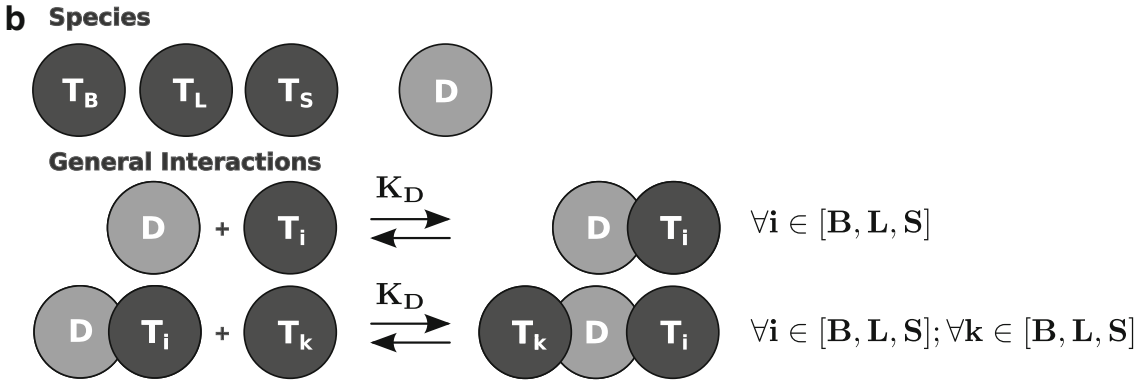
An objective of this framework is to make model development more fluid. One way this has been accomplished is through more natural process-centric descriptions of the system mentioned above. Similarly some systems can become laborious to code due to the repetitive or combinatorial nature of the system. As an example consider the organs of a physiologically based pharmacokinetic model (PBPK). This is essentially the same set of ODES, the mathematical structure of which is identical, repeated for each organ. When modeling interactions such as those in an anti-drug antibody (ADA) assay, there are different drug species, possibly the target, and a distribution of ADAs (both concentration and affinity) participating in many competing equilibrium reactions forming many different complexes. Enumeration of these relationships and writing the ODEs describing these interactions can quickly become intractable. To address these, the ability to create mathematical sets has been incorporated.

A mathematical set example is shown in Fig. 3b, which describes the interactions that can occur in a ligand-binding assay (**supplement folder: case_study_sets**). In this case a



Contents of system.txt (parameters only)

```
#name       value  lower  upper   units  edit  grouping
#                  bound  bound
<P> Weight  70.0   eps    inf     kg     yes   System # Organism weight
<P> CL      0.0129 eps    inf     L/hr   yes   System # Systemic Clearance
<P> Q       0.0329 eps    inf     L/hr   yes   System # Inter-compartmental clearance
<P> Vp      3.1    eps    inf     L      yes   System # Vol. central compartment
<P> Vt      2.8    eps    inf     L      yes   System # Vol. peripheral compartment


<PSET:default>         mAb in Human
<PSET:mouse>           mAb in Mouse
<PSET:mouse:Weight>    0.020   # 20 gram mouse
<PSET:mouse:CL>        7.71e-6
<PSET:mouse:Q>         0.0
<PSET:mouse:Vp>        1.6e-3
<PSET:mouse:Vt>        1       # arbitrary
```



Contents of system.txt

```
# Sets
# The target set is duplicated in order
# to create trimers with two types of
# target present
<SET:TSi>  TL; TB; TS
<SET:TSk>  TL; TB; TS

#name     value lower upper units edit grouping
<P> koff   0.1  eps   inf   1/hr  yes  Interaction
<P> KD     0.1  eps   inf   nM    yes  Interaction
<P> D0     0.1  eps   inf   nM    yes  Drug
<P> T0_TL  0.1  eps   inf   nM    yes  Target
<P> T0_TB  1.0  eps   inf   nM    yes  Target
<P> T0_TS  1.0  eps   inf   nM    yes  Target
```

```
# Here we calculate the kon
<As> kon  = koff/KD
# This defines the initial conditions for
# each drug and target. These values are
# pulled from the parameters above
<I> D     = D0
<I> {TSi} = T0_{TSi}
# drug/receptor equilibrium
# Drug binding to single target:
D       + {TSi}    <=kon:koff=>       D_{TSi}
# Drug/target dimer binding to another target:
D_{TSi} + {TSks}   <=kon:koff=> {TSk}_D_{TSi}
# Molar amount contributing to assay signal
<O> Signal = TL_D_TB + TB_D_TL
```

Fig. 3 **a** A two-compartment model of mAb disposition in humans (*left*) reduced down to a one-compartment model in mice (*right*) in the same system file through parameterization; **b** Modeling the effects of target contamination in a sample (T_S) on assay output using biotinylated target (T_B) and fluorescently labeled target (T_L) as the pull-down and signaling molecules in an assay used to measure drug levels (D) in serum using mathematical sets to enumerate all of the different equilibria that exist in the assay

drug D is being detected by mixing biotinylated target (T_B) with fluorescently labeled target (T_L). Normally at equilibrium, drug D complexes with both T_B and T_L present that would be pulled down and provide a measurable signal. In this case, we want to understand the interference of target present in the sample (T_S) on the signal. This necessarily involves accounting for all of the possible dimer and trimer complexes that can occur.

Mathematical sets are defined using the <SET:?> notation. In this example two identical sets TSi and TSk are defined with the members TL, TB and TS. A set is used by simply referencing it by name within curly braces {} in any of the descriptors defined above. For example, the drug (D0) and target (T0_TS) concentration in the sample and the concentration of the reagents in the assay (T0_TB and T0_TL) are specified as system parameters. These are identified as initial conditions using the <I> descriptor. For the drug this is simply done by specifying D0. For the three different target species this is done with a single entry (<I> {TSi} = T0_{TSi}). This initial condition assignment is expanded internally and repeated for each member of the set TSi. Similarly, the drug with one binding site is also specified using the equilibrium descriptor.

These demonstrate how to use sets to reduce repetitive tasks. Next it is necessary to enumerate all of the different complexes that can form when the drug is fully bound. To do this, an equilibrium descriptor is used which contains both sets (TSi and TSk). When multiple sets are present in a descriptor, every combination is enumerated. It is this combinatorial enumeration that necessitates creation of two identical sets. The system is initialized in terms of the total amount of each monomeric species present. After simulation to steady-state, various complexes will form and the measured output 'Signal' is specified as the total number trimers with both a TB and TL present. By varying the value of T0_TS the sensitivity of the assay to sample impurities can be ascertained.

Using sets requires a certain level of abstraction but this can significantly reduce the amount of coding necessary to represent complex systems. In Fig. 4a single organ is shown for a PBPK model of monoclonal antibody disposition [23] (**supplement folder: case_study_pbpk**). All mass flows entering and volumetric flows leaving an organ have been defined abstractly such that the physiological connectivity is maintained. This allows a single set of ODEs to be written for each compartment within an organ (ORG) in the set notation. These are shown in the lower portion of Fig. 4. These equations are then expanded out to characterize all of the organs. While the diagram contains a portion of the system descriptors, the system.txt file contains all of the elements required to implement this model (including examples of performing summations across sets).

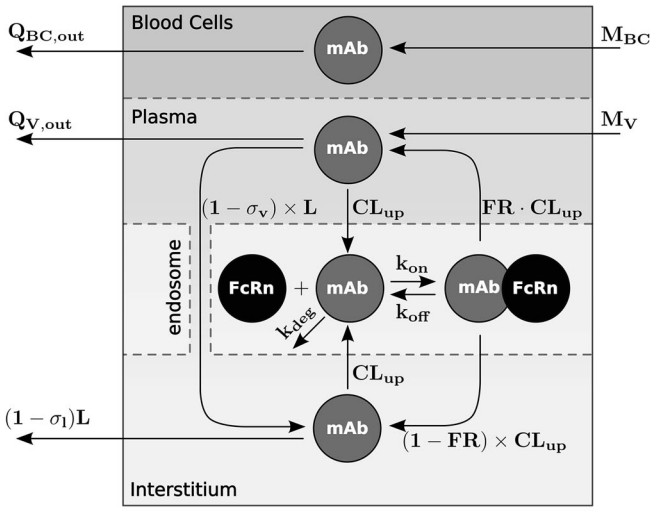
Standalone application

Each model created using this framework can be accessed using different target applications previously mentioned. To further extend the utility of the framework, a deployable application is provided to allow basic access to the system. The application with the TMDD system from Fig. 2b running is shown in Fig. 5. The different parameter sets in the model are available through a pull-down menu, and by selecting a set it will update the values in the table below. The system file is used to determine which parameters can be seen/modified and how they are grouped. Basic dosing information is also picked up from the system file. If a loading dose is desired, this can be described in detail and chronic dosing can be achieved by repeating the last dose at a specified interval. General infusion information can also be modified from within the GUI as well. Placeholders for both input types must be created in the system file in order for them to be modifiable at the application level.

Basic plot controls have been implemented to allow the user to explore the system behavior. This can allow the user to examine different dosing scenarios, identify how modifying parameters can impact the system, and compare different model outputs. This tool is intended to be a model exploration tool and to provide immediate graphical output. If the user prefers to create figures in other software, there is a comma separated variable (CSV) export option as well. This option creates three time-stamped files. The first is the CSV file containing columns for different timescales, each of the states in the model, as well as all of the model outputs. A screenshot of the interface is also created as well as a text file with the current parameter values. These last two files are intended to provide the user with the context surrounding the CSV export.

To inform the end user about the model structure, bounds on parameters, inherent assumptions, etc., it is possible to embed a model diagram containing this information. This is done by saving the information in a figure called system.jpg in the same directory as the system.txt file, and it will be visible through the 'View Model' button in the application. By default, the application runs the model according to the values specified in the GUI. It may be desirable to modify components of the system internally based on the user input. For example, it may be more ideal to have a user specify a mean and standard deviation information as input parameters and to sample from the resulting distribution before running the simulation. This can be accomplished using call back routines (see system_help.txt for more information).

While a model can be run from the GUI directly from within MATLAB, it is also possible to create a deployable application. The template contains a file called **build_exe.m**, which will compile the GUI into an executable.



Flow Abstraction

Volumetric Flow calculations:

From Lung; Organs upstream of current organ

$$\begin{aligned} Q_{BC,tot} &= Q_{BC} + \sum_{US} Q_{BC,US} \\ Q_{V,tot} &= Q_V + \sum_{US} Q_{V,US} - L_{US} \\ Q_{BC,out} &= Q_{BC,tot} \\ Q_{V,out} &= Q_{V,tot} - L \end{aligned}$$

Mass Flow calculations:

For all organs except the lung:

$$\begin{aligned} M_V &= C_{V,LUNG}Q_V + \sum_{US} C_{V,US}(Q_{V,US} - L_{US}) \\ M_{BC} &= C_{BC,LUNG}Q_{BC} + \sum_{US} C_{BC,US}(Q_{BC,US}) \end{aligned}$$

For the lung

$$\begin{aligned} M_V &= (Q_V)C_{PLASMA} \\ M_{BC} &= (Q_{BC})C_{BC} \end{aligned}$$

Relevant portion of system.txt file

```
# defining the organ set
<SET:ORG>  HEART;  LUNG;  MUSCLE;  SKIN; ADIPOSE; BONE; BRAIN; KIDNEY; LIVER; SM_INT; LG_INT; PANCREAS; THYMUS; SPLEEN; OTHER

#  Plasma (Vascular Space)
#                Vascular     Vascular Mass            Loss to                                Pinocytosis             Pinocytosis
#                Mass In      Leaving                  Interstitium                           Uptake                  Return
<ODE:C_V_{ORG}> (M_VI_{ORG} - Q_VOUT_{ORG}*C_V_{ORG} - (1.0-SIGMA_V_{ORG})*L_{ORG}*C_V_{ORG} - CL_UP_{ORG}*C_V_{ORG} + CL_UP_{ORG}*FR*C_E_B_{ORG})/V_V_{ORG}
#  Blood Cells (Vascular Space)
<ODE:C_BC_{ORG}> MT_BC_{ORG}/V_BC_{ORG}

#
# Endosomaal space
#                Endosomal Uptake                                FcRn Binding                    FcRn                    Degradation
# Free Drug                                                      Association                     Disassociation
<ODE:C_E_UB_{ORG}>  (C_V_{ORG} + C_I_{ORG})*CL_UP_{ORG}/V_E_{ORG} - K_on_FCRN*C_E_UB_{ORG}*FCRN_{ORG}  + K_off_FCRN*C_E_B_{ORG} - K_deg*C_E_UB_{ORG}
# Free FCRN
<ODE:FCRN_{ORG}>                C_E_B_{ORG}*CL_UP_{ORG}/V_E_{ORG} - K_on_FCRN*C_E_UB_{ORG}*FCRN_{ORG}  + K_off_FCRN*C_E_B_{ORG}
# FCRN Bound
<ODE:C_E_B_{ORG}>             - C_E_B_{ORG}*CL_UP_{ORG}/V_E_{ORG} + K_on_FCRN*C_E_UB_{ORG}*FCRN_{ORG}  - K_off_FCRN*C_E_B_{ORG}

#
# Interstitial Space
#
<ODE:C_I_{ORG}>   (1.0-SIGMA_V_{ORG})*L_{ORG}*C_V_{ORG}/V_I_{ORG}
<ODE:C_I_{ORG}> - (1.0-SIGMA_I_{ORG})*L_{ORG}*C_I_{ORG}/V_I_{ORG}
<ODE:C_I_{ORG}> - CL_UP_{ORG}*C_I_{ORG}/V_I_{ORG}
<ODE:C_I_{ORG}> + CL_UP_{ORG}*(1.0-FR)*C_E_B_{ORG}/V_I_{ORG}
```

Fig. 4 Organ component of a PBPK model of mAb disposition demonstrating the utility of using mathematical sets to define components of a system. The mass flows entering the organ and volumetric flows leaving the organ are defined using the equations (*right*) with the ODEs for all organs in the system written using the set notation (*bottom*)

In order to deploy the resulting executable, it is necessary for the end user to install the MATLAB Compiler Runtime (MCR) (available within the MATLAB distribution). Both the compiled application and the MCR can be distributed royalty free.

Discussion

Currently there are a number of modeling tools available to inform decisions with regard to drug discovery and development. While some software provides utility in terms of ease of use, other packages are valued for the granular control they provide. These applications are further differentiated in terms of the type of analyses they support (simulation, estimation, sensitivity analysis, etc.). In a heterogeneous environment where people of many backgrounds use a diverse set of tools, interoperability can become quite challenging. The workflow and tools presented here represent a step towards addressing this issue.

The text-based format introduced here is intended to provide the modeling and simulation scientist with a flexible set of methods for describing a physiological system in mathematical terms and creating the source files needed to take advantage of several tools and/or software platforms. Handing over the underlying model to different users with different software preferences has been challenging and extremely time consuming. Adoption of this tool will now make this task easier and avoid transcription errors when implementation of the model is required across different software platforms. Building on concepts from the text based structure of NONMEM [24] control files, it is possible to construct a system by simply entering the system parameters, secondary parameters, and ODEs. Alternatively

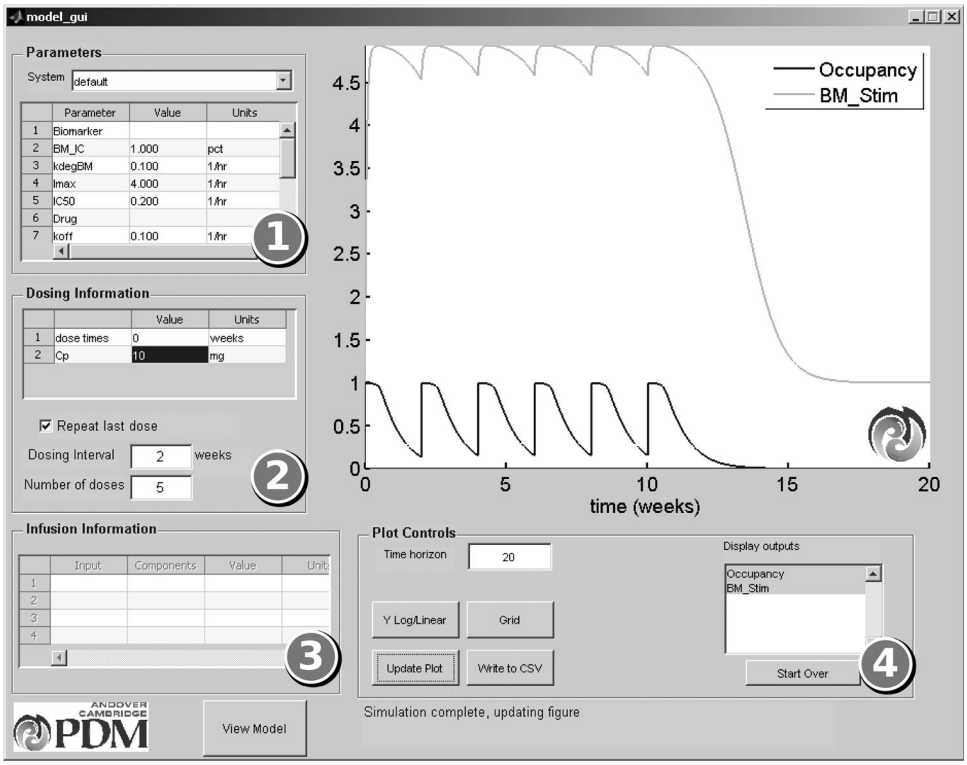


Fig. 5 Stand alone application allowing users with no programming experience to utilize any model developed using this framework: (*1*) Different parameter sets can be selected using the pull-down menu, repopulating the parameter table with the subset of parameters specified as editable; (*2*) loading doses can be entered (a dose value for each time specified) and the last dose can be repeated in order to simulate chronic dosing schedules; (*3*) infusions can also be modified by the user; (*4*) basic plotting controls are available to specify the desired outputs to be shown, *y*-axis scale, etc. while users can have more control by using the CSV export option

it is possible to simply describe the underlying processes that govern the system. By using mathematical sets, coding intractable systems becomes manageable and complex systems are easier to implement. Special care has been taken to ensure that the resultant output files, while crafted in an automated fashion, are readable to facilitate modification if necessary. This is aimed at increasing the number of end users who have an interest from a model creation standpoint. The deployable application provides a less intimidating way to explore models within the project team setting as well as allowing a user with no programming skills to inspect the system.

The examples provided here are intended to be introductory in nature. Features not mentioned explicitly, for example the creation of piecewise continuous functions using if/then statements, are possible. As with all other features, these are outlined in detail the system_help.txt file. However, given the objective of making the same system available in several pieces of software, there are limitations that are implicit. Characteristics that are integral to a given system and not widely implemented across software will be difficult to implement in this format. For example a system dependent on delayed differential equations is intrinsically linked to this feature, and are not currently implementable in the current framework. As such, the current framework is limited to defining systems that can be described by a set of coupled ODEs.

As outlined above, several different output formats are currently supported. Extension to support other software which take text input is straightforward, and the benefit being that all models previously developed in this framework are instantly available on the newly supported software. Case studies discussed above were generally limited to the structural model, but variance parameters and error models can be specified and are included in relevant files (see ADAPT input files). Keeping with the concept of having a canonical input file, the current plan is to more fully include support for population components as well. This information will then be included in corresponding outputs types (currently Monolix and ADAPT). Furthermore, based on interest the model and support files for other software will be added as well.

Acknowledgments We would like to express our appreciation to Indranil Bhattacharya, Itrat Harrold, Ryan Nolan, Robert Parker, and Yulia Vugmeyster for critical review of the manuscript.

References

1. Lipscomb JC, Haddad S, Poet T, Krishnan K (2012) Physiologically-based pharmacokinetic (PBPK) models in toxicity testing and risk assessment. In: Johanson G (ed) In technologies for toxicity. Springer, New York, pp 76–95
2. Abuqayyas L, Balthasar JP (2012) Application of knockout mouse models to investigate the influence of FcγR on the tissue distribution

and elimination of 8C2, a murine IgG1 monoclonal antibody. Int J Pharm 439:8–16. doi:10.1016/j.ijpharm.2012.09.042
3. Jones HM, Dickins M, Youdim K, Gosset JR, Attkins NJ, Hay TL et al (2012) Application of PBPK modelling in drug discovery and development at Pfizer. Xenobiotica 42:94–106. doi:10.3109/00498254.2011.627477
4. Iyengar R, Zhao S, Chung S-W, Mager DE, Gallo JM (2012) Merging systems biology with pharmacodynamics. Sci Transl Med 4:126–227. doi:10.1126/scitranslmed.3003563
5. Luu KT, Kraynov E, Kuang B, Vicini P, Zhong W-Z (2013) Modeling, simulation, and translation framework for the preclinical development of monoclonal antibodies. AAPS J 15:551–558. doi:10.1208/s12248-013-9464-8
6. Chien JY, Friedrich S, Heathman MA, Alwis DP, Sinha V (2005) Pharmacokinetics/pharmacodynamics and the stages of drug development: role of modeling and simulation. AAPS J 7:544–559. doi:10.1208/aapsj070355
7. Lavé T, Parrott N, Grimm HP, Fleury A, Reddy M (2007) Challenges and opportunities with modelling and simulation in drug discovery and drug development. Xenobiotica 37:1295–1310. doi:10.1080/00498250701534885
8. Rosenthal RE (2006) GAMS—a user's guide. GAMS Development Corporation, Washington, DC
9. Harrold JM, Parker RS (2009) Clinically relevant cancer chemotherapy dose scheduling via mixed-integer optimization. Comput Chem Eng 33:2042–2054. doi:10.1016/j
10. Fritzson P (2011) Introduction to modeling and simulation of technical and physical systems with modelica. Wiley, Hoboken
11. Hucka, M (2010) The systems biology markup language (SBML): language specification for level 3 version 1 core. http://sbml.org/Documents/Specifications. Accessed 24 November 3013
12. Moodie, S L, Swat, M J, Kristensen, N R, Le Novère, N (2013). PharmML: the pharmacometrics markup language. Drug disease model resource. http://www.ddmore.eu/pharmml. Accessed 24 November 2013
13. Holford, N (2013). *Model Coding Language Specification Draft 5 version 0.8*. Drug Disease Model Resource. http://www.ddmore.eu/mdl. Accessed 24 November 2013
14. D'Argenio DZ, Schumitzky A, Wang X (2009) ADAPT 5 user's guide: pharmacokinetic/pharmacodynamic systems analysis software. Biomedical simulations resource. University of Southern California, Los Angeles
15. Macey R, Oster G, Zahnley T (2000) Berkeley Madonna user's guide, Department of Molecular and Cellular Biology. University of California, Berkeley
16. Maiwald T, Timmer J (2008) Dynamical modeling and multi-experiment fitting with PottersWheel. Bioinformatics 24: 2037–2043. doi:10.1093/bioinformatics/btn350
17. Lavielle M, Mentré F (2007) Estimation of population pharmacokinetic parameters of saquinavir in HIV patients with the MONOLIX software. J Pharmacokinet Pharmacodyn 34:229–249. doi:10.1007/s10928-006-9043-z
18. Chan PLS, Jacqmin P, Lavielle M, McFadyen L, Weatherley B (2010) The use of the SAEM algorithm in MONOLIX software for estimation of population pharmacokinetic-pharmacodynamic-viral dynamics parameters of maraviroc in asymptomatic HIV subjects. J Pharmacokinet Biopharm 38:41–61. doi:10.1007/s10928-010-9175-z
19. Mager DE (2006) Target-mediated drug disposition and dynamics. Biochem Pharmacol 72:1–10. doi:10.1016/j.bcp.2005.12.041
20. Dayneka NL, Garg V, Jusko WJ (1993) Comparison of four basic models of indirect pharmacodynamic responses. J Pharmacokinet Biopharm 21:457–478. doi:10.1007/BF01061691
21. Dirks NL, Meibohm B (2010) Population pharmacokinetics of therapeutic monoclonal antibodies. Clin Pharmacokinet 49:633–659. doi:10.2165/11535960-000000000-00000
22. Vieira P, Rajewsky K (1988) The half-lives of serum immunoglobulins in adult mice. Eur J Immunol 18:313–316. doi:10.1002/eji.1830180221
23. Shah DK, Betts AM (2012) Towards a platform PBPK model to characterize the plasma and tissue disposition of monoclonal antibodies in preclinical species and human. J Pharmacokinet Biopharm 39:67–86. doi:10.1007/s10928-011-9232-2
24. Beal SL, Sheiner LB, Boeckmann A, Bauer RJ (2009) NONMEM user's guides. (1989–2009). Icon Development Solutions, Ellicott City