Chapter 3

Modeling Pharmacokinetics

Frederic Y. Bois and Céline Brochot

Abstract

Pharmacokinetics is the study of the fate of xenobiotics in a living organism. Physiologically based pharmacokinetic (PBPK) models provide realistic descriptions of xenobiotics' absorption, distribution, metabolism, and excretion processes. They model the body as a set of homogeneous compartments representing organs, and their parameters refer to anatomical, physiological, biochemical, and physicochemical entities. They offer a quantitative mechanistic framework to understand and simulate the time-course of the concentration of a substance in various organs and body fluids. These models are well suited for performing extrapolations inherent to toxicology and pharmacology (e.g., between species or doses) and for integrating data obtained from various sources (e.g., in vitro or in vivo experiments, structure–activity models). In this chapter, we describe the practical development and basic use of a PBPK model from model building to model simulations, through implementation with an easily accessible free software.

Key words 1,3-Butadiene, PBPK, Monte Carlo simulations, Numerical integration, *R* software

1 Introduction

The therapeutic or toxic effects of chemical substances not only depend on interactions with biomolecules at the cellular level, but also on the amount of the active substance reaching target cells (i.e., where the effects arise). Therefore, conceptually, two phases can be distinguished in the time course of such effects: the absorption, transport, and elimination of substances into, in, and out of the body including target tissues (pharmacokinetics), and their action on these targets (pharmacodynamics). Schematically, pharmacokinetics (or toxicokinetics for toxic molecules) can be defined as the action of the body on substances, and pharmacodynamics as the action of substances on the body. Pharmacokinetic and pharmacodynamics first aim at a qualitative understanding of the underlying biology. They also use mathematical models to analyze and extrapolate measurements of various biomarkers of exposure, susceptibility or effect, in order to quantitatively predict effects. This chapter focuses on toxicokinetic models and in particular on physiologically based pharmacokinetic (PBPK) models.

Emilio Benfenati (ed.), *In Silico Methods for Predicting Drug Toxicity*, Methods in Molecular Biology, vol. 1425,
DOI 10.1007/978-1-4939-3609-0_3,

Toxicokinetic models aim to link an external exposure to an internal dosimetry in humans (e.g., concentration in blood, urine, or in tissues) by describing the process of absorption, distribution, metabolism, and excretion (ADME) that undergoes a substance in living organisms. A class of toxicokinetic models, the physiologically based pharmacokinetic (PBPK) models, bases the description on the ADME processes on the physiology and the anatomy of individuals, and the biochemistry of the compounds. A PBPK model subdivides the body in compartments representing organs connected through a fluid, usually blood. Model parameters correspond to physiological and biochemical entities specific to the body and compounds, such as organ volumes, tissue blood flows, affinities of the compounds for the tissues, or the metabolic clearance.

The first works in pharmacokinetic modeling were based on physiological descriptions of the body [1–6]. However, at the time, the corresponding mathematical models were too complex to be solved. Research and applications then focused on simpler one-, two-, or three-compartment models [7], which proved to be adequate for describing and interpolating concentration–time profiles of many drugs in blood or other biological matrices. However, for substances with complex kinetics, or when inter-species extrapolations were required, simple models were insufficient and research continued on physiological models [8–12].

Over the years, the ever-increasing computing capabilities and the advent of statistical approaches applicable to uncertainty and population variability modeling have turned PBPK models into well-developed tools for safety assessment of chemical substances [13]. A significant advance has been the development of quantitative structure–properties models for the chemical-dependent parameters of PBPK models (e.g., tissue affinities) [14, 15]. Those developments are still ongoing and have led to large generic models which can give quick, even if approximate, answers to pharmacokinetic questions, solely on the basis of a chemical's formula and limited data [16–18].

The mechanistic basis of PBPK models is particularly well adapted to toxicological risk assessment [19, 20] and also in the pharmaceutical industry for the development of new therapeutic substances [21], in particular for dealing with extrapolations inherent to these domains (in vitro to in vivo, laboratory animals to human populations, various exposure or dosing schemes, etc.). PBPK models can be applied in two different steps of the risk assessment framework. First, these models can be used to better characterize the relationship between the exposure dose and the adverse effects by modeling the internal exposure in the target tissues (i.e., where the toxic effects arise) [22]. Secondly, PBPK models can be used in the exposure assessment to estimate the external exposure using human biomonitoring data, like the concentrations of chemicals in blood or urine [23, 24]. These predictions can then

be compared to existing exposure guidance or reference values such as tolerable daily intakes [25].

To provide a general overview of the basis and applications of PBPK modeling, the first section of this chapter describes the development of a PBPK model (model formulation, parameter estimation). We then propose to illustrate the different steps with 1,3-butadiene, a volatile organic compound that is carcinogenic to humans (group 1 in the IARC classification).

2 Development of a PBPK Model

In this section, we present the steps to follow in developing a PBPK model. Recently, the International Programme on Chemical Safety provided guidance on the characterization and application of PBPK models in risk assessment [20]. The guidance aimed to propose a standardized framework to review and critically evaluate the available toxicological data, and describe thoroughly the development of the model, i.e., structure, equations, parameter estimation, model evaluation, and validation. The ICRP framework also aimed to harmonize good modeling practices between risk assessors and model developers [26–28].

2.1 Principles and Model Equations

A PBPK model represents the organism of interest—human, rat, mouse, etc.—as a set of compartments, each corresponding to an organ, group of organs or tissues (e.g., adipose tissue, bone, brain, gut) having similar blood perfusion rate (or permeability) and affinity for the substance of interest. Transport of molecules between those compartments by blood, lymph, or diffusion, and further absorption, distribution, metabolism, or excretion (ADME) processes are described by mathematical equations (formally differential equations) whose structure is governed by physiology (e.g., blood flow in exit of gut goes to liver) [29, 30]. As such, PBPK modeling is an integrated approach to understand and predict the pharmacokinetic behavior of chemical substances in the body.

Drug distribution into a tissue can be rate-limited by either perfusion or permeability. Perfusion-rate-limited kinetics apply when the tissue membranes present no barrier to diffusion. Blood flow, assuming that the drug is transported mainly by blood, as is often the case, is then the limiting factor to distribution in the various cells of the body. That is usually true for small lipophilic drugs. A simple perfusion-limited PBPK model is depicted in Fig. 1. It includes the liver, well-perfused tissues (lumping brain, kidneys, and other viscera), poorly perfused tissues (muscles and skin), and fat. The organs have been grouped into those compartments under the criteria of blood perfusion rate and lipid content. Under such criteria, the liver should be lumped with the well-perfused tissues, but is left separate here as it is supposed to be the site of

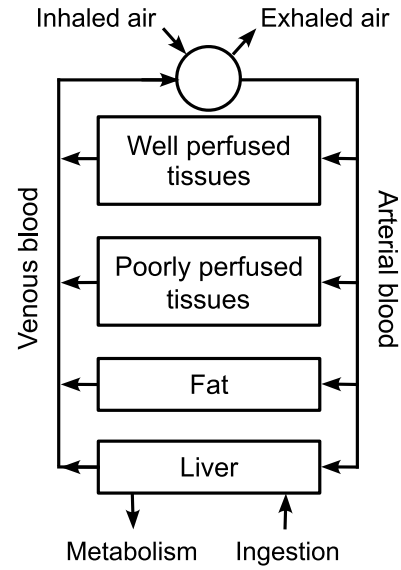

Fig. 1 Schematic representation of a simple, perfusion-limited, PBPK model. The model equations are detailed in Subheading 2 of the text

metabolism, a target effect site, and a port of entry for oral absorption (assuming that the gut is a passive absorption site which feeds into the liver via the portal vein). Bone can be excluded from the model if the substance of interest does not distribute to it. The substance is brought to each of these compartments via arterial blood. Under perfusion limitation, the instantaneous rate of entry for the quantity of drug in a compartment is simply equal to the (blood) volumetric flow rate through the organ times the incoming blood concentration. At the organ exit, the substance's venous blood concentration is assumed to be in equilibrium with the compartment concentration, with an equilibrium ratio named "partition coefficient" or "affinity constant" [30]. In the following we will note Qi the quantity of substance in compartment i, Ci the corresponding concentration, Vi the volume of compartment i, Fi the blood flow to that compartment, and PCi the corresponding tissue over blood partition coefficient. Note that all differentials are written for quantities, rather than concentrations because molecules are transported. Arguably, they are proportional to differentials for concentrations, but only if volumes are constant (and they may not be). For consistency, we strongly suggest you work with quantities. The rate of change of the quantity of substance in the poorly perfused compartment, for example, can therefore be described by the following differential equation:

$$\frac{\partial Q_{\mathrm{pp}}}{\partial t} = F_{\mathrm{pp}} \times \left(C_{\mathrm{art}} - \frac{Q_{\mathrm{pp}}}{P_{\mathrm{pp}} V_{\mathrm{pp}}} \right) \tag{1}$$

where Q_{pp} is the quantity of substance at any given time in the poorly perfused compartment, F_{pp} the blood volumetric flow rate through that group of organs, C_{art} the substance's arterial blood concentration, P_{pp} the poorly perfused tissues over blood partition coefficient, and V_{pp} the volume of the poorly perfused compartment. Since Q_{pp} kinetics are governed by a differential equation, it is part of the so-called "state variables" of the model. The tissue over blood partition coefficient P_{pp} measures the relative affinity of the substance for the tissue compared to blood. It is easy to check that, at equilibrium,

$$\frac{\partial Q_{pp}}{\partial t} = 0 \Rightarrow C_{art} - \frac{Q_{pp}}{P_{pp} V_{pp}} = 0 \Rightarrow P_{pp} = \frac{C_{pp}}{C_{art}} \tag{2}$$

if we denote by C_{pp} the concentration of the substance in the poorly perfused compartment. Similarly, for the well-perfused and the fat compartments we can write the following equations for the two state variables Q_{wp}, and Q_{fat}, respectively:

$$\frac{\partial Q_{wp}}{\partial t} = F_{wp} \times \left(C_{art} - \frac{Q_{wp}}{P_{wp} V_{wp}} \right) \tag{3}$$

$$\frac{\partial Q_{fat}}{\partial t} = F_{fat} \times \left(C_{fat} - \frac{Q_{fat}}{P_{fat} V_{fat}} \right) \tag{4}$$

The equation for the last state variable, Q_{liv} (for the liver) is a bit more complex, with a term for metabolic clearance, with first-order rate constant k_{met}, and a term corresponding to the oral ingestion rate of the compound (quantity absorbed per unit time), R_{ing} which corresponds to the administration rate if gut absorption is complete, or to a fraction of it otherwise:

$$\frac{\partial Q_{liv}}{\partial t} = F_{liv} \left(C_{art} - \frac{Q_{liv}}{P_{liv} V_{liv}} \right) - k_{met} Q_{liv} + R_{ing} \tag{5}$$

Obviously, this is a minimal model for metabolism, and much more complex terms may be used for saturable metabolism, binding to blood proteins, multiple enzymes, metabolic interactions, extrahepatic metabolism, etc. If the substance is volatile, and if accumulation in the lung tissue itself is neglected, the arterial blood concentration C_{art} can be computed as follows, assuming instantaneous equilibrium between blood and air in the lung:

$$C_{art} = \frac{F_{pul} \left(1 - r_{ds}\right) C_{inh} + F_{tot} C_{ven}}{F_{pul} \left(1 - r_{ds}\right) / P_a + F_{tot}} \tag{6}$$

where F_{tot} is the blood flow to the lung, F_{pul} the pulmonary ventilation rate, r_{ds} the fraction of dead space (upper airways' volume unavailable for blood-air exchange) in the lung, P_a the blood over

air partition coefficient, and C_{inh} is the concentration inhaled. Equation 6 can be derived from a simple balance of mass exchanges between blood and air under equilibrium conditions. C_{ven} is the concentration of compound in venous blood and can be obtained as the sum of compound concentrations in venous blood at the organ exits weighted by corresponding blood flows:

$$C_{ven} = \frac{\sum_{x \in \{pp,wp,fat,liv\}} \left(\frac{F_x Q_x}{P_x V_x} \right)}{F_{pp} + F_{wp} + F_{fat} + F_{liv}} \tag{7}$$

Finally, the substance's concentration in exhaled air, C_{exh}, can be obtained under the same equilibrium conditions as for Eq. 6:

$$C_{exh} = (1 - r_{ds}) \frac{C_{art}}{P_a} + r_{ds} C_{inh} \tag{8}$$

Note that C_{art}, C_{ven}, and C_{exh}, are not specified by differential equations, but by algebraic equations. Those three variables are not fundamental in our model and could be expressed using only parameters and state variables. They are just (very) convenient "output variables" that we may want to record during simulation and that facilitate model writing.

The above model assumes that all the substance present in blood is available for exchange with tissues. This may not be true if a fraction of the substance is bound, for example to proteins, in blood or tissues. In that case it is often assumed that binding/unbinding is rapid compared to the other processes. The equations are then written in terms of unbound quantities and the rapid equilibrium assumption is used to keep track of the balance bound/unbound quantity in each organ or tissue [30].

Diffusion across vascular barriers or cellular membranes can be slower than perfusion. This condition is likely to be met by large polar molecules. In that case, to account for diffusion limitation, a vascular sub-compartment is usually added to each organ or tissue of interest. Diffusion between that vascular sub-compartment and the rest of the tissue is modeled using the Fick's law. A diffusion barrier can also exist between the extracellular and intracellular compartments. Consequently, PBPK models exhibit very different degrees of complexity, depending on the number of compartments used and their eventual subdivisions [31].

2.2 Parameter Estimation

A PBPK model needs a considerable amount of information to parameterize. At the system level, we find substance-independent anatomical (e.g., organ volume), physiological (e.g., cardiac output), and some biochemical parameters (e.g., enzyme concentrations). All those are generic, in the sense that they do not depend on the substance(s) of interest, and are relatively well documented in

humans and laboratory animals [29, 32–36]. They can be assigned once for ever, at least in first approximation, for an "average" individual in a given species at a given time.

There are also, inevitably, substance-specific parameters which reflect the specific interactions between the body and the substance of interest. In many cases, values for those parameters are not readily available. However, such parameters often depend, at least in part, on the physicochemical characteristics of molecule studied (e.g., partition coefficients depend on lipophilicity, passive renal clearance depends on molecular weight). In that case, they can be estimated, for example by quantitative structure–activity relationships (QSARs) [37, 38], also referred to as quantitative structure–property relationships (QSPRs) when "simple" parameter values are predicted. Molecular simulation (quantum chemistry) models can also be used [39, 40], in particular for the difficult problem of metabolic parameters' estimation. QSARs are statistical models (often a regression) relating one or more parameters describing chemical structure (predictors) to a quantitative measure of a property or activity (here a parameter value in a PBPK model) [15, 41–44]. However, when predictive structure–property models are not available (as is often the case with metabolism, for example), the parameters have to be measured in vitro (for an extensive review *see* [45, 46]) or estimated from in vivo experiments and are much more difficult to obtain.

However, using average parameter values does not correctly reflect the range of responses expected in a human population, nor the uncertainty about the value obtained by QSARs, in vitro experiments or in vivo estimation [47]. Inter-individual variability in PK can have direct consequences on efficacy and toxicity, especially for substances with a narrow therapeutic window. Therefore, simulation of inter-individual variability should be an integral part of the prediction of PK in humans. The mechanistic framework of PBPK models provides the capacity of predicting inter-individual variability in PK when the required information is adequately incorporated. To that effect, two modeling strategies have been developed in parallel: The first approach has been mostly used for data-rich substances. It couples a pharmacokinetic model to describe time-series measurements at the individual level and a multilevel (random effect) statistical model to extract a posteriori estimates of variability from a group of subjects [48, 49]. In a Bayesian context, a PBPK model can be used at the individual level, and allows easy inclusion of many subject-specific covariates [50]. The second approach also takes advantage of the predictive capacity of PBPK models but simply assigns a priori distributions to the model parameters (e.g., metabolic parameters, blood flows, organ volumes) and forms distributions of model predictions by Monte Carlo simulations [51].

2.3 Solving the Model Equations

Many software programs can actually be used to build and simulate a PBPK model. Some are very general simulation platforms—*R* [52], GNU MCSim [53, 54], Octave [55], Scilab [56], Matlab® [57], Mathematica® [58], to name a few. Those platforms usually propose some PBPK-specific packages or functionalities that ease model development. An alternative is to use specialized software (e.g., PK-Sim® [59], Simcyp® [60], GastroPlus® [61], Merlin-expo [62]), which has often an attractive interface. However, in that case the model equations cannot usually be modified and only the parameter values can be changed or assigned pre-set values or distributions.

2.4 Evaluation of the Model

The evaluation (checking) of the model is an integral part of its development to objectively demonstrate the reliability and relevance of the model. Model evaluation is often associated with a defined purpose, such as a measure of internal dosimetry relevant to the mode of action of the substance (e.g., the area under the curve or maximal concentration in the target tissues during critical time windows). The objective here is to establish confidence in the predictive capabilities of the model for a few key variables. A common way to evaluate a model's predictability is to confront its predictions to an independent data set, i.e., that has not been used for model development. That is called cross-validation in statistical jargon. For example, the evaluation step could check that the model is able to reproduce the peaks and troughs of tissue concentrations under repeated exposure scenarios. Model evaluation is not limited to a confrontation between model predictions and data, but also requires checking the plausibility of the model structure, its parameterization and the mathematical correctness of equations (e.g., the conservation of mass, organ volumes, and blood flows). Because of their mechanistic description of ADME processes, PBPK model structures and parameter values must be in accordance with biological reality. Parameter values inconsistent with physiological and biological knowledge limit the use of the model for extrapolation to other exposure scenarios, and ultimately need to be corrected by the acquisition of new data, for example.

2.5 Model Validation and Validity Domain

Most models are valid only on a defined domain. That is true even for the most fundamental models in physics. The term "validation" is rarely used in the context of toxicokinetic modeling as it is almost impossible to validate in all generality a model of the whole body. Actually, it is not done because it is bound to fail. It would require experimental data for all state variables (time evolution of concentration in all compartments) and model parameters under innumerable exposure scenarios. In that context, to be useful, the validation process should first define a validity domain. For example, we should not expect PBPK models to give accurate descriptions of within-organ differences in concentrations (organs are

described as homogeneous "boxes"). There is actually an avenue of research for improved organ descriptions. As far as time scale is concerned, we are doing pretty well for long-term [17], but for inhalation at the lung level in particular, PBPK models are not suitable for time scales lower than a couple of minutes (the cyclicity of breathing is not described). Metabolism and the description of metabolites distribution is a deeper problem, as it branches on the open-ended field of systems biology [63]. In that area the domain of validity becomes harder to define and is usually much smaller than that of the parent molecule. The model's domain of validity should be documented, to the extent possible, and even more carefully as we venture into original and exotic applications. Fortunately, the assumptions consciously made during model development usually help in delineating the domain of validity.

3 A PBPK Model for 1,3-Butadiene

In this section, we propose to apply the model development process presented above to the development of a PBPK model for 1,3-butadiene, a volatile organic compound. First, some background information on 1,3-butadiene will be provided to fulfill some requirements of the guidance defined by the International Programme on Chemical Safety [20]. Because the aim here is not to run a risk assessment on butadiene, most sections of the guidance will be omitted (e.g., the comparison with the default approaches).

3.1 Setting Up Background

An extensive literature exists on 1,3-butadiene human uses, exposures, toxicokinetics, and mode of action, *see* for example [64, 65].

1,3-Butadiene (CAS No. 106-99-0) is a colorless gas under normal conditions. It is used for production of synthetic rubber, thermoplastic resins and other plastics, and is also found in cigarette smoke and combustion engine fumes. It enters the environment from engine exhaust emissions, biomass combustion, and from industrial on-site uses. The highest atmospheric concentrations have been measured in cities and close to industrial sources. The general population is exposed to 1,3-butadiene primarily through ambient and indoor air. Tobacco smoke may contribute significant amounts of 1,3-butadiene at the individual level. It is a known carcinogen, acting through its metabolites [65].

1,3-Butadiene metabolism is a complex series of oxidation and reduction steps [65]. Briefly, the first step in the metabolic conversion of butadiene is the cytochrome P450-mediated oxidation to 1,2-epoxy-3-butene (EB). EB may subsequently be exhaled, conjugated with glutathione, further oxidized to 1,2:3,4-diepoxybutane (DEB), or hydrolyzed to 3-butene-1,2-diol (BDD). DEB can then be hydrolyzed to 3,4-epoxy-1,2-butanediol (EBD) or conjugated with glutathione. BDD can be further oxidized to EBD.

EBD can be hydrolyzed or conjugated with glutathione. The metabolism for 1,3-butadiene to EB is the rate-limiting step for the formation of all its toxic epoxy metabolites. It makes sense, given the above, to define the cumulated amount of 1,3-butadiene metabolites formed in the body as the measure of its internal dose for cancer risk assessment purposes.

3.2 Model Development and Evaluation

3.2.1 Software Choice

In our butadiene example, we will use the *R* software and its package deSolve. We will assume that the reader has a minimal working of knowledge of *R* and has *R* and deSolve installed. *R* is freely available for the major operating systems (Unix/Linux, Windows, Mac OS) and deSolve provides excellent functions for integrating differential equations. *R* is easy to use, but not particularly fast. If you need to run many simulations (say several thousands or more) you should code your model in C language, compile it, and have deSolve call your compiled code (see the deSolve manual for that). An even faster alternative (if you need to do Bayesian model calibration, for example) is to use *GNU MCSim*. You can actually use *GNU MCSim* to develop C code for deSolve.

3.2.2 Defining the Model Structure and Equations

Our research group has previously developed and published a PBPK model for 1,3-butadiene on the basis of data collected on 133 human volunteers during controlled low dose exposures. We used it for various studies and as an example of Bayesian PBPK analysis [66–68]. That model (*see* Fig. 2) is a minimal description of butadiene distribution and metabolism in the human body after inhalation. Three compartments lump together tissues with similar perfusion rate (blood flow per unit of tissue mass): the "well-perfused" compartment regroups the liver, brain, lungs,

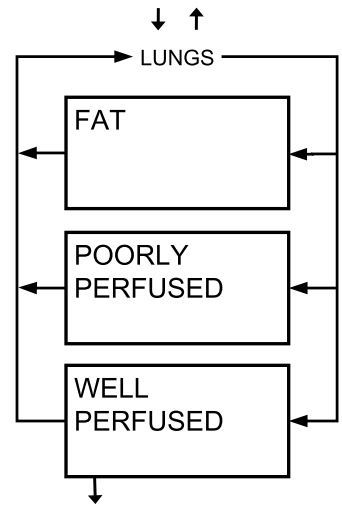

Fig. 2 Representation of the PBPK model used for 1,3-butadiene. The model equations and parameters are detailed in Subheading 3 of the text

kidneys, and other viscera; the "poorly perfused" compartment lumps muscles and skin; and the third is "fat" tissues. Butadiene can be metabolized into an epoxide in the liver, kidneys, and lung, which are part of the well-perfused compartment. Our model will therefore include four essential "state" variables, which will each have a governing differential equation: the quantities of butadiene in the fat, in the well-perfused compartment, in the poorly perfused compartment, and the quantity metabolized. Actually, the latter is a "terminal" state variable which depends on the others state variables and has no dependent. We could dispense with it if we did not want to compute and output it. That would save computation time, which grows approximately with the square of the number of state variables of the model.

In an *R* script code for use with deSolve, we first need to define the model state variable and assign them initial values (values they will take at the start of a simulation, those are called "boundary conditions" in technical jargon). The syntax is quite simple (the full script is given in Appendix):

```
y = c("Q_fat" = 0,   # Quantity of butadiene in fat (mg)
  "Q_wp"  = 0,   # ~       in well-perfused (mg)
  "Q_pp"  = 0,   # ~       in poorly-perfused (mg)
  "Q_met" = 0)   # ~       metabolized (mg)
```

That requests the creation of y as a vector of four named components, all initialized here at the value zero (i.e., we assume no previous exposure to butadiene, or no significant levels of butadiene in the body in case of a previous exposure). The portions of lines starting with the pound sign (#) are simply comments for the reader and are ignored by the software. We have chosen milligrams as the unit for butadiene quantities and it is useful to indicate it here. In *R* indentation and spacing do not matter and we strive for readability.

We then need to define similarly, as a named vector, the model parameters:

```
parameters = c(
  "BDM"    = 73,         # Body mass (kg)
  "Height" = 1.6,        # Body height (m)
  "Age"    = 40,         # in years
  "Sex"    = 1,          # code 1 is male, 2 is female
  "Flow_pul"      = 5,    # Pulmonary ventilation rate (L/min)
  "Pct_Deadspace" = 0.7,  # Fraction of pulmonary deadspace
  "Vent_Perf"     = 1.14, # Ventilation over perfusion ratio
  "Pct_LBDM_wp"   = 0.2,  # wp tissue as fraction of lean mass
  "Pct_Flow_fat"  = 0.1,  # Fraction of cardiac output to fat
  "Pct_Flow_pp"   = 0.35, # ~                    to pp
  "PC_art" = 2,          # Blood/air partition coefficient
  "PC_fat" = 22,         # Fat/blood ~
  "PC_wp"  = 0.8,        # wp/blood  ~
  "PC_pp"  = 0.8,        # pp/blood  ~
```

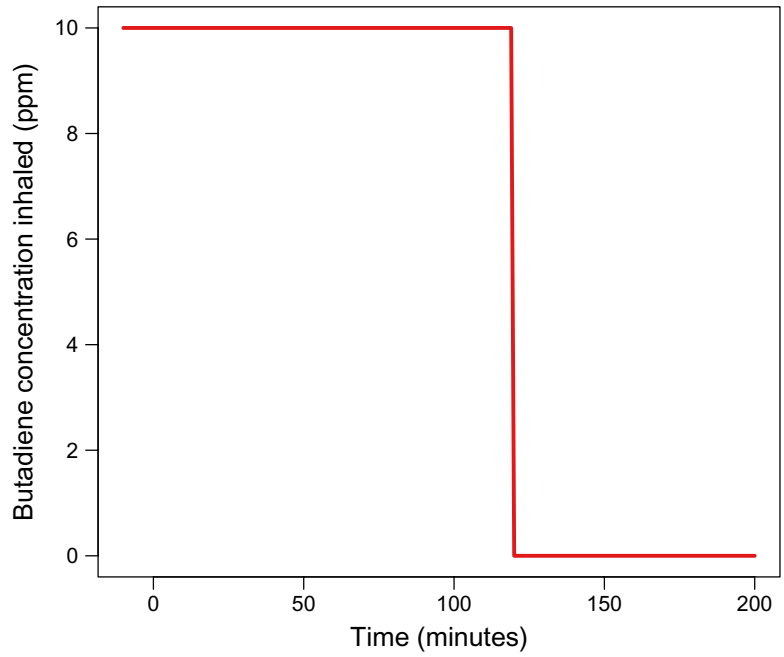

Fig. 3 Plot of the time–concentration profile of butadiene inhaled generated by the function $C_{inh}(t)$ of the example script. $C_{inh}(t)$ is used as a forcing function for the model simulations

```
"Kmetwp" = 0.25)      # Rate constant for metabolism (1/min)
```

We will see next how those parameters are used in the model equations, but you notice already that they are not exactly, except for the partition coefficients and metabolic rate constant, the parameters used in Eqs. 1, and 3–5. They are in fact scaling coefficients used to model parameter correlations in an actual subject.

Before we get to the model core equations, we need to define the value of the concentration of butadiene in inhaled air. This is an "input" to the model and we will allow it to change with time, so it is a dynamic boundary condition to the model (deSolve uses the term "forcing function"). We use here a convenient feature of *R*, defining C_{inh} as an approximating function.

```
C_inh = approxfun(x = c(0, 100), y = c(10, 0),
                  method = "constant", f = 0, rule = 2)
```

The instruction above defines a function of time $C_{inh}(t)$, right continuous (option $f = 0$) and constant by segments (the option method = "linear" would yield a function linear by segments). At times 0 and 100 (x values), it takes values y 10 and then 0, respectively. Before time zero and after time 100, $C_{inh}(t)$ will take the closest y value defined (option rule = 2). Figure 3 shows the behavior of the function $C_{inh}(t)$ so defined.

Formally you do not necessarily need such an input function in your model. C_{inh} could simply be a constant, or no input could be used if you were to model just the elimination of butadiene out of

body following exposure. Indeed, the initial values of the state variables would have to be non-null in that case.

Now we need to define a function that will compute the derivatives at the core of the model, as a function of time t—used for example when parameters are time varying, or for computing $C_{inh}(t)$, of the current state variable values y, and of the parameters. Here is the (simplified) code of that function which we called "bd.model" (intermediate calculations have been deleted for clarity, we will see them later):

```
bd.model = function(t, y, parameters) { # function header
 # function body:
 with (as.list(y), {
  with (as.list(parameters), {
  # … (part of the code omitted for now)
  # Time derivatives for quantities
  dQ_fat = Flow_fat * (C_art - Cout_fat)
  dQ_wp  = Flow_wp  * (C_art - Cout_wp) - dQmet_wp
  dQ_pp  = Flow_pp  * (C_art - Cout_pp)
  dQ_met = dQmet_wp;
     return(list(c(dQ_fat, dQ_wp, dQ_pp, dQ_met),     # derivatives
           c("C_ven" = C_ven, "C_art" = C_art)))      # extra outputs
  }) # end with parameters
 }) # end with y
} # end of function bd.model()
```

The first two "with" nested blocks (they extend up to the end of the function) are an obscure but useful feature of *R*. Remember that y and "parameters" are arrays with named components. In *R*, you should refer to their individual components by writing for example "parameters["PC_fat"]" for the fat over blood partition coefficient. That can become clumsy and the "with" statements allow you to simplify the notation and call simply "PC_fat".

The most important part of the "bd.model" function is the calculation of the derivatives. As you can see they are given an arbitrary name and computed similarly to the equations given above (e.g., Eq. 1). Obviously we need to have defined the temporary variables "Cout_fat", "Cout_wp", and "dQmet_wp" but they are part of the omitted code and we will see them next. Finally, the function needs to return (as a list, that is imposed by deSolve) the derivatives computed and eventually the output variables we might be interested in (in our case, for example C_{ven} and C_{art}).

The code we omitted for clarity was simply intermediate calculations. First some obvious conversion factors:

```
# Define some useful constants
 MW_bu = 54.0914     # butadiene molecular weight (in grams)
```

```
        ppm_per_mM = 24450 # ppm to mM under normal conditions
        # Conversions from/to ppm
        ppm_per_mg_per_l = ppm_per_mM / MW_bu
        mg_per_l_per_ppm = 1 / ppm_per_mg_per_l
```

The following instructions scale the compartment volumes to body mass. The equation for the fraction of fat is taken from [69]. That way, the volumes correlate as they should to body mass or lean body mass:

```
        # Calculate fraction of body fat
        Pct_BDM_fat = (1.2 * BDM / (Height * Height) - 10.8 *(2 - Sex) +
                0.23 * Age - 5.4) * 0.01
        # Actual volumes, 10% of body mass (bones…) receive no butadiene
        Eff_V_fat = Pct_BDM_fat * BDM
        Eff_V_wp = Pct_LBDM_wp * BDM * (1 - Pct_BDM_fat)
        Eff_V_pp = 0.9 * BDM - Eff_V_fat - Eff_V_wp
```

The blood flows are scaled similarly to maintain adequate perfusion per unit mass:

```
        # Calculate alveolar flow from total pulmonary flow
        Flow_alv = Flow_pul * (1 - Pct_Deadspace)
        # Calculate total blood flow from Flow_alv and the V/P ratio
        Flow_tot = Flow_alv / Vent_Perf
        # Calculate actual blood flows from total flow and percent flows
        Flow_fat = Pct_Flow_fat * Flow_tot
        Flow_pp = Pct_Flow_pp * Flow_tot
        Flow_wp = Flow_tot * (1 - Pct_Flow_pp - Pct_Flow_fat)
```

We have now everything needed to compute concentrations at time t in the various compartments or at their exit:

```
        # Calculate the concentrations
        C_fat = Q_fat / Eff_V_fat
        C_wp = Q_wp / Eff_V_wp
        C_pp = Q_pp / Eff_V_pp
        # Venous blood concentrations at the organ exit
        Cout_fat = C_fat / PC_fat
        Cout_wp = C_wp / PC_wp
        Cout_pp = C_pp / PC_pp
```

The next two lines are typical computational tricks. The right-hand sides will be used several times in the subsequent calculations. It is faster, and more readable to define them as temporary variables:

```
        # Sum of Flow * Concentration for all compartments
        dQ_ven = Flow_fat * Cout_fat + Flow_wp * Cout_wp +
            Flow_pp * Cout_pp
```

```
                # Quantity metabolized in liver (included in
well-perfused)
        dQmet_wp = Kmetwp * Q_wp
        C_inh.current = C_inh(t) # to avoid calling C_inh() twice
```

The last series of intermediate computations obtain C_{art}—as in Eq. 6, with a unit conversion for $C_{inh}(t)$, C_{ven} as in Eq. 7 (those two will be defined as outputs in the function's return statement), the alveolar air concentration C_{alv}, and finally the exhaled air concentration C_{exh}:

```
        # Arterial blood concentration
        # Convert input given in ppm to mg/l to match other units
        C_art = (Flow_alv * C_inh.current * mg_per_l_per_ppm +
dQ_ven) /
                (Flow_tot + Flow_alv / PC_art)
        # Venous blood concentration (mg/L)
        C_ven = dQ_ven / Flow_tot
        # Alveolar air concentration (mg/L)
        C_alv = C_art / PC_art
        # Exhaled air concentration (ppm!)
        if (C_alv <= 0) {
          C_exh = 10E-30 # avoid round off errors
        } else {
         C_exh = (1 - Pct_Deadspace) * C_alv * ppm_per_mg_per_l
+
                Pct_Deadspace * C_inh.current
        }
```

The calculation of C_{exh} just above is an example of computational trick to avoid rounding errors (useful if you later want to take the log of C_{exh}, you want to avoid values like -7×10^{-16} for example). It also illustrates one idiosyncrasy of *R*: spacing and disposition do not matter *except* that "} else {" must be on the same line.

3.2.3 Running the Model

The *R* script we detailed above is almost ready to perform simulations. We just need to define the output times (times at which we will want to look at the results, here a sequence from zero to 1440 min, every 10 min), load the deSolve library (so far we have only used standard *R* functions) and call the integration routine "ode", storing its results in the variable "results":

```
        # Define the computation output times (minutes)
        times = seq(from=0, to=1440, by=10)
        # Call the ODE solver
        library(deSolve)
        results = ode(times = times, func = bd.model, y = Y, parms
= parms)
```

By default, deSolve uses the *lsode* integration routine for stiff systems [70]. This is a very efficient solver, but you have the choice of several integrators (*see* the deSolve manual for details).

The content of results can be looked at, saved to a file, further manipulated or simply plotted:

```
# results is basically a table
results
# Plot the results of the simulation
plot(results)
```

Figure 4 shows the plot obtained (just for the four butadiene quantities state variables). That is in essence all it takes to write and simulate a PBPK model.

3.2.4 Running Monte Carlo Simulations

Running Monte Carlo simulations in *R*, for uncertainty or sensitivity analyses [49], is rather easy. *R* is fundamentally a statistical software and is well equipped for random numbers generation. The skeleton for a Monte Carlo simulation script is simply a loop of *n* iterations:

```
for (iteration in 1:1000) { # 1000 Monte Carlo simulations
  # Sample randomly some parameters
  …
  # Reduce output times eventually
  times = c(0, 1440)
```

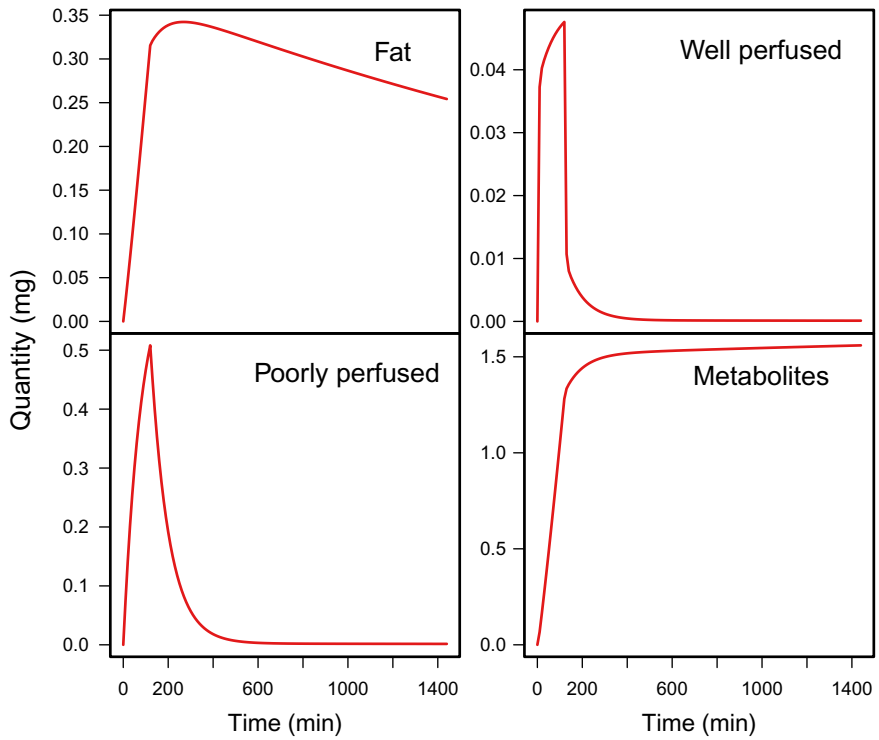

Fig. 4 Simulated time courses of the quantities of butadiene in the compartments of the sample PBPK model. Inhalation exposure was specified as shown in Fig. 3

```
  # Integrate
tmp = ode(times = times, func = bd.model, y = y,
          parms = parameters)
  # Accumulate results in a table
…
        } # end Monte Carlo loop
```

Here too the ellipsis (…) refers to pieces of code we will detail below. The full script is given in Appendix). The calculations inside the "for" loop are performed a thousand time. At each iteration, new parameter values are randomly sampled. For example, if we choose to sample only four parameters (we could sample all) from normal distributions, the code would look like:

```
  # Sample randomly some parameters
  parameters["BDM"]      = rnorm(1, 73,   7.3)
  parameters["Flow_pul"] = rnorm(1, 5,    0.5)
  parameters["PC_art"]   = rnorm(1, 2,    0.2)
  parameters["Kmetwp"]   = rnorm(1, 0.25, 0.025)
```

For each parameter, one normal random variable is drawn with a mean set to the value used in the simple script above, and a standard deviation equal to 10 % of the mean. When doing Monte Carlo simulations, you usually do not want to look at the distributions of state or output variables at thousands of different times (that is heavy). Here we decided to look at them only at time 1440 min, so we reset the times array. Note that the starting time (here zero) still needs to be defined among the times. The integrator is then called and its results stored in the "tmp" table. But that is only one set of results in a thousand and we need to accumulate those results. The following few lines of code show how to keep only the results obtained at time 1440 (line 2 or the tmp table) but without the output time (which is always 1440) (the "-1" in "tmp[2,-1]" removes the first column). It is also very useful to store the sampled parameter values:

```
        if (iteration == 1) { # initialize
         results = tmp[2,-1]
                      sampled.parms   =   c(parameters["BDM"],
parameters["Flow_pul"],
                       parameters["PC_art"], parameters["Kmetwp"])
        } else { # accumulate
         results = rbind(results, tmp[2,-1])
         sampled.parms = rbind(sampled.parms,
                                          c(parameters["BDM"],
parameters["Flow_pul"],
                     parameters["PC_art"], parameters["Kmetwp"]))
        }
```

When the Monte Carlo loop is finished we probably want to save the accumulated results in a file (unless the simulations are very fast to compute):

```
          # Save the results
```

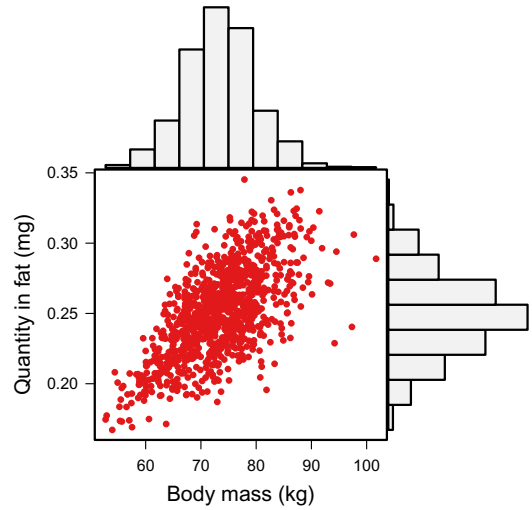

Fig. 5 Illustration of the PBPK model Monte Carlo simulation results. The dot plot shows the quantity of butadiene in fat after 1 day as a function of sampled body mass. The random sampling of other parameters explains the dispersion of the results, however the quantity in fat is clearly sensitive to body mass. The marginal histograms show the distributions of the sampled values for body mass and of the predicted quantities of butadiene in fat. A sizeable uncertainty affects those results

```
save(sampled.parms, results, file="MTC.dat.xz",
compress = "xz")
# use load(file="MTC.dat.xz") to read them back in
```

Finally, such large amounts of information are best handled with statistical and graphical methods. Figure 5 shows a nicer version of the three simple plots which would be produced by the following lines:

```
# Plot the results
hist(sampled.parms[,1])
hist(results[,1])
plot(sampled.parms[,1], results[,1])
```

Figure 5 shows the relationship between the Monte Carlo sampled body mass values and the resulting prediction for the quantity in fat after a day. You can observe an obvious and expected correlation between the two (butadiene storage in fat increases with the fat compartment volume which in turn increases with body mass). The increase in butadiene storage is roughly proportional to body mass, so that is a sensitive parameter. The relationship is not perfect because three other parameters were sampled. We can that way study the sensitivity of any model prediction, at any time, with respect to any model parameter [49]. The plot also shows the marginal distributions of body masses and butadiene quantities in fat. The uncertainty attached to predictions is about ±50 %. That type of histogram can give an idea of the reliability of any model prediction.

A thousand Monte Carlo simulations took us a few minutes on a laptop computer. A thousand is actually a small number if you want to accurately characterize upper or lower percentiles of the resulting distributions. If computation time becomes an issue you can divide it by a factor 10 if you compile your model in C—*GNU MCSim* [53, 54] can actually produce a C code compatible with deSolve without having to learn the C language. A factor 100 can be gained if you work only with *GNU MCSim*.

4 Conclusion

PBPK modeling is more and more used in research, development, and regulation [71, 72]. Obviously, the precision and accuracy of PBPK model will be only as good as those of the QSAR predictions or in vitro data used to set their parameters. Quality assurance of those components is therefore an important issue [26, 73], and we have seen that in several areas (metabolism in particular), research work is still needed. As to the models themselves, their validity will probably be easier to check if they are generic and with a stable and well-documented structure [74]. This requirement, however, runs somewhat contrary to the next challenge: Coupling PBPK models to predictive pharmacology or toxicity models, both at the cellular level and at the organ level [75]. We hope however, that this step-by-step introduction to PBPK model development and simulation will help the reader in his/her first steps into that exciting area.

Acknowledgment

This work was partly supported by the European Commission, 7th FP project 4-FUN (grant agreement 308440) and the French Ministry for the Environment (Programme 190 toxicologie).

5 Appendix

R script for the butadiene PBPK model:

```
    #=============================================
==================
    # Butadiene human PBPK model
    # Define and initialize the state variables
    y = c("Q_fat" = 0,   # Quantity of butadiene in fat (mg)
          "Q_wp"  = 0,   # ~       in well-perfused (mg)
          "Q_pp"  = 0,   # ~       in poorly-perfused (mg)
          "Q_met" = 0)   # ~       metabolized (mg)
    # Define the model parameters
    # Units:
```

```
# Volumes: liter
# Time:   minute
# Flows:  liter / minute
parameters = c(
  "BDM"    = 73,         # Body mass (kg)
  "Height" = 1.6,        # Body height (m)
  "Age"    = 40,         # in years
  "Sex"    = 1,          # code 1 is male, 2 is female
  "Flow_pul"      = 5,   # Pulmonary ventilation rate (L/min)
  "Pct_Deadspace" = 0.7, # Fraction of pulmonary deadspace
  "Vent_Perf"     = 1.14, # Ventilation over perfusion ratio
  "Pct_LBDM_wp"   = 0.2, # wp tissue as fraction of lean mass
  "Pct_Flow_fat"  = 0.1, # Fraction of cardiac output to fat
  "Pct_Flow_pp"   = 0.35, # ~                    to pp
  "PC_art" = 2,          # Blood/air partition coefficient
  "PC_fat" = 22,         # Fat/blood ~
  "PC_wp"  = 0.8,        # wp/blood  ~
  "PC_pp"  = 0.8,        # pp/blood  ~
  "Kmetwp" = 0.25)       # Rate constant for metabolism (1/min)
# The input air concentration (in parts per million) can vary with time
C_inh = approxfun(x = c(0,120), y = c(10,0), method="constant", f=0, rule=2)
# Check the input concentration profile just defined
plot(C_inh(1:300), xlab = "Time (min)",
     ylab = "Butadiene air concentration (ppm)", type = "l")
# Define the model equations
bd.model = function(t, y, parameters) {
 with (as.list(y), {
  with (as.list(parameters), {
  # Define some useful constants
  MW_bu = 54.0914   # butadiene molecular weight (in grams)
    ppm_per_mM = 24450 # ppm to mM under normal conditions
  # Conversions from/to ppm
  ppm_per_mg_per_l = ppm_per_mM / MW_bu
  mg_per_l_per_ppm = 1 / ppm_per_mg_per_l
  # Calculate Flow_alv from total pulmonary flow
  Flow_alv = Flow_pul * (1 - Pct_Deadspace)
  # Calculate total blood flow from Flow_alv and the V/P ratio
  Flow_tot = Flow_alv / Vent_Perf
  # Calculate fraction of body fat
  Pct_BDM_fat = (1.2 * BDM / (Height * Height) - 10.8 *(2 - Sex) +
                 0.23 * Age - 5.4) * 0.01
  # Actual volumes, 10% of body mass (bones…) get no butadiene
```

```
    Eff_V_fat = Pct_BDM_fat * BDM
    Eff_V_wp  = Pct_LBDM_wp  * BDM * (1 - Pct_BDM_fat)
    Eff_V_pp  = 0.9 * BDM - Eff_V_fat - Eff_V_wp
     # Calculate actual blood flows from total flow and percent
flows
    Flow_fat = Pct_Flow_fat * Flow_tot
    Flow_pp  = Pct_Flow_pp  * Flow_tot
    Flow_wp  = Flow_tot * (1 - Pct_Flow_pp - Pct_Flow_fat)
    # Calculate the concentrations
    C_fat = Q_fat / Eff_V_fat
    C_wp  = Q_wp  / Eff_V_wp
    C_pp  = Q_pp  / Eff_V_pp
    # Venous blood concentrations at the organ exit
    Cout_fat = C_fat / PC_fat
    Cout_wp  = C_wp  / PC_wp
    Cout_pp  = C_pp  / PC_pp
    # Sum of Flow * Concentration for all compartments
     dQ_ven = Flow_fat * Cout_fat + Flow_wp * Cout_wp +
Flow_pp * Cout_pp
    C_inh.current = C_inh(t) # to avoid calling C_inh() twice
    # Arterial blood concentration
    # Convert input given in ppm to mg/l to match other units
     C_art = (Flow_alv * C_inh.current * mg_per_l_per_ppm +
dQ_ven) /
          (Flow_tot + Flow_alv / PC_art)
    # Venous blood concentration (mg/L)
    C_ven = dQ_ven / Flow_tot
    # Alveolar air concentration (mg/L)
    C_alv = C_art / PC_art
    # Exhaled air concentration (ppm!)
    if (C_alv <= 0) {
      C_exh = 10E-30 # avoid round off errors
    } else {
     C_exh = (1 - Pct_Deadspace) * C_alv * ppm_per_mg_per_l +
          Pct_Deadspace * C_inh.current
    }
    # Quantity metabolized in liver (included in well-perfused)
    dQmet_wp = Kmetwp * Q_wp
    # Differentials for quantities
    dQ_fat = Flow_fat * (C_art - Cout_fat)
    dQ_wp  = Flow_wp  * (C_art - Cout_wp) - dQmet_wp
    dQ_pp  = Flow_pp  * (C_art - Cout_pp)
    dQ_met = dQmet_wp
    # The function bd.model must return at least the derivatives
    list(c(dQ_fat, dQ_wp, dQ_pp, dQ_met),     # derivatives
        c("C_ven" = C_ven, "C_art" = C_art)) # extra outputs
    }) # end with parameters
  }) # end with y
```

```
} # end bd.model
# Define the computation output times
times = seq(from=0, to=1440, by=10)
# Call the ODE solver
library(deSolve)
results = ode(times = times, func = bd.model, y = y, parms =
parameters)
# results is basically a table
results
# Plot the results of the simulation
plot(results)
# End
# End Simple Simulation.
#================================================
==================
#================================================
==================
# Monte Carlo simulations
# We assume that a simple simulation has already been run, so
that
# y, parameters, C_inh, and bd.model have all been defined
and that
# deSolve has been loaded.
for (iteration in 1:1000) { # 1000 Monte Carlo simulations…
 # Sample randomly some parameters
 parameters["BDM"]     = rnorm(1, 73,   7.3)
 parameters["Flow_pul"] = rnorm(1, 5,    0.5)
 parameters["PC_art"]   = rnorm(1, 2,    0.2)
 parameters["Kmetwp"]   = rnorm(1, 0.25, 0.025)
 # Reduce output times eventually. We only care about time
1440,
 # but time zero still needs to be specified
 times = c(0, 1440)
 # Integrate
 tmp = ode(times = times, func = bd.model, y = y, parms =
parameters)
 if (iteration == 1) { # initialize
  results = tmp[2,-1]
            sampled.parms      =      c(parameters["BDM"],
parameters["Flow_pul"],
             parameters["PC_art"], parameters["Kmetwp"])
 } else { # accumulate
  results = rbind(results, tmp[2,-1])
  sampled.parms = rbind(sampled.parms,
                                      c(parameters["BDM"],
parameters["Flow_pul"],
            parameters["PC_art"], parameters["Kmetwp"]))
 }
```

```
    } # end Monte Carlo loop
    # Save the results, specially if they took a long time to
compute
    save(sampled.parms, results, file="MTC.dat.xz", compress
= "xz")
    # use load(file="MTC.dat.xz") to read them back in
    # Plot the results
    hist(sampled.parms[,1])
    hist(results[,1])
    plot(sampled.parms[,1], results[,1])
    # End Monte Carlo Simulations.
    #=========================================
==================
```

References

1. Haggard HW (1924) The absorption, distribution, and elimination of ethyl ether. I. The amount of ether absorbed in relation to the concentration inhaled and its fate in the body. J Biol Chem 59:737–751
2. Haggard HW (1924) The absorption, distribution, and elimination of ethyl ether. II. Analysis of the mechanism of absorption and elimination of such a gas or vapor as ethyl ether. J Biol Chem 59:753–770
3. Haggard HW (1924) The absorption, distribution, and elimination of ethyl ether. III. The relation of the concentration of ether, or any similar volatile substance, in the central nervous system to the concentration in the arterial blood, and the buffer action of the body. J Biol Chem 59:771–781
4. Haggard HW (1924) The absorption, distribution, and elimination of ethyl ether. IV. The anesthetic tension of ether and the physiological response to various concentrations. J Biol Chem 59:783–793
5. Haggard HW (1924) The absorption, distribution, and elimination of ethyl ether. V. The importance of the volume of breathing during the induction and termination of ether anesthesia. J Biol Chem 59:795–802
6. Teorell T (1937) Kinetics of distribution of substances administered to the body. Arch Int Pharmacodyn Ther 57:205–240
7. Gibaldi M, Perrier D (1982) Pharmacokinetics. Marcel Dekker, New York
8. Bischoff KB, Dedrick RL, Zaharko DS et al (1971) Methotrexate pharmacokinetics. J Pharm Sci 60:1128–1133
9. Dedrick RL, Forrester DD, Cannon JN et al (1973) Pharmacokinetics of 1-beta-D-arabinofuranosylcytosine (ARA-C) deamination in several species. Biochem Pharmacol 22: 2405–2417
10. Gerlowski LE, Jain RK (1983) Physiologically based pharmacokinetic modeling: principles and applications. J Pharm Sci 72:1103–1127
11. Droz PO, Guillemin MP (1983) Human styrene exposure—V. Development of a model for biological monitoring. Int Arch Occup Environ Health 53:19–36
12. Lutz RJ, Dedrick RL, Tuey D et al (1984) Comparison of the pharmacokinetics of several polychlorinated biphenyls in mouse, rat, dog, and monkey by means of a physiological pharmacokinetic model. Drug Metab Dispos 12:527–535
13. Rowland M, Peck C, Tucker G (2011) Physiologically-based pharmacokinetics in drug development and regulatory science. Annu Rev Pharmacol Toxicol 51:45–73
14. Nestorov I, Aarons L, Rowland M (1998) Quantitative structure-pharmacokinetics relationships: II. A mechanistically based model to evaluate the relationship between tissue distribution parameters and compound lipophilicity. J Pharmacokinet Biopharm 26:521–545
15. Peyret T, Poulin P, Krishnan K (2010) A unified algorithm for predicting partition coefficients for PBPK modeling of drugs and environmental chemicals. Toxicol Appl Pharmacol 249:197–207
16. Jones HM, Parrott N, Jorga K et al (2006) A novel strategy for physiologically based predictions of human pharmacokinetics. Clin Pharmacokinet 45:511–542
17. Beaudouin R, Micallef S, Brochot C (2010) A stochastic whole-body physiologically based pharmacokinetic model to assess the impact of inter-individual variability on tissue dosimetry

over the human lifespan. Regul Toxicol Pharmacol 57:103–116

18. Clewell HJ III, Gentry PR, Covington TR et al (2004) Evaluation of the potential impact of age- and gender-specific pharmacokinetic differences on tissue dosimetry. Toxicol Sci 79: 381–393
19. United States Environmental Protection Agency (US EPA) (2006) Approaches for the application of Physiologically Based Pharmacokinetic (PBPK) models and supporting data in risk assessment (final report). United States Environmental Protection Agency, Washington, DC
20. International Programme on Chemical Safety (IPCS) (2010) Characterization and application of physiologically based pharmacokinetic models in risk assessment, World Health Organization. International Programme on Chemical Safety, Geneva
21. Peters SA (2011) Physiologically Based Pharmacokinetic (pbpk) modeling and simulations: principles, methods, and applications in the pharmaceutical industry. Wiley, Hoboken, NJ
22. Andersen ME (1995) What do we mean by … dose? Inhal Toxicol 7:909–915
23. Clewell HJ, Tan YM, Campbell JL et al (2008) Quantitative interpretation of human biomonitoring data. Toxicol Appl Pharmacol 231:122–133
24. Ulaszewska MM, Ciffroy P, Tahraoui F et al (2012) Interpreting PCB levels in breast milk using a physiologically based pharmacokinetic model to reconstruct the dynamic exposure of Italian women. J Expo Sci Environ Epidemiol 22:601–609
25. Zeman FA, Boudet C, Tack K et al (2013) Exposure assessment of phthalates in French pregnant women: results of the ELFE pilot study. Int J Hyg Environ Health 216:271–279
26. Loizou G, Spendiff M, Barton HA et al (2008) Development of good modelling practice for physiologically based pharmacokinetic models for use in risk assessment: the first steps. Regul Toxicol Pharmacol 50:400–411
27. Barton HA, Chiu WA, Setzer W et al (2007) Characterizing uncertainty and variability in physiologically-based pharmacokinetic (PBPK) models: state of the science and needs for research and implementation. Toxicol Sci 99:395–402
28. Andersen ME (1995) Development of physiologically based pharmacokinetic and physiologically based pharmacodynamic models for applications in toxicology and risk assessment. Toxicol Lett 79:35–44
29. Clewell RA, Clewell HJ (2008) Development and specification of physiologically based pharmacokinetic models for use in risk assessment. Regul Toxicol Pharmacol 50:129–143
30. Campbell JL Jr, Clewell RA, Gentry PR et al (2012) Physiologically based pharmacokinetic/toxicokinetic modeling. In: Reisfeld B, Mayeno AN (eds) Computational toxicology, Methods in molecular biology series. Humana, New York, pp 439–499
31. Thompson MD, Beard DA (2011) Development of appropriate equations for physiologically based pharmacokinetic modeling of permeability-limited and flow-limited transport. J Pharmacokinet Pharmacodyn 38:405–421
32. International Commission on Radiological Protection (ICRP) (1975) Report of the Task Group on Reference Man—a report prepared by a Task Group of Committee 2 of the International Commission on Radiological Protection. Pergamon, Oxford
33. International Commission on Radiological Protection (ICRP) (2002) Basic anatomical and physiological data for use in radiological protection: reference values. ICRP Publication 89. Ann ICRP 32(3–4):5–265
34. Davies B, Morris T (1993) Physiological parameters in laboratory animals and humans. Pharm Res 10:1093–1095
35. Brown RP, Delp MD, Lindstedt ST et al (1997) Physiological parameter values for physiologically based pharmacokinetic models. Toxicol Ind Health 14:407–484
36. Young JF, Branham WS, Sheenan DM et al (1997) Physiological "constants" for PBPK models for pregnancy. J Toxicol Environ Health 52:385–401
37. Ekins S, Waller CL, Swaan PW et al (2000) Progress in predicting human ADME parameters in silico. J Pharmacol Toxicol Methods 44:251–272
38. Poulin P, Haddad S (2012) Advancing prediction of tissue distribution and volume of distribution of highly lipophilic compounds from a simplified tissue-composition-based model as a mechanistic animal alternative method. J Pharm Sci 101:2250–2261
39. Butina D, Segall MD, Frankcombe K (2002) Predicting ADME properties in silico: methods and models. Drug Discov Today 7:S83–S88
40. Buch I, Giorgino T, De Fabritiis G (2011) Complete reconstruction of an enzyme-inhibitor binding process by molecular dynamics simulations. Proc Natl Acad Sci U S A 108(25): 10184–10189
41. Fiserova-Bergerova V, Diaz ML (1986) Determination and prediction of tissue-gas

partition coefficients. Int Arch Occup Environ Health 58:75–87
42. Gargas ML, Seybold PG, Andersen ME (1988) Modeling the tissue solubilities and metabolic rate constant (Vmax) of halogenated methanes, ethanes, and ethylenes. Toxicol Lett 43:235–256
43. Poulin P, Theil FP (2000) A priori prediction of tissue: plasma partition coefficients of drugs to facilitate the use of physiologically-based pharmacokinetic models in drug discovery. J Pharm Sci 89:16–35
44. Poulin P, Ekins S, Theil FP (2011) A hybrid approach to advancing quantitative prediction of tissue distribution of basic drugs in human. Toxicol Appl Pharmacol 250:194–212
45. Adler S, Basketter D, Creton S et al (2011) Alternative (non-animal) methods for cosmetics testing: current status and future prospects—2010. Arch Toxicol 85:367–485
46. Bal-Price A, Jennings P (eds) (2014) In vitro toxicology systems. Humana, New York
47. Bois FY, Jamei M (2012) Population-based pharmacokinetic modeling and simulation. In: Lyubimov AV (ed) Encyclopedia of drug metabolism and interactions. Wiley, Hoboken, pp 1–27
48. Yuh L, Beal S, Davidian M et al (1994) Population pharmacokinetic/pharmacodynamic methodology and applications: a bibliography. Biometrics 50:566–575
49. Bernillon P, Bois FY (2000) Statistical issues in toxicokinetic modeling: a Bayesian perspective. Environ Health Perspect 108(Suppl 5):883–893
50. Gelman A, Bois FY, Jiang J (1996) Physiological pharmacokinetic analysis using population modeling and informative prior distributions. J Am Stat Assoc 91:1400–1412
51. Bois FY, Jamei M, Clewell HJ (2010) PBPK modelling of inter-individual variability in the pharmacokinetics of environmental chemicals. Toxicology 278:256–267
52. R Development Core Team (2013) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria
53. Bois FY (2009) GNU MCSim: Bayesian statistical inference for SBML-coded systems biology models. Bioinformatics 25:1453–1454
54. Anonymous. GNU MCSim. http://www.gnu.org/software/mcsim/
55. Anonymous. GNU Octave. http://www.gnu.org/software/octave/
56. Anonymous Scilab. http://www.scilab.org/
57. Anonymous. MATLAB and Simulink for technical computing—MathWorks. http://mathworks.com/
58. Anonymous. Wolfram mathematica: definitive system for modern technical computing. http://www.wolfram.com/mathematica/
59. Anonymous. Bayer technology services: PK-Sim®. http://www.systems-biology.com/products/pk-sim.html
60. Anonymous. Simcyp—population based pharmacokinetic modelling and simulation. http://www.simcyp.com/
61. Anonymous. Simulations Plus, Inc|Modeling & Simulation Software|Consulting Services for Pharmaceutical Research. http://www.simulations-plus.com/
62. Anonymous. MERLIN-Expo|Exposure Assessment Software. http://merlin-expo.4funproject.eu/
63. Bois FY (2009) Physiologically-based modelling and prediction of drug interactions. Basic Clin Pharmacol Toxicol 106:154–161
64. United States Environmental Protection Agency (US EPA) (2002) Health Assessment of 1,3-Butadiene. United States Environmental Protection Agency, Office of Research and Development, National Center for Environmental Assessment, Washington Office, Washington, DC
65. Kirman CR, Albertini RJ, Sweeney LM et al (2010) 1,3-Butadiene: I. Review of metabolism and the implications to human health risk assessment. Crit Rev Toxicol 40:1–11
66. Bois FY, Smith T, Gelman A et al (1999) Optimal design for a study of butadiene toxicokinetics in humans. Toxicol Sci 49:213–224
67. Mezzetti M, Ibrahim JG, Bois FY et al (2003) A Bayesian compartmental model for the evaluation of 1,3-butadiene metabolism. J R Stat Soc Ser C 52:291–305
68. Bois F (2012) Bayesian inference. In: Reisfeld B, Mayeno AN (eds) Computational toxicology, vol II. Humana, New York, pp 597–636
69. Deurenberg P, Weststrate JA, Seidell JC (1991) Body mass index as a measure of body fatness: age- and sex-specific prediction formulas. Br J Nutr 65:105–141
70. Byrne GD, Hindmarsh AC (1987) Stiff ODE solvers: a review of current and coming attractions. J Comput Phys 70:1–62
71. Caldwell JC, Evans MV, Krishnan K (2012) Cutting edge PBPK models and analyses: providing the basis for future modeling efforts and bridges to emerging toxicology paradigms. J Toxicol 2012:852384
72. Zhao P, Zhang L, Grillo JA et al (2011) Applications of Physiologically Based Pharmacokinetic (PBPK) modeling and simu-

lation during regulatory review. Clin Pharmacol Ther 89:259–267

73. Zhao P, Rowland M, Huang S-M (2012) Best practice in the use of physiologically based pharmacokinetic modeling and simulation to address clinical pharmacology regulatory questions. Clin Pharmacol Ther 92:17–20
74. Rostami-Hodjegan A, Tamai I, Pang KS (2012) Physiologically based pharmacokinetic (PBPK) modeling: it is here to stay! Biopharm Drug Dispos 33:47–50
75. Geenen S, Yates JW, Kenna JG et al (2013) Multiscale modelling approach combining a kinetic model of glutathione metabolism with PBPK models of paracetamol and the potential glutathione-depletion biomarkers ophthalmic acid and 5-oxoproline in humans and rats. Integr Biol 5:877–888