Bayesian Dependency Modelling and Inference for PBPK Model Parameters: A *Nimble* Approach

Frederic Y. Bois (https://orcid.org/0000-0002-4154-0391),

Vo Hong Thanh (https://orcid.org/0000-0003-4599-0215),

Hiroshi Momiji (https://orcid.org/0000-0002-3015-5770),

Heeseung Jo,

Masoud Jamei (https://orcid.org/0000-0002-3443-0194),

Amin Rostami-Hodjegan (https://orcid.org/0000-0003-3917-844X).

CERTARA UK Limited, Simcyp division, Sheffield, UK.

Corresponding author: Frederic Y. Bois, Frederic.Bois@certara.com

CERTARA UK Limited, Simcyp division, Sheffield, UK.

Tel: +44 (0) 740 780 3944

ACKOWLEDGEMENTS

The authors are thankful to the Simcyp Library team for helping with literature access.

CONFLICT OF INTEREST

All authors are employees and potential shareholders of Certara.

FUNDING

N/A

Abstract

The increasing reliance on physiologically-based pharmacokinetic (PBPK) models instead of clinical trials, for decision making in drug development, calls for a framework able to represent complex parameter dependencies when creating realistic virtual subjects and populations. We describe here a graph-based solution to harness such dependencies and extend it to hierarchical population models usable for statistical inference. We show how to model complex parameter and data dependencies, perform efficient Monte Carlo simulations to generate virtual individuals, but also rigorous Bayesian inference on parameters when observed individual covariates are available. Plasma concentration versus time data of theophylline in human volunteers are used as an example of application of a PBPK model with built-in covariate structure, as implemented in the Simcyp Simulator. A range of models of increasing sophistication is considered. We shows that application of a stationary Markov chain Monte Carlo (MCMC) sampler can successfully reduce the complexity of full Bayesian inference.

Keywords: Algorithms; Bayesian inference; PBPK model; Population pharmacokinetic model; Statistical inference; Theophylline.

1 Introduction

Usual population pharmacokinetic analyses of covariates suffer from the “tobacco smoking syndrome” of early epidemiology: by not considering known effects of smoking, standard analyses of epidemiological data either rediscovered each time those effects, or lost power when trying to discover other effects of interest[1, 2]. In pharmacokinetics modelling, this syndrome translates into cases such as ascribing high trough values of clozapine in female patients to lower clearance rather than correctly explaining them by longer half-lives due to higher fat content in composition of the female body and its association with larger volume of distribution[3]. Recently, it was shown that ignoring interaction terms in covariate analysis

might be masking the age-dependent effect of genotype on drug clearance[4] or gene-dependent effects of smoking[5]. To remedy this problem, a top-down approach could assign, in a Bayesian framework, informative priors to known covariate effects. To our knowledge, that is never done, despites the range of techniques, including Bayesian ones, that are used[6, 7]. Bottom-up approaches, on the contrary, rely entirely on prior knowledge and include known covariate effects in population-based physiologically-based pharmacokinetic (PBPK) model run in predictive simulations. We propose to extend a Bayesian middle out approach[8–13], which both accounts for prior information and evidence from clinical data, to deal with complex covariate models. However, this is an uncharted territory in which caution should be exercised.

One problem is that publications on PBPK models tend to focus on explaining their structural basis and demonstrating their prediction capabilities in applications[14–17]. Less attention is usually devoted to their parameters, with the exception of early papers focusing on reference values[18, 19], or some papers which describe "parameter scaling" or "covariate modelling" in the sense of modelling parameter dependencies *a priori*[8, 20–22]. Covariate modeling has a different focus in population pharmacokinetics, aiming to discover, *a posteriori*, unknown correlations between structural model parameters[7, 23–25]. In fact, correct modelling of parameter dependencies and joint distributions, either *a priori* or *a posteriori*, is as important as correct structural modelling, because they are all part of the model. This is true both qualitatively: incorrect equations make for poor predictions; and quantitatively: correct equations with poor parameterization also make for poor predictions. The complexity of the current knowledge database is also a challenge: the Simcyp Simulator® models, for example, may include hundreds of differential equations, which have thousands of parameters with complex dependencies.

Therefore, we propose to treat the parameter dependencies of a PKPK as a Bayesian network in which the model structural (differential) equations play the role of a deterministic link with

system-level data, such as tissue and organ concentrations. Adopting this view allows us to update the prior distributions of any parameter and sets of them with data, using Bayesian inference. We also show how to introduce data or partial information directly relevant to parameter values. In any inference framework, Bayesian or not, uncertainty about all parameters should be accounted for and the distribution of all uncertain model parameters should be updated. However, that is potentially very expensive from a computational point of view with complex models. Sensitivity analysis can help eliminate the least sensitive parameters (likely to be non-identifiable)[26], but it is often used to select only the most sensitive parameters, leaving many others "ancillary" parameters aside, with unclear justification. We explore the possibility of allowing "ancillary" parameters to affect inference on a proposed set of most sensitive parameters, but without updating the prior distribution of those extra parameters. Our statistical models are of increasing complexity, as in a multiverse approach[27]. This can be done with stationary MCMC sampler, which we implemented using the Bayesian numerical package *Nimble* in *R*. We illustrate our approach with a statistical population analysis of theophylline data, as in[10, 13].

2 Theoretical background

We use probabilistic networks[28–30], represented as directed acyclic graphs (DAG), to model parameter dependences. DAG represent variables as nodes and their relationships as directed edges. An edge from one node (parent) to another (child) represents a directed dependency between these nodes, and that dependency can either be probabilistic (*via* a distribution, conditional on the child's parents) or deterministic. The joint multivariate distribution of nodes is therefore encoded in the network. Figure 1 shows an example of such a DAG, for the minimal PBPK model of theophylline pharmacokinetics in humans.

A DAG representation of parameter dependences provides a formal computational framework for simulating, sampling, and updating all nodes' values on the basis of data. For simulation, it is enough to compute or sample individual node following their topological order (first the nodes without parents, then their direct descendants *etc*.), which can be computed by topological sorting using efficient algorithms[31]. When data nodes are included in the network, they introduce constrains on their ancestors and the joint distribution of the node values may not have an analytical representation. In that case, sampling node values from their joint posterior distribution is best achieved by Bayesian numerical methods such as Markov Chain Monte Carlo (MCMC) simulations. In particular, the Metropolis Hastings (MH) random walk algorithm is a well-known MCMC method[13, 32–34], which we make use of here. Note that the MH sampler does not immediately start sampling from the target posterior distribution but must be iterated until the samples it provides converge in distribution to the posterior of interest. This is usually checked by running a set of independent simulated Markov chains[35]. In a formal statistical inference framework, covariate data about an individual should inform the value of all related (dependent or parent) parameters, not just the dependent ones. It is also useful to be able to easily change the structure of the parameter dependency model; a flexible network representation of parameter dependencies helps with that.

However, full Bayesian updating of a very large parameter network is computationally expensive and potentially plagued by identifiability problems, even with informative priors at the population level. Therefore, we simply propose to sample randomly a set of ancillary parameters, always from their prior distributions. In an MH scheme, this means that candidate values for those parameters are always accepted. We call such a sampler “stationary”; it does in fact Monte Carlo sampling, but in the wider context of MCMC sampling from a posterior distribution. This leads to a marginalization of the joint posterior distribution of parameters of interest (updated) with respect to the prior of the ancillary parameters. We developed two

versions of the stationary sampler: one for single ancillary parameter sampling and the other for block sampling of all ancillary parameters.

An intriguing possibility, which we also briefly explore in the following, is that stationary sampling, by augmenting the size of the parameter space, may smoothen the gradients of joint posterior parameter distribution, and therefore help speed up the convergence of MCMC samplers to that distribution, similarly to what tempered MCMC sampling achieves[11].

3 Materials and Methods

3.1 Experimental data

A fully published theophylline data set was used[13, 36, 37]. Eighteen healthy male subjects aged from 21- to 31-year-old were administered an immediate release formulation of theophylline. Oral doses of theophylline of 320 mg (1776 μmol), 268 mg, and 190 mg were given to subjects A-K, L, and M-R, respectively. The concentration (in mg/L) of theophylline in blood plasma samples was measured. Concentrations at time zero were null and not used. Body weight was recorded only for subjects A to L and was missing for the others.

3.2 Models

3.2.1 PBPK model structural equations

The Simcyp® minimal PBPK model, with systemic, portal vein and liver well-stirred compartments, first order absorption, and hepatic CYP1A2 metabolism was used to model theophylline pharmacokinetics, as in Wedagedera e*t al.* [13] (model equations are provided in the Supplemental Material for that publication).

3.2.2 Parameter dependency models

We studied five parameter dependency models, labelled *A* to *E* in the following. The ***basic parameter dependency model (A)*** associated to the above minimal PBPK model is shown in Figure 1. The dependency model actually used in Simcyp® is a bit more complex because the subjects' sex is considered. Since all the subjects for which we have theophylline data are males, only the corresponding part of the model was used. The list of parameters, their symbols, units, and values are given in **Error! Reference source not found.**. Some parameters ($K_{p,liv}$, BP, D_{liv}, $CYP1A2_{lab}$, $MPPGL$, K_m, f_a, f_g, f_u, $CL_{r,h}$, and $Creat$) had constant values for all subjects. To simulate the i^{th} random subject, four "parent" parameters (Age_i, V_{ssi}, K_{ai}, V_{maxi}) were randomly sampled from prior distributions. In the following, the ~ operator means "distributed as" and the normal distribution is parameterized with mean and SD. The age distribution corresponds to the information given in the theophylline dataset[36, 37]. The three other parameters are very dependent on theophylline PK and we assigned them (for Monte Carlo simulations) informative distributions centered around the values estimated previously[13].

$$Age_i \sim \text{Uniform}(21,31) \quad (1)$$

$$V_{ss_i} \sim \text{Normal}(0.5,0.05) \quad (2)$$

$$K_{a_i} \sim \text{Normal}(2,0.2) \quad (3)$$

$$V_{max_i} \sim \text{Normal}(10,1) \quad (4)$$

The other parameters depended deterministically or stochastically on the above ones:

$$Ht_i \sim \text{Normal}(175.32 + 0.113 \times Age_i - 0.0025 \times Age_i^2, 5) \quad (5)$$

$$\mu_{BM_i} = \exp(2.643 + 0.0099 \times Ht_i) \quad (6)$$

$$BM_i \sim \text{Normal}\left(\mu_{BM_i}, 0.15 \times \mu_{BM_i}\right) \quad (7)$$

$$BSA_i = 0.00718 \times BM_i^{0.425} \times Ht_i^{0.725} \quad (8)$$

$$V_{h_i} \sim \text{Normal}(0.722 \times BSA_i^{1.176}, 0.025) \quad (9)$$

$$LW_i = V_{h_i} \times D_{liv} \quad (10)$$

$$V_{sys_i} = V_{ss_i} \times BM_i - V_{h_i} \times K_{p,liv}/BP \quad (11)$$

$$GFR_i = (140 - Age_i) \times \frac{BM_i \times 1.73}{0.814 \times Creat \times BSA_i} \quad (12)$$

$$RF_i = {}^{GFR_i}/_{130} \quad (13)$$

$$Cl_{r_i} = RF_i \times Cl_{r,h} \quad (14)$$

$$CO_i = BSA_i \times 60 \times \left(3 - 0.01 \times (Age_i - 20)\right) \quad (15)$$

$$Q_{ha_i} = 0.196 \times CO_i \quad (16)$$

$$Q_{pv_i} = 0.065 \times CO_i \quad (17)$$

$$Q_{h_i} = Q_{ha_i} + Q_{pv_i} \quad (18)$$

$$V_{max,sc_i} = V_{max_i} \times CYP1A2_{lab} \times MPPGL \times LW_i \times 60 \times 10^{-5} \quad (19)$$

Table 1: Caucasian healthy male demographic and minimal PBPK model parameters used in our analyses.

Parameter	Symbol	Units	Value	Reference
Body weight	*BM*	kg	70	[36, 37]
Liver to plasma partition coefficient	$K_{p,liv}$	-	1	[45]
Blood to plasma partition coefficient	*BP*	-	0.815	[46]
Liver density	D_{liv}	g/L	1080	[47]
CYP1A2 enzyme abundance in liver	$CYP1A2_{lab}$	pmol/mg protein	52	[48]
Microsomal protein per gram of liver	*MPPGL*	mg/g liver	39.79	[49]
Michaelis-Menten coefficient	K_m	µM	377	[50]
Fraction absorbed	f_a	-	1	- [a]
Fraction escaping gut metabolism	f_g	-	1	- [a]
Fraction of drug unbound in plasma	f_u	-	0.5	[51]
Renal clearance in healthy subjects	$CL_{r,h}$	L/h	0.31	[52]
Blood serum creatinine concentration	*Creat*	µmol/L	76.5	[53]
Volume of distribution at steady state	V_{ss}	L/kg	-	- [b]
First order absorption rate	K_a	hr^{-1}	-	- [b]
Maximum rate of metabolism	V_{max}	pmol/min/pmol CYP1A2	-	- [b]
Age	*Age*	year	-	- [c]
Body height	*Ht*	cm	-	- [c]
Body surface area	*BSA*	m^2	-	- [c]
Liver tissue volume	V_h	L	-	- [c]
Liver weight	*LW*	g	-	- [c]
Blood plasma volume	V_{sys}	L	-	- [c]
Glomerular filtration rate	*GFR*	ml/min	-	- [c]
Renal function	*RF*	-	-	- [c]
Renal clearance from blood plasma	CL_r	L/h	-	- [c]
Cardiac output	*CO*	L/h	-	- [c]
Hepatic arterial blood flow	Q_{ha}	L/h	-	- [c]
Portal vein blood flow	Q_{pv}	L/h	-	- [c]
Total hepatic blood flow	Q_h	L/h	-	- [c]
Scaled maximum rate of metabolism	$V_{max,sc}$	µmol/L	-	- [c]

[a] Value assumed.

[b] Value always randomly sampled (see text).

[c] Dependent parameter, constant or sampled

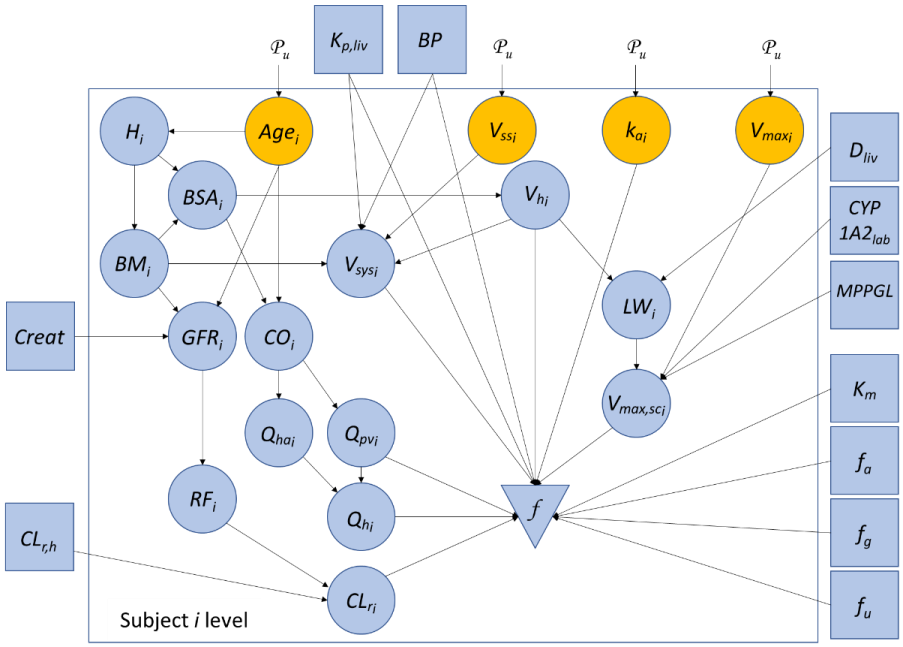

Figure 1: Basic parameter dependency graph (model *A*) for the minimal PBPK model of theophylline pharmacokinetics in humans. P_u are prior distributions; Square nodes have fixed values; Round orange nodes are randomly sampled from their priors. Round blue nodes are dependent parameters; Triangular node *f* is the structural ODE model.

Model *B is a population PK model* previously published[13], but which we re-implemented in the *R Nimble* (Figure 2). It was built for Bayesian inference on V_{ss}, K_a, and V_{max} population means ($\boldsymbol{\mu}$), SDs ($\boldsymbol{\Omega}$) and individual values, together with measurement error parameters, on the basis of the theophylline concentration and body mass data. Body mass data, when recorded, were considered as exactly known; when missing, they were imputed at their expected value (given the age range) of 80.58 kg. Age of all subjects was also imputed to the expected value of 27.6 years (see[13]). The statistical model we consider assume that the j^{th} theophylline plasma concentration measurement for the i^{th}subject was normally distributed with a mean, at time t_{ij}, given by the structural PBPK model, *f*, and residual error SD $\boldsymbol{\sigma}_1$:

$$y_{ij} \sim \text{Normal}\left(f\left(t_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\varphi}\right), \sigma_1\left(f\left(t_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\varphi}\right), b_0, b_1\right)\right), \quad 1 \le i \le N, \; 1 \le j \le n_i \qquad (20)$$

where $\boldsymbol{\theta}_i$ was a vector of estimated subject-specific parameters (V_{ssi}, K_{ai}, V_{maxi}), $\boldsymbol{\varphi}$ the vector of fixed parameters (square nodes other than Y_i in Figure 2), and (b_0, b_1) are residual error model parameters (also estimated).

At the population level, the subject-specific parameters were assumed to be distributed according to a multivariate log-normal distribution, with population geometric means $\boldsymbol{\mu}$ (estimated) and population covariance matrix $\boldsymbol{\Omega}$ in log-space:

$$\log(\boldsymbol{\theta}_i) \sim \text{Normal}(\log(\boldsymbol{\mu}), \boldsymbol{\Omega}) \tag{21}$$

The priors for V_{ss}, K_a, and V_{max} population means were uniform:

$$V_{max} \sim \text{Uniform}(0.05, 50) \tag{22}$$

$$K_a \sim \text{Uniform}(0.2, 40) \tag{23}$$

$$V_{ss} \sim \text{Uniform}(0.05, 10) \tag{24}$$

The priors for the population geometric SDs of K_a, V_{ss}, and V_{max} were set to normal distributions with mean 1.1, SD 0.3 and truncation bounds 1 and 4.0 (corresponding to a half-normal for the population SDs in log-space).

The residual error model, $\boldsymbol{\sigma}_1$, describes modeling and measurement errors and was chosen to be proportional to model predictions with an added constant error:

$$\sigma_1\big(f\big(t_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\varphi}\big), b_0, b_1\big) = \Big(b_0 + b_1 \times f\big(t_{ij}, \boldsymbol{\theta}_i, \boldsymbol{\varphi}\big)\Big)^2 \tag{25}$$

The prior for b_0 was half-normal with a SD of 5, and the prior for b_1 half-normal with a SD of 0.1, as in[13].

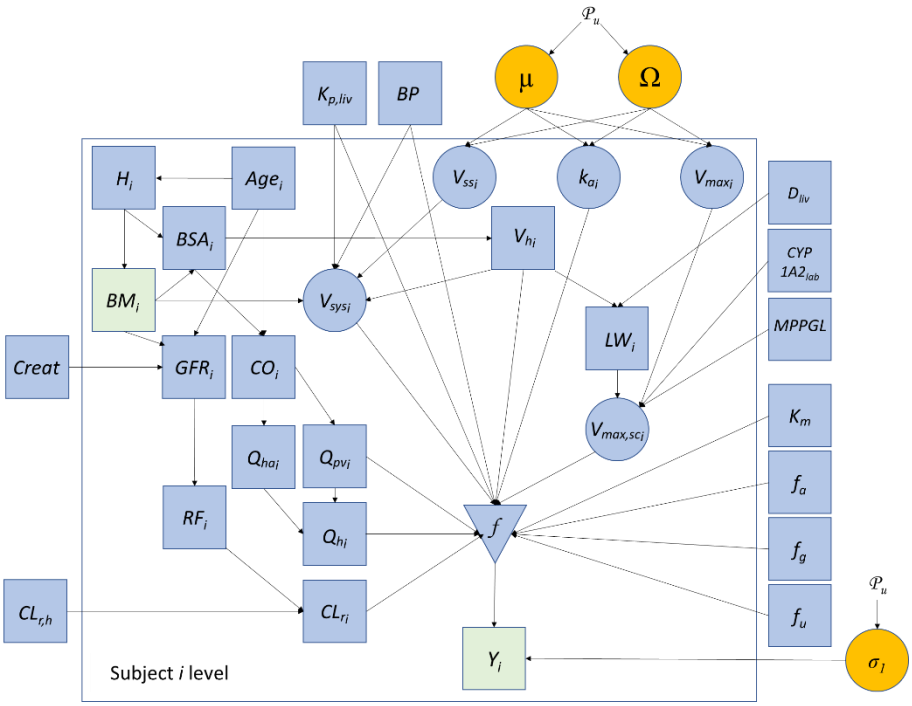

Figure 2: Population PK parameter dependency graph (model *B*) for Bayesian inference on the minimal PBPK model of theophylline pharmacokinetics in humans. P_u are prior distributions; Square nodes have fixed values; Green nodes are data values (missing BM_i data were imputed); Round orange nodes have priors. Round blue nodes are stochastic dependent parameters; Triangular node *f* is the structural ODE model.

Model *C is an improved population PK model* in which population inference is performed on V_{ss}, K_a, and V_{max}, together with theophylline concentration measurement error parameters, as above, but also on subjects' ages and body masses (Figure 3). No population hierarchy was put on age, but subjects' age was assigned a uniform prior, to reflect the imprecise information we have on it[36]:

$$Age_i \sim \text{Uniform}(21,31) \tag{26}$$

Body mass, when recorded, was considered as data, normally distributed around the true subject's body mass BM_i (dependent on the subject's height, Ht_i) with constant measurement error SD $\boldsymbol{\sigma}_2$ (set at 0.5 kg):

$$BM_{obs_i} \sim \text{Normal}(BM_i, \sigma_2) \tag{27}$$

$$BM_i \sim \text{Normal}\left(\mu_{BM_i}, 0.15 \times \mu_{BM_i}\right) \tag{28}$$

$$\mu_{BM_i} = \exp(2.643 + 0.0099 \times Ht_i) \tag{29}$$

In the case of missing data on body mass, BM_i was sampled conditionally to height (Eq. 28). Note that subject height is also sampled and updated in this model.

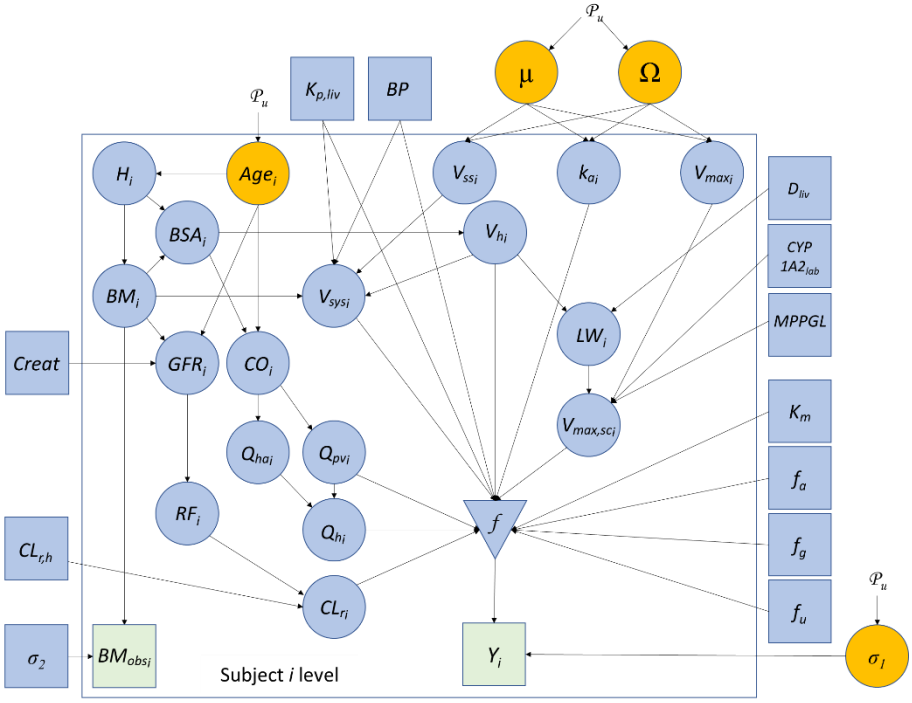

Figure 3: Improved population PK parameter dependency graph (model *C*) for Bayesian inference on the minimal PBPK model of theophylline pharmacokinetics in humans. P_u are prior distributions; Square nodes have fixed values; Green nodes are data values; Round orange nodes have priors. Round blue nodes are stochastic dependent parameters; Triangular node *f* is the structural ODE model.

Model *D* uses stationary sampling for all top parent parameters previously fixed at default values, except D_{liv} and σ_2 (Figure 4). Stationary sampling implies that those ten nodes were always sampled from the following informative priors with about 5% CV, which reflect our strong beliefs in their values:

$$K_{p,liv} \sim \text{Normal}(1, 0.05) \tag{30}$$

$$BP \sim \text{Uniform}(0.75, 0.9) \tag{31}$$

$$CYP1A2_{lab} \sim \text{Normal}(52,2.6) \tag{32}$$

$$MPPGL \sim \text{Normal}(39.79,2) \tag{33}$$

$$K_m \sim \text{Normal}(377,20) \tag{34}$$

$$f_a \sim \text{Uniform}(0.9,1) \tag{35}$$

$$f_g \sim \text{Uniform}(0.9,1) \tag{36}$$

$$f_u \sim \text{Normal}(0.5,0.025) \tag{37}$$

$$Cl_{r,h} \sim \text{Normal}(0.31,0.015) \tag{38}$$

$$Creat \sim \text{Normal}(76.5,4) \tag{39}$$

Two versions of the stationary sampler were used: one-at-a-time node sampling and block sampling of the above parameters. All the other nodes were sampled as in model *C*.

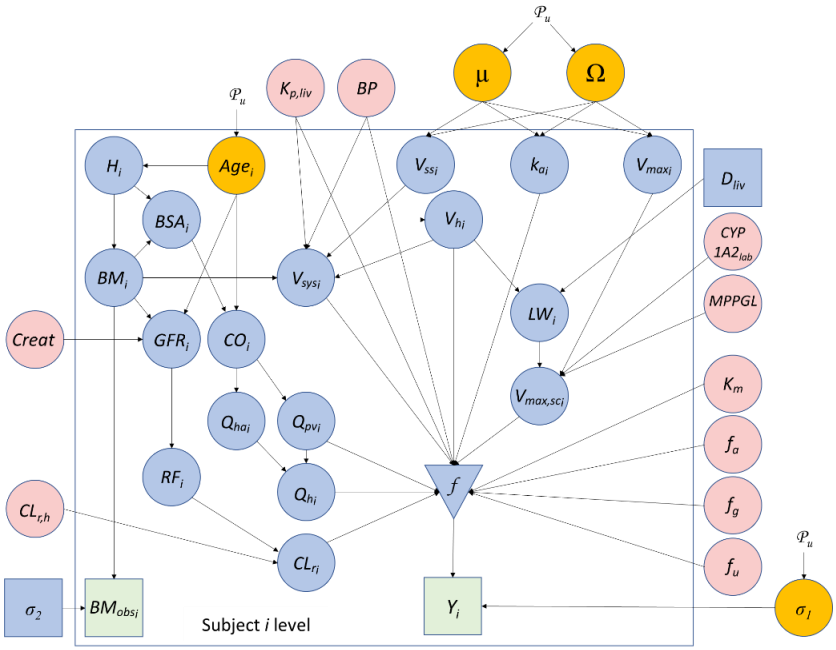

Figure 4: Model *D* parameter dependency graph for Bayesian inference on the minimal PBPK model of theophylline pharmacokinetics in humans. Pink nodes are sampled from their priors using a stationary sampler; P_u are prior distributions; Square nodes have fixed values; Green nodes are data values; Round orange nodes have priors. Round blue nodes are stochastic dependent parameters; Triangular node f is the structural ODE model

Model *E* performs fully Bayesian inference on all stochastic nodes of the previous model by simply replacing the stationary samplers (by the default Metropolis-Hasting random walk sampler provided by *Nimble*, therefore turning the pink nodes in Figure 4 into blue nodes. Two versions of random walk sampler provided by *Nimble* were used: one-at-a-time node sampling of all nodes and a mix of block sampling of the pink nodes in Figure 4 and individual sampling of the other nodes.

3.3 Simulation and Estimation Procedures

3.3.1 Monte Carlo simulations from prior distributions

Monte Carlo simulations are quite straightforward to run: Given the above basic dependency model *A*, the parent random nodes are sampled, and all the dependent nodes are updated in topological order. Ten thousand subjects were simulated. Theophylline dose administered was set at 1776 μmol (320 mg).

3.3.2 MCMC simulations for statistical inference

When updating parameter distributions on the basis of data, the experimental conditions described above were simulated. For model *B*, four chains of 10,000 iterations each were simulated with the default Metropolis-Hastings random walk sampler provided by *Nimble* (model *D* also used our custom stationary sampler for some nodes, see above). The initial 5,000 samples were discarded, forming a final sample of 20,000 posterior parameter vectors, thinned down to 4,000 vectors by keeping only one five of them. For models *C* to *E*, the number of iterations was extended to 55,000 (the first 5,000 still discarded). After thinning, 10,000 were kept. The convergence of the chains was assessed visually and confirmed using Gelman and Rubin [35] potential scale reduction factor $\hat{R}$; values of $\hat{R}$ lower than 1.05 indicated well converged chains. The C and *Nimble* R codes, as well as data files used are given as

Supplemental Material. The prior distributions used for the various models are given in the model equations above.

3.4 Software used

All simulations were performed with the *R* package *Nimble* [34, 38] under *R* version 4.0.5[39]. The *Lsoda* implicit solver of the R package *deSolve* was used to integrate the structural model, with relative and absolute tolerances of 10^{-8}. Graphs and additional statistical treatment were performed in *R*. Software was run on a Dell Precision 5540 Notebook computer, with 16GB RAM and 6 Intel Core i7-9850H processors clocked at 2.6 GHz (for models *A* and *B*), and on a Hewlett-Packard-Z420 Workstation computer with 12 Intel Xeon E5-1650 v2 processors clocked at 3.5 GHz for the other models. The ratio of workstation over laptop run times for the same computation is about 1.15 and the timing results given below for model *A* and *B* were multiplied by that ratio for fair comparison with the other models.

4 Results

4.1 Monte Carlo simulations from prior distributions (model A)

Figure 5 shows histograms of the parameter values sampled using the basic dependency model *A*. Simulations for 10,000 subjects took about one minute to run, with a non-compiled model. The samples obtained match closely the assigned prior distributions. Figure 6 displays a set of representative parameter correlations implied by that dependency model (the full correlations plot is too large to display usefully). Note that some correlations coefficients are equal to one, because the corresponding dependencies are purely deterministic.

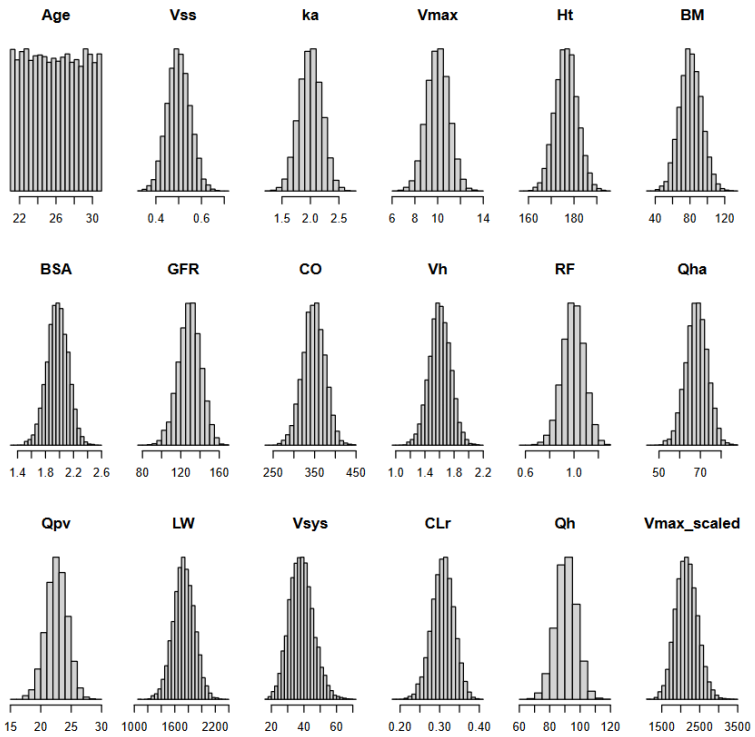

Figure 5: Histograms of the minimal PBPK parameter values sampled in 10,000 Monte Carlo simulations from their prior distributions, as specified in the basic dependency model *A*.

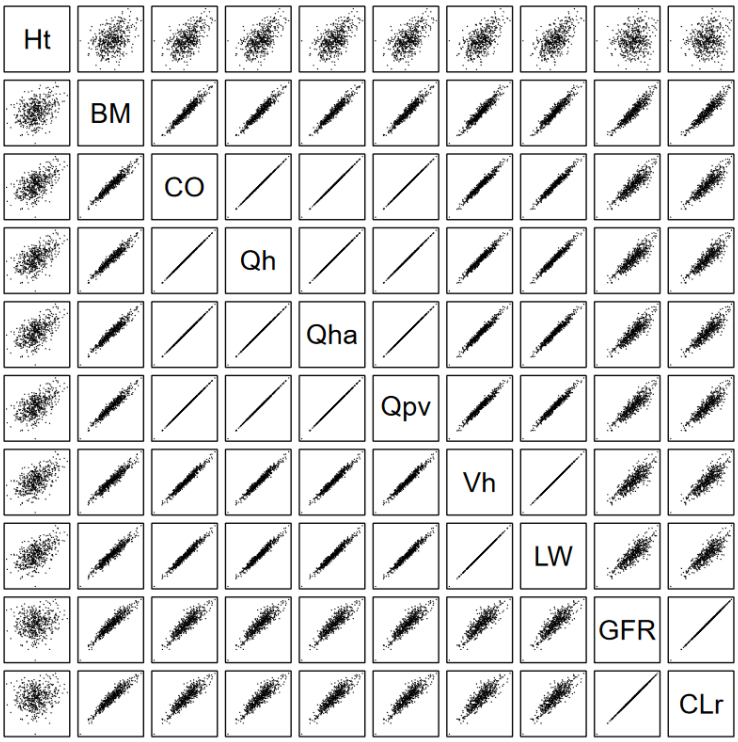

Figure 6: Representative prior correlations between PBPK parameters induced by the basic dependency model in a sample of 10,000 Monte Carlo generated virtual subjects.

4.2 MCMC simulations with most parameters fixed (model B)

For those simulations, subjects' age and body weight were held constant, as previously[13]. We therefore expect equivalent results, since the only difference is an implementation with *Nimble*. Ten thousand iterations of four MCMC chains led to an $\hat{R}$ convergence diagnostic lower than 1.02 for all parameters and ran in 15 minutes when chains were run in series, which is intermediate between the Stan and *GNU MCSim* run times. A comparison of the posterior distributions of the population parameters of interest in our original work (V_{ss}, K_a, and V_{max}) is shown in Figure 7 (summaries of the posteriors for population and individual parameters are given in Table S1). Central estimates are not different from the other implementation, and only K_a population parameters seem to have a slightly larger variance; however, K_a is the most difficult parameter to estimate at the subject level with those data. The data fits obtained at the individual levels were similar to those achieved with the other software (Figure S1).

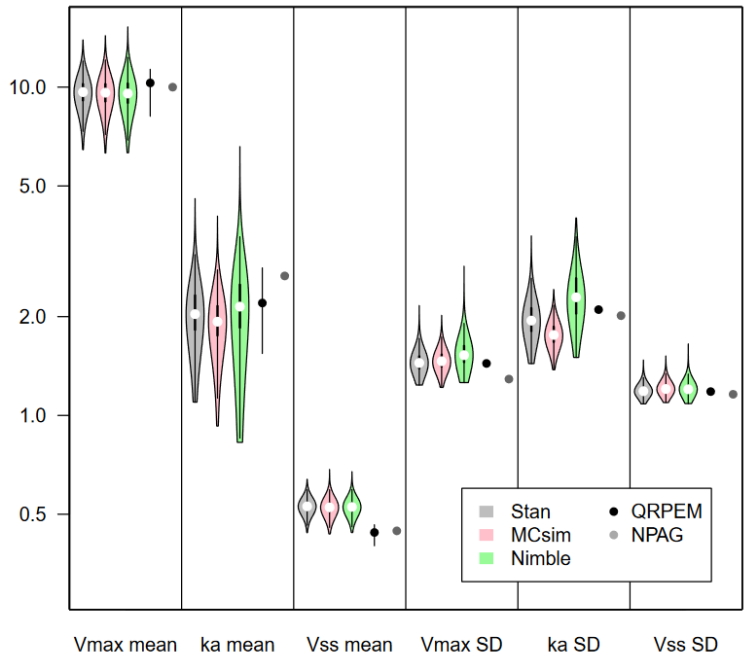

Figure 7: Violin plots of the population parameter estimates for V_{max}, K_a, and V_{ss} with model *B* (subjects' age and body weight held constant) and different software platforms. The *Simcyp*® QRPEM, NPAG, *Stan* and *GNU MCSim* implementations are described in[13].

4.3 Model C

How does our inference change if we model explicitly the uncertainty in body mass and age and consider them as random variables? In that case, fifty thousand iterations of four MCMC chains led to an $\hat{R}$ convergence diagnostic lower than 1.05 for all parameters and ran in 180 minutes when run in series. The data fits obtained at the individual levels were similar to those achieved with model *B* (Figure S2). Summaries of the posteriors for population and individual parameters are given in Table S2. The posterior distributions of individual *BM* are plotted in Figure 8. Expectedly, for the first twelve subjects, *BM* is well-identified by its reasonably precise measurement. For the remaining subjects, it turns out that *BM* is not strongly constrained by its parent parameters (including age, see Figure 3) or by the theophylline concentration data, so posterior uncertainty about it is large. Subject's ages are even less constrained by the data and have posteriors identical to their priors (uniform between 21 and 31 years) (see Table S2). Age is "shielded" from the *BM* data by subject's height, which has a specific distribution and variance. The individual estimates of V_{ss}, K_a, and V_{max} do not change much for the individuals with precise *BM* data. However, for the subjects with missing *BM* data, those estimates are more imprecise and shifted by shrinkage toward the population means (Figure 9 and Figure S3).

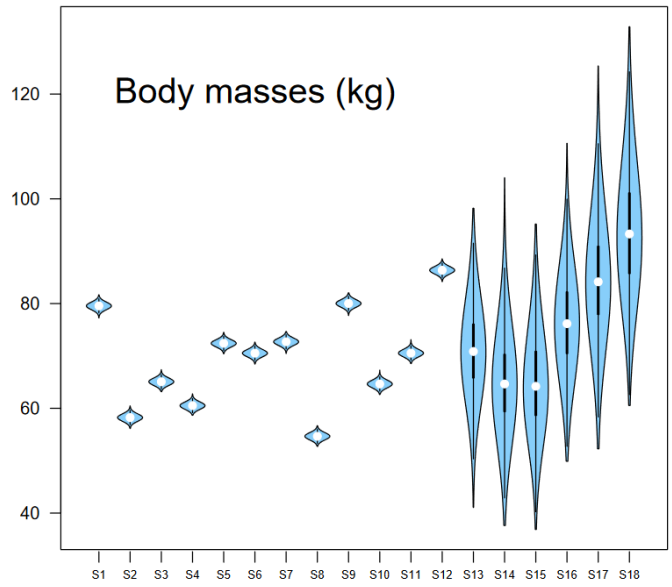

Figure 8: Violin plots of the individual parameter estimates of body mass (*BM*) with model *C* (accounting for subjects' age and *BM* uncertainty). Data on BM is missing for the last six subjects, hence the large posterior uncertainty on them.

Our interest resides mostly in the posterior distributions of V_{ss}, K_a, and V_{max} population parameters for which a comparison of models *B* and *C* estimates is shown in Figure 10. In terms of central estimates, the three population means and the population SD of k_a have increased by about 2%; V_{max} and V_{ss} populations SDs are decreased by about 5%. The precision of V_{max} and V_{ss} population means and population SDs estimates has increased by 5 to 10%; the precision of k_a population mean and SD is decreased by 4 to 7%. These effects are mostly due to the fact that parameter estimates for subjects with missing body mass have moved toward the population means and are more uncertain.

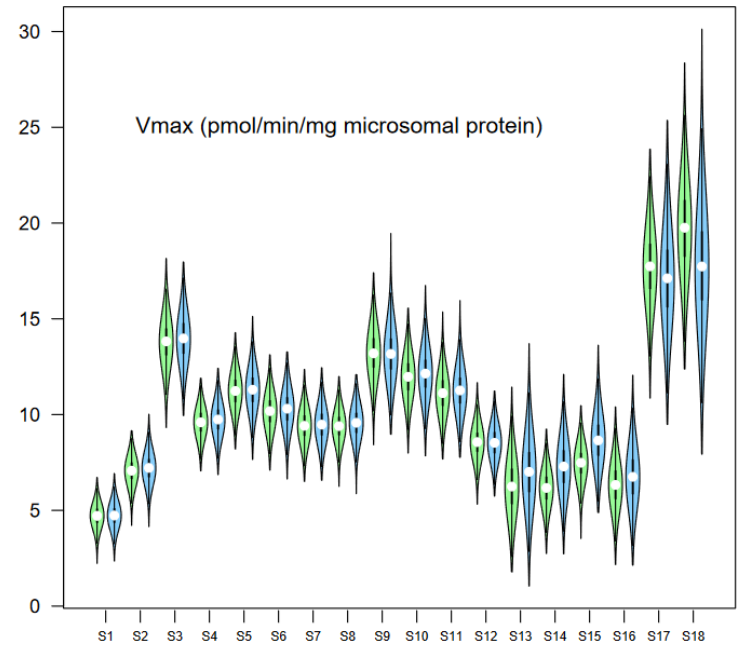

Figure 9: Comparison of violin plots of the individual parameter estimates of V_{max} with model *B* (in green) and model *C* (in blue; accounting for subjects' age and *BM* uncertainty).

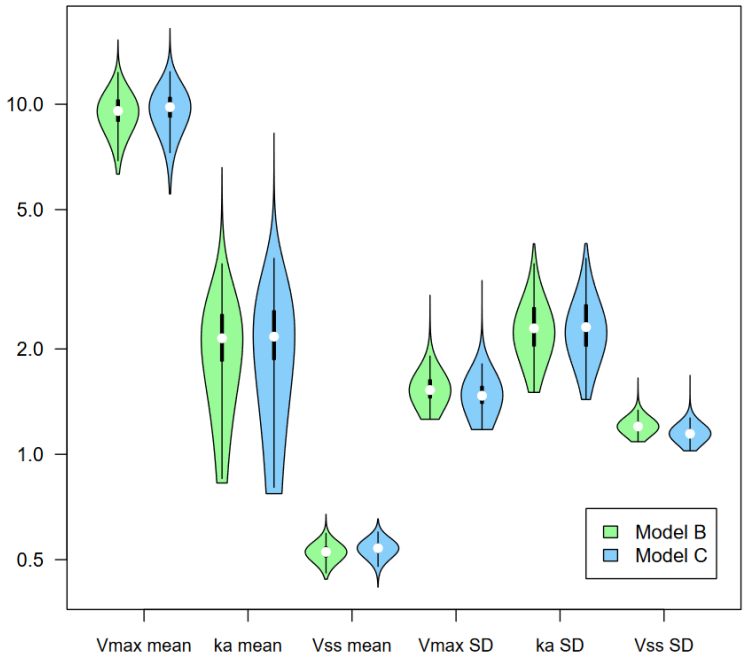

Figure 10: Comparison of model *B* (fixed subjects' age and BM) and model *C* (uncertain subjects' age and *BM*) V_{max}, K_a, and V_{ss} population parameter estimates.

4.4 Model D

Fixing most of the top model parameters (such as $K_{p,liv}$, *BP*, *etc.*) at their population average value is as arbitrary as fixing body weight and age, when we do in fact know that they vary between subjects. Model *D* relaxes that assumption and samples also those parameters during MCMC simulations, but from their prior distribution (using the stationary sampler). Two versions of the stationary sampler were investigated: a one parameter at a time sampling and a block sampling in which the ten new random nodes are jointly sampled. Fifty thousand iterations of four MCMC chains led to an $\hat{R}$ convergence diagnostic lower than 1.05 for all parameters with both versions of the MCMC sampling. Despite the increased parameter space, convergence was not slower than for model *D* compared to model *C* (see Figure S4), but it was not faster either. Four chains in series ran in 500 minutes (8.35 hours) when updating one parameter at a time and only 200 minutes (3.3 hours) for block sampling, which is just a little slower than for model *C*. The data fits obtained at the individual levels were on average similar to those achieved with model *C* (Figure S5), but with a larger uncertainty (error model parameters b_0 and b_1 are inflated and more uncertain than in other models, see Table S3) (because individual parameter estimates are more uncertain, as shown next).

Summaries of the posteriors for population and individual parameters are given in Table S3 for one at a time sampling and in Table S4 for block sampling. There are no significant differences between the results of the two sampling methods and the parameters sampled from their priors (with the stationary sampler) have the expected distribution (Figure S6).

A comparison of models *B* and *D* population parameters estimates of V_{ss}, K_a, and V_{max} is shown in Figure 11. Individual posterior estimates are shown in Figure S7. Allowing for uncertainty in the previously fixed top parameters significantly increased the uncertainty on *all* individual parameter estimates but had a dampened effect on the three population means and population

SDs of interest. The central estimates of the population means were changed by 5 to 10%; the SDs are not much shifted, but the precision of all population parameter estimates decreased by 5 to 13%. Here also the estimates obtained with the two sampling methods do not differ. Block sampler is just faster.

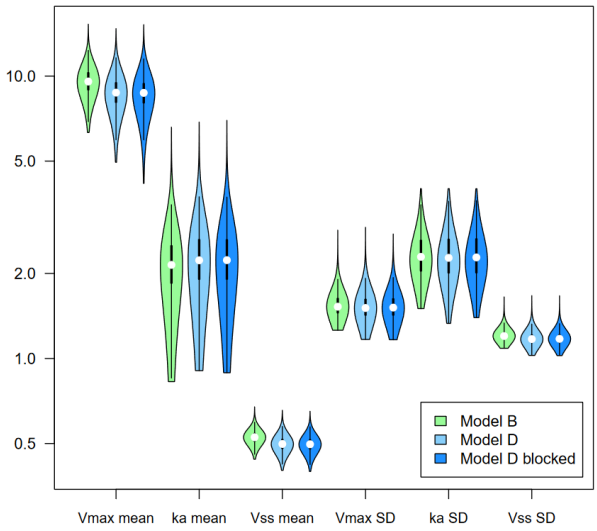

Figure 11: Comparison of model *B* (fixed subjects' age and BM) and model *D* (uncertainty in most of the top parameters, with stationary sampling) V_{max}, K_a, and V_{ss} population parameter estimates.

4.5 Model E

Model *E* goes one step beyond model *D* and updates all parameter values from their joint posterior distribution by random-walk (Metropolis-Hastings) MCMC simulations. Here also, two versions of the random-walk sampler were investigated: one parameter at a time sampling and block sampling (of the ten pink nodes in Figure 4). Fifty thousand iterations of four MCMC chains led to an $\hat{R}$ convergence diagnostic lower than 1.05 for all parameters with both versions of the MCMC sampling. They ran, in series, in 465 minutes (7.75 hours) when updating one parameter at a time and 192 minutes (3.2 hours) for block sampling, which is close to model *C* run time. The data fits obtained at the individual levels were on average similar to those achieved with model *D* (Figure S8), but with a lower uncertainty (because individual parameter

estimates are more uncertain, as shown next). The posterior distributions of the ten top parameters parameter which were sampled with the stationary sampler in model *D* are identical to their prior, except for *BP*, f_a, f_g, and $K_{p,liv}$, which have been updated, showing that information was available about those parameters in the theophylline kinetic data (Figure S9). The update is clear for BP, which is shifted down; Parameters f_a and f_g are shifted slighly up; $K_{p,liv}$, is very slighly moved down. Statistical summaries of the posteriors for population and individual parameters are given in Table S5 for one at a time sampling and in Table S6 for block sampling. Here again, there are no significant differences between the results of the two sampling methods.

A comparison of models *B* and *E* population parameters estimates of V_{ss}, K_a, and V_{max} is shown in Figure 12. Individual posterior estimates are shown in Figure S10. As for model *D*, allowing for uncertainty in the previously fixed top parameters significantly increased he uncertainty on individual parameter estimates but had a dampened effect on the three population means and population SDs of interest. Figures S11 and S12 show comparisons of model *D* and model *E* parameter estimates (estimates are not much shifted in model E, compared to model D, and precisions are slightly more affected).

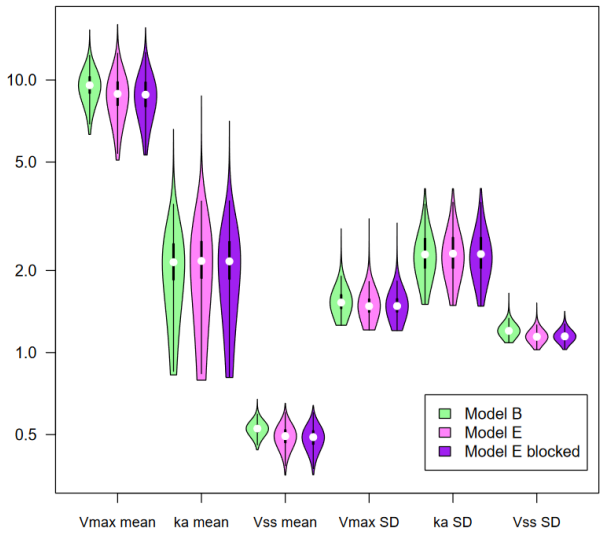

Figure 12: Comparison of model *B* (fixed subjects' age and BM) and model *E* (uncertainty in most of the top parameters, with fully Bayesian updating) V_{max}, K_a, and V_{ss} population parameter estimates.

5 Discussion

To our knowledge this is the first time that an in-depth Bayesian exploration of different treatments of parameter dependencies and updating, including the use of a stationary MCMC sampler, have been investigated. We used a PBPK model and made heavy used of informative priors grounded in literature meta-analyses. A previous publication[10] (which also used theophylline as an example) discussed updating a generic covariate model based on age, sex and body height, but the priors used were not meant to be informative. It seems that assigning informative priors to known covariate effects is never done in usual population pharmacokinetic modeling. This probably due to the fact that the empirical covariate models commonly developed cannot be used beyond the context of the applications for which they were developed. The problem of non-reusable analyses is that like Sisyphus, we endlessly roll the same boulder uphill[40].

The new MCMC sampler we introduce allow an assessment of the impact on inference of uncertainty in many "ancillary" parameter values at reduced computational cost. This way,

inference can focus on the most sensitive parameters without forgetting uncertainty about the other ones. All parameters have to be sampled together so that the distribution of ancillary parameters can influence the distribution of the primary target ones. The key is to not update the distribution of those other parameters (sampling them always from their prior distributions) and to use block sampling for speed. Block sampling in that case is easy, because most of the times parameter priors are either independent, or have known joint distributions and covariances, which removes the problem of finding an approximately correct covariance structure for the proposal kernel distribution. Our approach is akin to the idea of breaking up inference between the pharmacokinetic and the pharmacodynamic parts of a linked model, as discussed in[41]. That deviates from strict Bayesian inference, which would update the joint distribution of all parameters, and is only approximate.

Nimble can be used to easily perform Monte Carlo simulations from prior distributions (model *A*). Those are very fast, because even though *Nimble* is an *R* package, the models are samplers can be compiled. Figure 6 illustrates the covariances which are encoded in the parameter dependency model of our minimal PBPK model. Relationships between parameters range from complete independence to deterministic functions.

Bayesian inference on the population and individual values of V_{max}, K_a, and V_{ss} using model *B*, in which subjects' ages and non-measured body masses are imputed, led to posterior distributions practically identical to those from Wedagedera *et al*.[13], small differences being due to random sampling. This simply shows that the implementation of the model used by Wedagedera *et al*. was consistently implemented in *Nimble*.

Accounting for uncertainty in subjects' ages and body masses in our inference (model *C*) led to inflated uncertainty about individual estimates of V_{max}, and to a lesser extent for K_a, and V_{ss}, for the least six subjects for which body mass information was missing. This translated into an increased shrinkage of the individual estimates toward the population means, and decreased

population variances (subjects being more “homogeneous”). The effects on population parameters are rather small though, and the improvements made in model *C* mostly affect inference about individual values (which could have an impact in precision dosing for those subjects, for example).

Model *D* takes an innovative approach to account for the uncertainty or variability of ten other top parent parameters in our inference. The results show increased uncertainty about all individual values of V_{max}, K_a, and V_{ss} (not just for the individuals with missing body mass data) and their gathering, by shrinkage, closer to the population mean. Here again, the populations means and variances are less affected, even though their uncertainty increases. These effects similar to those seen with model *C*, just amplified.

Model *E* is a fully Bayesian version of model *D* in which the distributions of all random variables are updated. Its results are practically the same as those of model D, except that some but not much, information is recovered on four parameters (*BP*, f_a, f_g, and $K_{p,liv}$,) that model D keeps at their prior.

In terms of computation speed, model *B*’s implementation in Nimble is slightly slower than the *GNU MCSim* implementation[13], which is very fast, but less flexible. Model *B* updates about 60 parameter distributions and one iteration requires about 60 calls to the PKPK model solver. The added complexity of inferring about age and missing body masses (model *C*) slows down computations by a factor ten (mostly because more parameters were sampled and five time as many simulations were required to reach convergence), bringing a clock time of 200 minutes for four simulated Markov chains, without parallelization. Obviously, parallelization of the chains would bring the clock time down to a bit less than an hour. For model *D*, one at a time sampling of ten additional parameters more than doubled the simulation time (bringing it to 500 minutes, because about 250 calls to the PBPK solver were required per iteration). However, with the same model, block sampling reduced the computation time to a bit more than 200

minutes (because updating the block of parameter value took 10 times less calls to the PBPK model than updating each of its components). Therefore, block sampling of stationary parameter comes at almost not cost for such a model. Note also that stationary samplers are immediately at convergence. Finally, the fully Bayesian model *E*, either in its blocked or non-blocked versions, did not take more time to compute than model *D*. This can be explained by the fact that it did not require more call to the PBPK model, and because not much information was available from the data, making convergence of the added parameters very fast. So, generally speaking, in a Bayesian framework with informative priors, if structural model solving is relatively burdensome, it does not really hurt to add relatively insensitive parameters into the inference, because they can be very efficiently sampled as a block. This is contrary to common wisdom about the curse of dimensionality. Given this, the advantage of stationary sampling over full Bayesian sampling is not so much about time than about preserving posterior distributions from the vagaries of specific datasets and eventual model misspecification.

A question is then how to choose the parameters to include in a stationary sampler? That is related to the question of which parameters to include in inference, when many are at hand (the standard in PBPK models). A generic approach would be to rank parameters by order of sensitivity (in principle, sensitivity of the posterior distribution of those parameters, as assessed by global sensitivity analysis, taking prior uncertainty into account); select the *N* most sensitive parameters and update them the Bayesian way; for the remaining parameters, do stationary sampling or leave them fix (in theory, we would like to sample only the parameters whose uncertainty has an impact on the inference about the *N* most sensitive parameters; in practice this is an area of our current research). However, more general principles can also be applied: PBPK model include purely physiological parameters, purely drug-specific parameters, parameters characterizing drug-body interactions, and trial design parameters. For physiological parameters, prior knowledge is abundant, and we tend to have strong population

priors with little uncertainty on the population means and variances. Since we may not want those to be altered by the vagaries of particular datasets, the stationary sampler seems perfectly suited. However, there may be cases where information about such parameters can be extracted from the data. We think that here, a separate analysis should be performed to update the previous parameter dependency model. We may also have a fixed physiological parameter whose value conflicts with the data of a particular clinical trial. For example, in the above case study, body masses are supposed to depend on subjects' age and height. Since we only have an age range and no data on height, the uncertainty on individual body masses is very large. That shows clearly for the six subjects with missing body mass data. But for the 12 subjects for which we have data, it seems that body mass variability is somewhat lower than in a general healthy volunteer population. Therefore, it would make sense in that case to alter our covariate model and allow for information on the first 12 subjects to transfer to the last 6 subjects *via* a hierarchical model.

Purely drug-specific parameters have usually strong priors, amounting to constants (for molecular weight, for example) but in some cases their values are obtained by QSAR models with significant uncertainty. It may make sense in that case to update their values to describe the PK data at hand. Finally, the parameters characterizing drug-body interactions are usually the ones targeted for updating, even if *in vitro* experiments may provide informative priors on them. Note, however, that given our parameter dependency model, we could have decided for example to fit the MPPGL parameter (milligram proteins per gram of liver) (because it conditions the scaled V_{max}) instead of fitting V_{max} to describe the clearance phase of the data. Both approaches would be expected to work equally well, but the prior on MPPGL is much tighter than the prior on V_{max} and we would in that case expect conflict between prior and data. A general rule is clearly to calibrate the most uncertain parameters. Finally, trial design

parameters should in general not be fitted, unless they are missing (which could happen in the case of censoring or unusual study design).

Note that the ten nodes introduced in Models *D* and *E* were modeled as fixed effects at the population level. They could have been introduced in a population model, with random effects. In fact, Simcyp® consider explicitly that $CYP1A2_{lab}$, *MPPGL*, f_a, and *Creat* have population variability and $CL_{r,h}$ is explicitly a population parameter. So, there is flexibility in implementing such additions. Similarly, one could introduce inter-occasion variability in this framework.

Lastly, we found no evidence that stationary sampling, by augmenting the size of the parameter space, leads to faster convergence of the overall MCMC sampling, even though it increases the sampling space dimension and should smoothen density gradients somewhat.

If we step back and consider the implications of our discussion in terms of bottom-up (mechanistic) *vs*. top down (data modelling approaches), the statistical problems they face are very different. Modelling strategies are very different and so are calls to parsimony: step by step, or more precisely sub-process by sub-process, in a complex model, more globally in a data-description oriented approach (which, we know, have limited explanation and prediction power). What is very different in mechanistic bottom-up approaches is the availability of prior information. Data-oriented modelling has little prior information available and tends to rely more heavily on maximum likelihood methods. Complex models may have thousands on parameters, but they also recruit information from thousands of experiments. In their case, Bayesian approaches make probably most sense[42–44]. We have shown here that informative prior distributions can dispel the curse of dimensionality.

We illustrated our reflections with a relatively simple structural and covariate model, using a fixed dependency structure and informative priors on parameter values. However, we expect

our conclusions, in particular about the behavior of the stationary sampler, to be generalizable to much more complex PBPK models and quantitative systems pharmacology models.

Author contributions

HJ, FB designed the research. VHT, HM, FB performed the research. VHT, HM, HJ, FB, MJ wrote the manuscript.

ORCID

Hiroshi Momiji https://orcid.org/0000-0002-3015-5770

Masoud Jamei https://orcid.org/0000-0002-3443-0194

Frederic Y. Bois https://orcid.org/0000-0002-4154-0391

References

1. Steenland K, Greenland S (2004) Monte Carlo sensitivity analysis and Bayesian analysis of smoking as an unmeasured confounder in a study of silica and lung cancer. American Journal of Epidemiology 160:384–392. https://doi.org/10.1093/aje/kwh211

2. de Vocht F, Kromhout H, Ferro G, et al (2009) Bayesian modelling of lung cancer risk and bitumen fume exposure adjusted for unmeasured confounding by smoking. Occupational and Environmental Medicine 66:502–508. https://doi.org/10.1136/oem.2008.042606

3. Rostami-Hodjegan A, Amin AM, Spencer EP, et al (2004) Influence of dose, cigarette smoking, age, sex, and metabolic activity on plasma clozapine concentrations: a predictive model and nomograms to aid clozapine dose adjustment and to assess compliance in individual patients. Journal of Clinical Psychopharmacology 24:70–78. https://doi.org/10.1097/01.jcp.0000106221.36344.4d

4. Salem F, Abduljalil K, Kamiyama Y, Rostami-Hodjegan A (2016) Considering age variation when coining drugs as high versus low hepatic extraction ratio. Drug Metabolism and Disposition 44:1099–1102. https://doi.org/10.1124/dmd.115.067595

5. Mostafa S, Polasek TM, Bousman C, et al Delineating gene–environment effects using virtual twins of patients treated with clozapine. CPT: Pharmacometrics and Systems Pharmacology in press:

6. Dartois C, Brendel K, Comets E, et al (2007) Overview of model-building strategies in population PK/PD analyses: 2002–2004 literature survey. British Journal of Clinical Pharmacology 64:603–612. https://doi.org/10.1111/j.1365-2125.2007.02975.x

7. Hutmacher MM, Kowalski KG (2015) Covariate selection in pharmacometric analyses: a review of methods: covariate selection in pharmacometric analysis. British Journal of Clinical Pharmacology 79:132–147. https://doi.org/10.1111/bcp.12451

8. Bois FY, Jamei M, Clewell HJ (2010) PBPK modelling of inter-individual variability in the pharmacokinetics of environmental chemicals. Toxicology 278:256–267

9. Tsamandouras N, Rostami-Hodjegan A, Aarons L (2013) Combining the 'bottom up' and 'top down' approaches in pharmacokinetic modelling: fitting PBPK models to observed clinical data. British Journal of Clinical Pharmacology 79:48–55. https://doi.org/10.1111/bcp.12234

10. Krauss M, Tappe K, Schuppert A, et al (2015) Bayesian population physiologically-based pharmacokinetic (PBPK) approach for a physiologically realistic characterization of interindividual variability in clinically relevant populations. PLoS One 10:e0139423. https://doi.org/10.1371/journal.pone.0139423

11. Bois FY, Hsieh N-H, Gao W, et al (2020) Well-tempered MCMC simulations for population pharmacokinetic models. Journal of Pharmacokinetics and Pharmacodynamics 47:543–559. https://doi.org/10.1007/s10928-020-09705-0

12. Hsieh N-H, Bois FY, Tsakalozou E, et al (2021) A Bayesian population physiologically based pharmacokinetic absorption modeling approach to support generic drug development: application to bupropion hydrochloride oral dosage forms. Journal of Pharmacokinetics and Pharmacodynamics 48:893–908. https://doi.org/10.1007/s10928-021-09778-5

13. Wedagedera JR, Afuape A, Chirumamilla SK, et al (2022) Population PBPK modeling using parametric and nonparametric methods of the Simcyp Simulator, and Bayesian samplers. CPT: Pharmacometrics and Systems Pharmacology 11:755–765. https://doi.org/10.1002/psp4.12787

14. Bois FY, Brochot C (2016) Modeling pharmacokinetics. In: Benfenati E (ed) In Silico Methods for Predicting Drug Toxicity. Springer New York, New York, NY, pp 37–62

15. Lin L, Wong H (2017) Predicting oral drug absorption: mini review on physiologically-based pharmacokinetic models. Pharmaceutics 9:41. https://doi.org/10.3390/pharmaceutics9040041

16. Glassman PM, Balthasar JP (2019) Physiologically-based modeling of monoclonal antibody pharmacokinetics in drug discovery and development. Drug Metabolism and Pharmacokinetics 34:3–13. https://doi.org/10.1016/j.dmpk.2018.11.002

17. Peters SA (2021) Physiologically Based Pharmacokinetic (PBPK) Modeling and Simulations: Principles, Methods, and Applications in the Pharmaceutical Industry, Second Edition, 1st ed. Wiley

18. Brown RP, Delp MD, Lindstedt ST, et al (1997) Physiological parameter values for physiologically based pharmacokinetic models. Toxicology and Industrial Health 14:407–484

19. Young JF, Branham WS, Sheenan DM, et al (1997) Physiological “constants” for PBPK models for pregnancy. Journal of Toxicology and Environmental Health 52:385–401

20. Watanabe K, Bois FY (1996) Interspecies extrapolation of physiological pharmacokinetic parameter distributions. Risk Analysis 16:741–754

21. McNally K, Cotton R, Hogg A, Loizou G (2015) Reprint of PopGen: A virtual human population generator. Toxicology 332:77–93. https://doi.org/10.1016/j.tox.2015.04.014

22. Jamei M, Dickinson GL, Rostami-Hodjegan A (2009) A framework for assessing inter-individual variability in pharmacokinetics using virtual human populations and integrating general knowledge of physical chemistry, biology, anatomy, physiology and genetics: a tale of “bottom-up” vs “top-down” recognition of covariates. Drug Metabolism and Pharmacokinetics 24:53–75

23. Wakefield J, Bennett J (1996) The Bayesian modeling of covariates for population pharmacokinetic models. Journal of the American Statistical Association 91:917–927

24. Lunn DJ (2008) Automated covariate selection and Bayesian model averaging in population PK/PD models. Journal of Pharmacokinetics and Pharmacodynamics 35:85–100

25. Hartung N, Wahl M, Rastogi A, Huisinga W (2021) Nonparametric goodness-of-fit testing for parametric covariate models in pharmacometric analyses. CPT: Pharmacometrics and Systems Pharmacology 10:564–576. https://doi.org/10.1002/psp4.12614

26. Liu D, Li L, Rostami-Hodjegan A, et al (2020) Considerations and caveats when applying global sensitivity analysis methods to physiologically based pharmacokinetic models. The AAPS Journal 22:93. https://doi.org/10.1208/s12248-020-00480-x

27. Niederlova V, Modrak M, Tsyklauri O, et al (2019) Meta-analysis of genotype-phenotype associations in Bardet-Biedl syndrome uncovers differences among causative genes. Human Mutation 40:2068–2087. https://doi.org/10.1002/humu.23862

28. Pearl J (1995) Causal diagrams for empirical research (with discussion). Biometrika 82:669–710

29. Dawid AP (2002) Influence diagrams for causal modelling and inference. International Statistical Review 70:161–189

30. Kjærulff UB, Madsen AL (2008) Bayesian Networks and Influence Diagrams. Springer New York, New York, NY

31. Pearce DJ, Kelly PHJ (2006) A dynamic topological sort algorithm for directed acyclic graphs. ACM Journal of Experimental Algorithmics 11:1.7

32. Metropolis N, Rosenbluth AW, Rosenbluth MN, et al (1953) Equation of state calculation by fast computing machines. Journal of Chemical Physics 21:1087–1092

33. Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57:97–109

34. de Valpine P, Turek D, Paciorek CJ, et al (2017) Programming with models: writing statistical algorithms for general model structures with NIMBLE. Journal of Computational and Graphical Statistics 26:403–413. https://doi.org/10.1080/10618600.2016.1172487

35. Gelman A, Rubin DB (1992) Inference from iterative simulation using multiple sequences (with discussion). Statistical Science 7:457–511

36. Trembath PW, Boobis SW (1980) Pharmacokinetics of a sustained-release theophylline formulation. British Journal of Clinical Pharmacology 9:365–369

37. Bates D, Maechler M, Bolker B (2019) MEMSS Package, https://cran.r-project.org/web/packages/MEMSS/index.html

38. NIMBLE Development Team (2022) NIMBLE: MCMC, Particle Filtering, and Programmable Hierarchical Modeling, https://cran.r-project.org, https://r-nimble.org

39. R Development Core Team (2013) R: A Language and Environment for Statistical Computing, http://www.R-project.org

40. Camus A (2012) [The Myth of Sisyphus]. Gallimard, Paris

41. Lunn D, Best N, Spiegelhalter D, et al (2009) Combining MCMC with "sequential" PKPD modelling. Journal of Pharmacokinetics and Pharmacodynamics 36:19–38

42. Gelman A, Bois FY, Jiang J (1996) Physiological pharmacokinetic analysis using population modeling and informative prior distributions. Journal of the American Statistical Association 91:1400–1412

43. Bernillon P, Bois FY (2000) Statistical issues in toxicokinetic modeling: a Bayesian perspective. Environmental Health Perspectives 108 (suppl. 5):883–893

44. Bois FY, Jamei M (2012) Population-based pharmacokinetic modeling and simulation. In: Lyubimov AV (ed) Encyclopedia of Drug Metabolism and Interactions. John Wiley & sons, Hoboken, pp 1–27

45. Jamei M, Marciniak S, Edwards D, et al (2013) The Simcyp population based simulator: architecture, implementation, and quality assurance. In Silico Pharmacology 1:9. https://doi.org/10.1186/2193-9616-1-9

46. Habib MP, Schifman RB, Shon BY, et al (1987) Evaluation of whole blood theophylline enzyme immunochromatography assay. Chest 92:129–131. https://doi.org/10.1378/chest.92.1.129

47. Howgate EM, Rowland Yeo K, Proctor NJ, et al (2006) Prediction of in vivo drug clearance from in vitro data. I. Impact of inter-individual variability. Xenobiotica 36:473–497. https://doi.org/10.1080/00498250600683197

48. Rowland Yeo K, Rostami-Hodjegan A, Tucker GT (2004) Abundance of cytochromes P450 in human liver: a meta-analysis. Br J Clin Pharmacol. 2004. British Journal of Clinical Pharmacology 57:687–688

49. Wilson ZE, Rostami-Hodjegan A, Burn JL, et al (2003) Inter-individual variability in levels of human microsomal protein and hepatocellularity per gram of liver: Microsomal protein and hepatocellularity. British Journal of Clinical Pharmacology 56:433–440. https://doi.org/10.1046/j.1365-2125.2003.01881.x

50. Ha H, Chen J, Freiburghaus A, Follath F (1995) Metabolism of theophylline by cDNA-expressed human cytochromes P-450. British Journal of Clinical Pharmacology 39:321–326. https://doi.org/10.1111/j.1365-2125.1995.tb04455.x

51. Lombardo F, Obach RS, Shalaeva MY, Gao F (2004) Prediction of human volume of distribution values for neutral and basic drugs. 2. Extended data set and leave-class-out statistics. Journal of Medicinal Chemistry 47:1242–1250. https://doi.org/10.1021/jm030408h

52. Jamei M, Marciniak S, Feng KR, et al (2009) The Simcyp population-based ADME simulator. Expert Opinion on Drug Metabolism & Toxicology 5:211–223

53. US Centers for Disease Control, National Center for Health Statistics National Health and Nutrition Examination Survey (NHANES, www.cdc.gov/nchs/nhanes.htm)

Supplemental Material for

Bayesian Dependency Modelling and Inference for PBPK Model Parameters: A Nimble Approach

Frederic Y. Bois, Thanh Vo Hong, Hiroshi Momiji, Heeseung Jo, Masoud Jamei, Amin Rostami-Hodjegan.

CERTARA UK Limited, Simcyp division, Sheffield, UK.

Contact: Frederic.Bois@certara.com

1. Plots

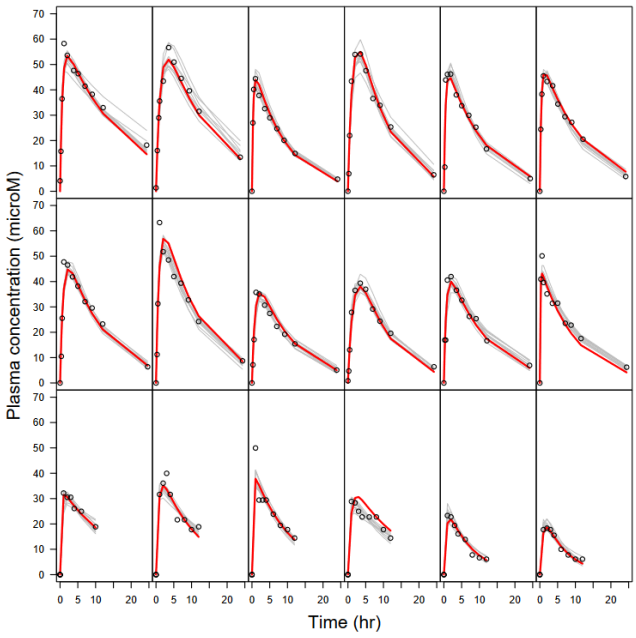

Figure S1: Individual posterior predictions for all subjects, using model *B*. The solid red curves are the individual predictions of theophylline plasma concentration time course; the circles are data points; the gray curves are ten time-course predictions using parameters estimates randomly drawn from their joint posterior distribution.

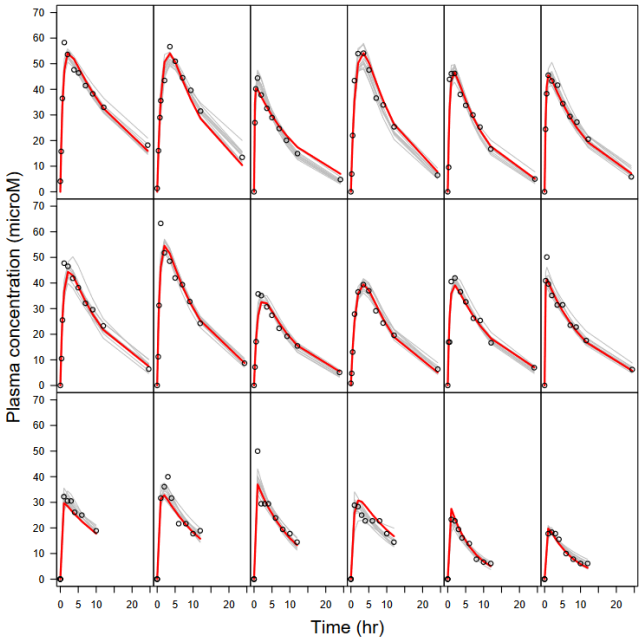

Figure S2: Individual posterior predictions for all subjects, using model *C*. The solid red curves are the individual predictions of theophylline plasma concentration time course; the circles are data points; the gray curves are ten time-course predictions using parameters estimates randomly drawn from their joint posterior distribution.

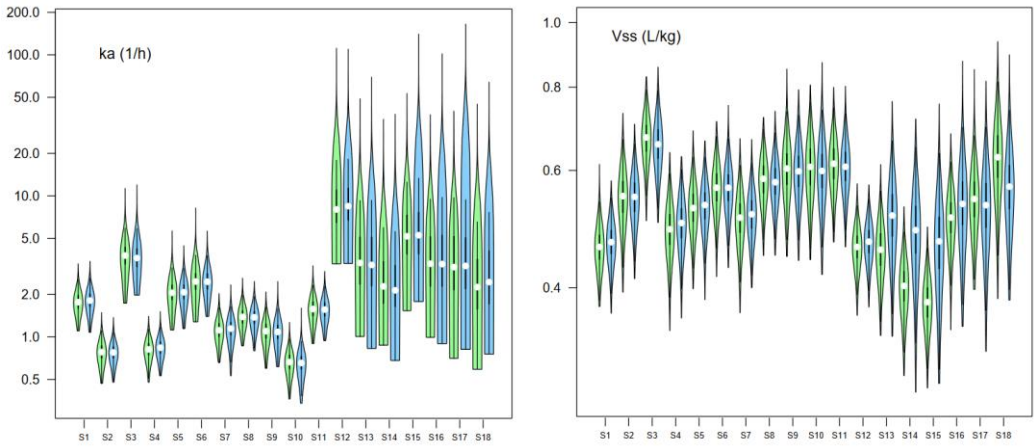

Figure S3: Comparison of violin plots of the individual parameter estimates of K_a (left panel) and V_{ss} (right panel) with model B (in green) and model *C* (in blue; accounting for subjects' age and *BM* uncertainty).

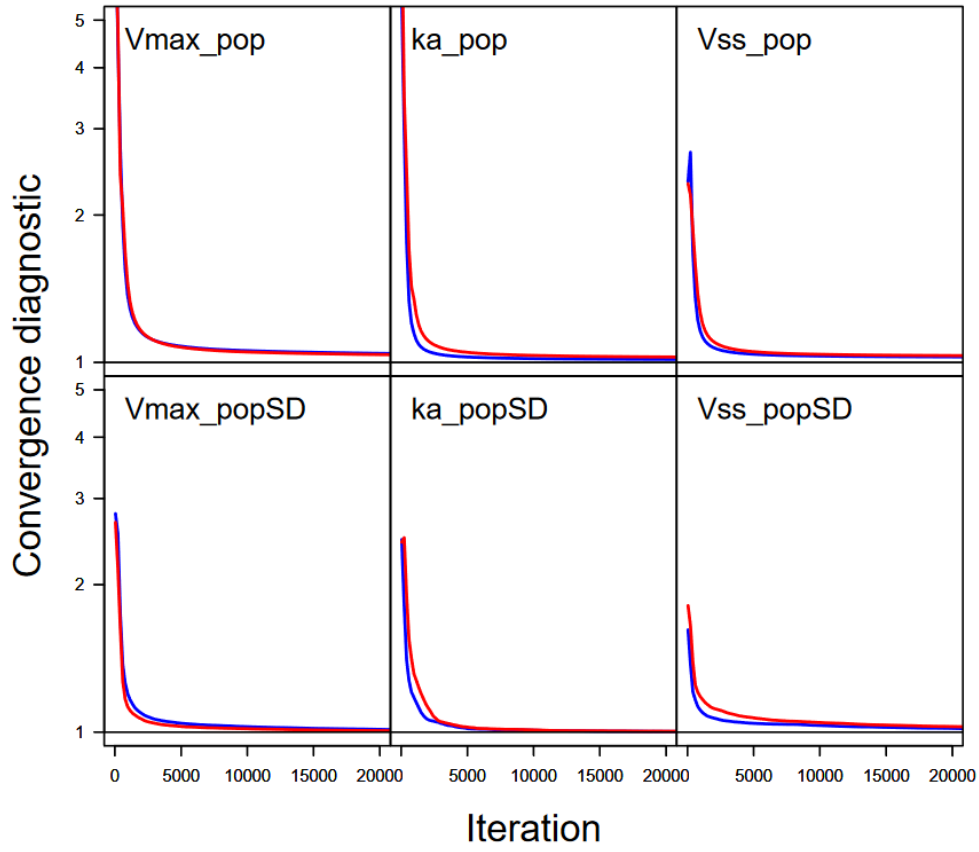

Figure S4: Comparison of convergence (measured by Gelman and Rubin's diagnostic) as a function of iteration number for the six population parameters (V_{max}, K_a, and V_{ss}) with model *C* (in blue) and model *D* (in red).

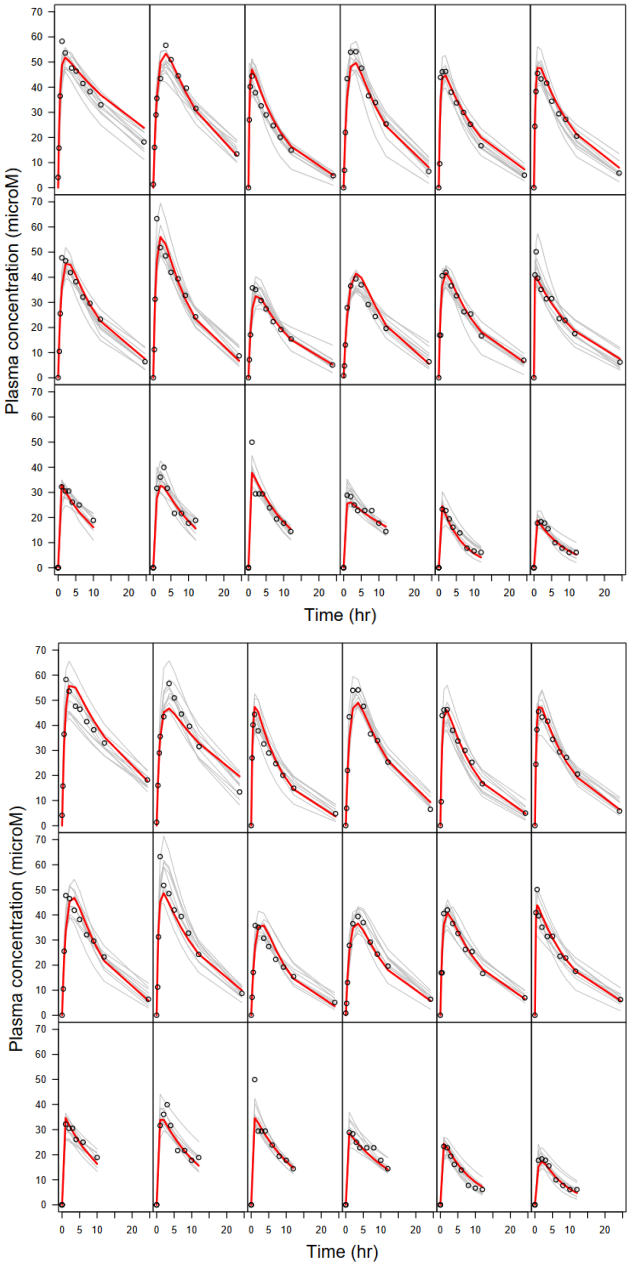

Figure S5: Individual posterior predictions for all subjects, using model *D*. Left panel: one at a time MCMC sampling; right panel: block sampling of newly introduced parameters. The solid red curves are the individual predictions of theophylline plasma concentration time course; the circles are data *points; the gray curves are ten time-course predictions using parameters estimates randomly drawn* from their joint posterior distribution.

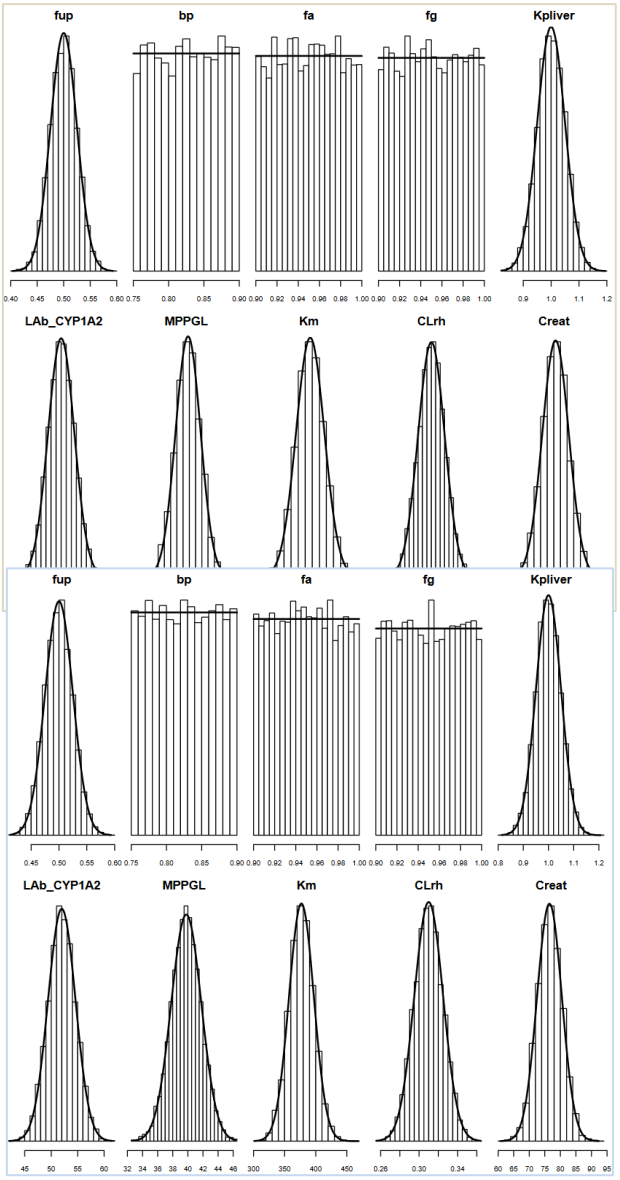

Figure S6: Histograms of the posterior distributions of the top model parameters (the ten pink nodes in Figure 4 of the main text) obtained with statistical model *D*. The corresponding prior distributions densities are given by the thick black lines. Left panel: one parameter at a time MCMC sampling; right panel: block sampling.

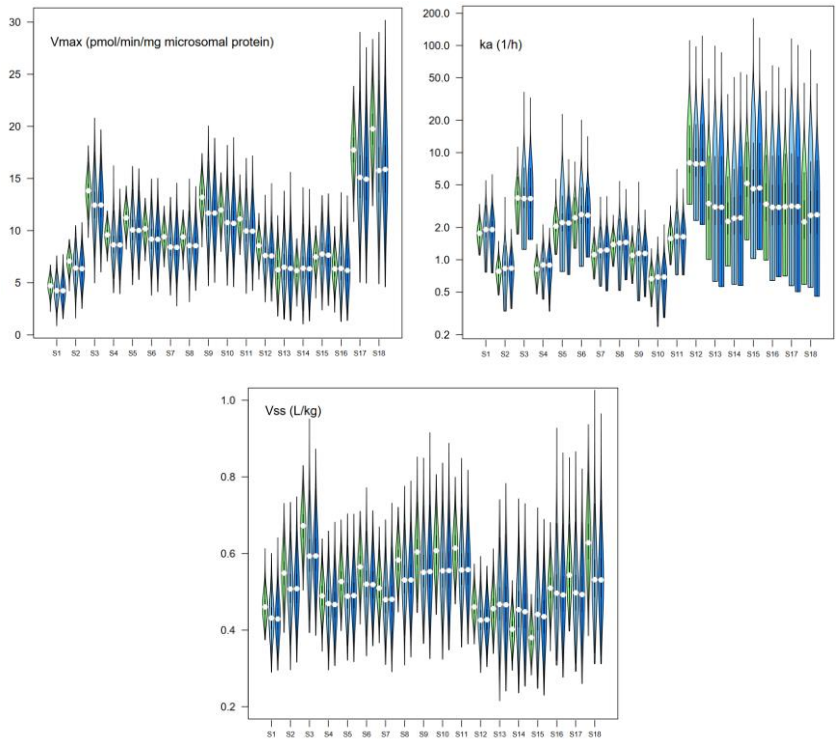

Figure S7: Comparison of violin plots of the individual parameter estimates of V_{max} (left panel) K_a (center) and V_{ss} (right panel) with model *B* (in green) and model *D* (accounting for subjects' age, *BM* and top parameters uncertainty; in light blue for one parameter at a time sampling; in dark blue for block sampling).

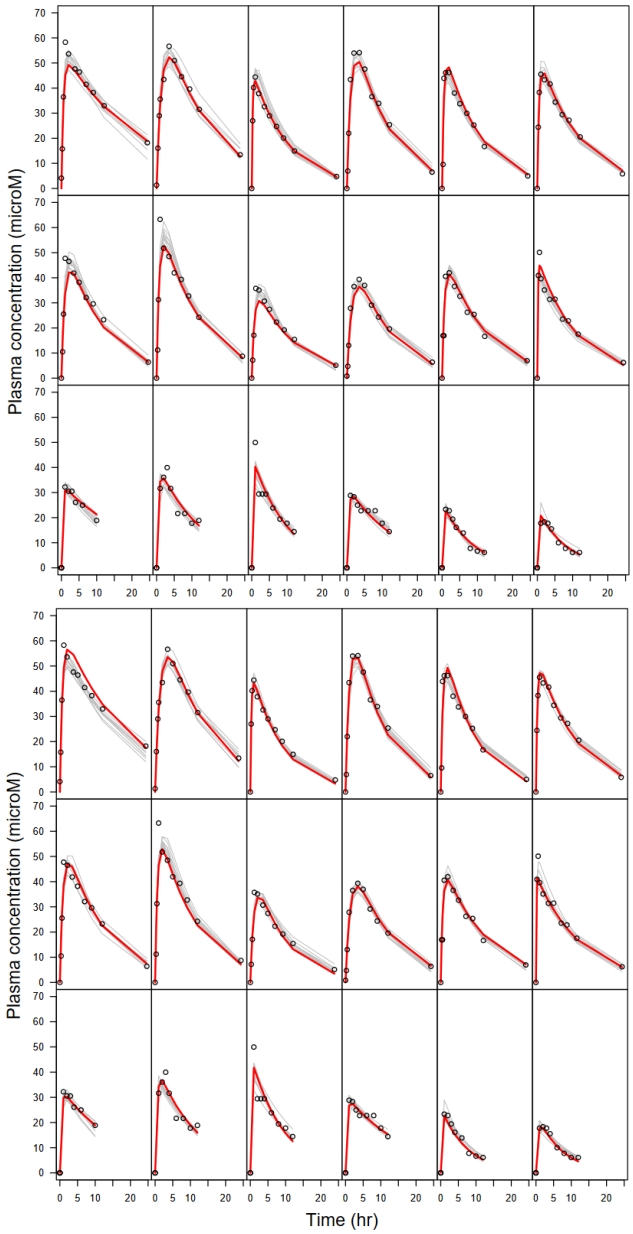

Figure S8: Individual posterior predictions for all subjects, using the fully Bayesian model *E*. Left panel: one at a time MCMC sampling; right panel: block sampling of newly introduced parameters. The solid red curves are the individual predictions of theophylline plasma concentration time course; the circles are data points; the gray curves are ten time-course predictions using parameters estimates randomly drawn from their joint posterior distribution.

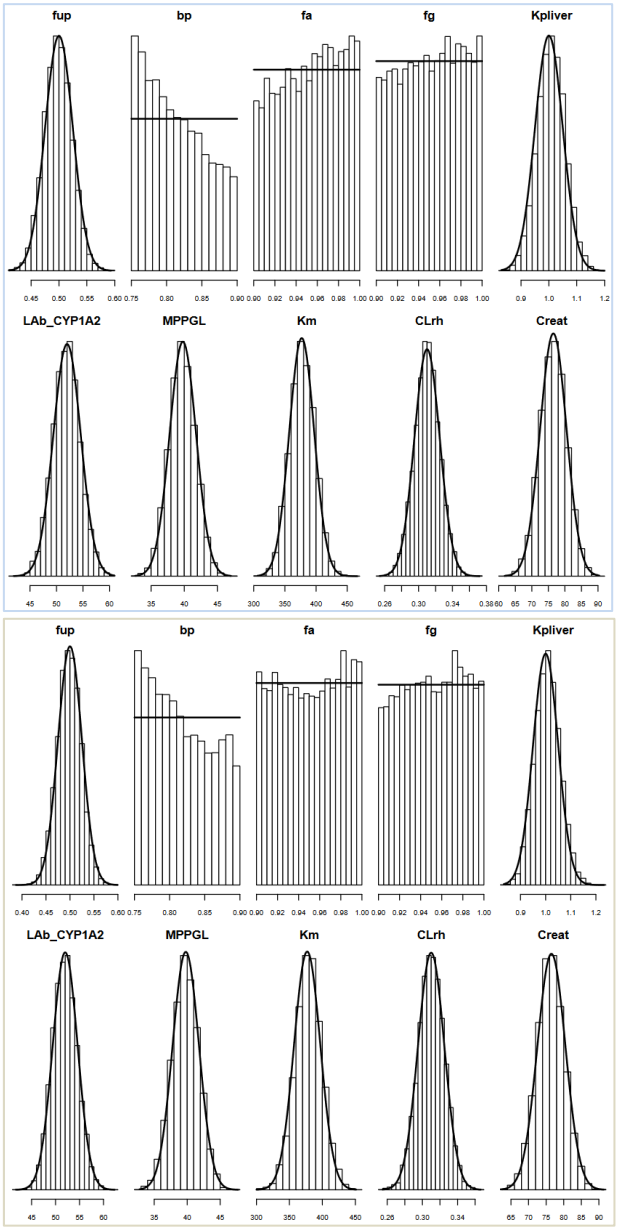

Figure S9: Histograms of the posterior distributions of the top model parameters (the ten pink nodes in Figure 4 of the main text) obtained with statistical model *E* (fully Bayesian). The corresponding prior distributions densities are given by the thick black lines. Left panel: one parameter at a time MCMC sampling; right panel: block sampling.

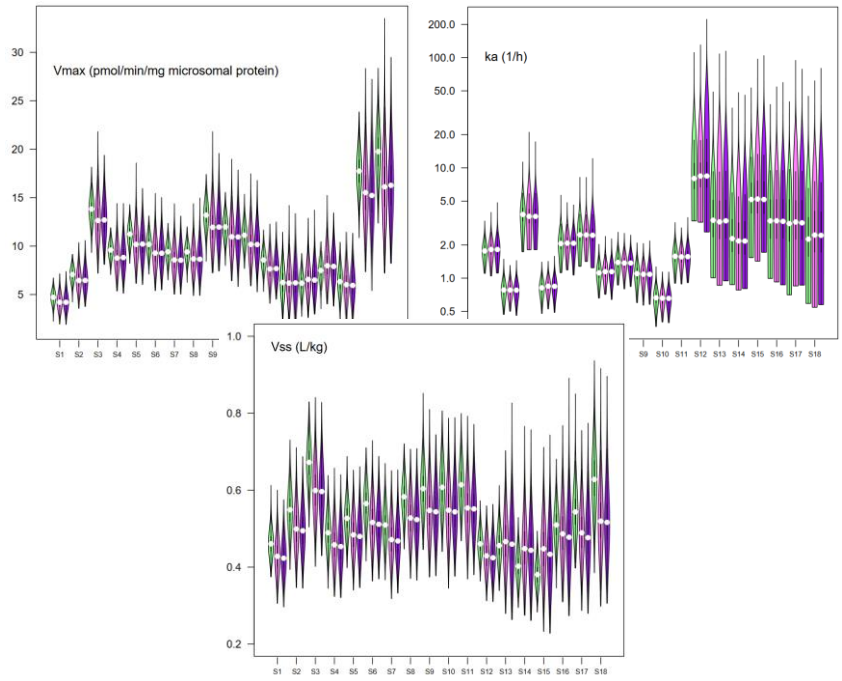

Figure S10: Comparison of violin plots of the individual parameter estimates of V_{max} (left panel) K_a (center) and V_{ss} (right panel) with model *B* (in green) and model *E* (accounting for subjects' age, *BM* and top parameters uncertainty; in dark pink for one parameter at a time sampling; in purple for block sampling).

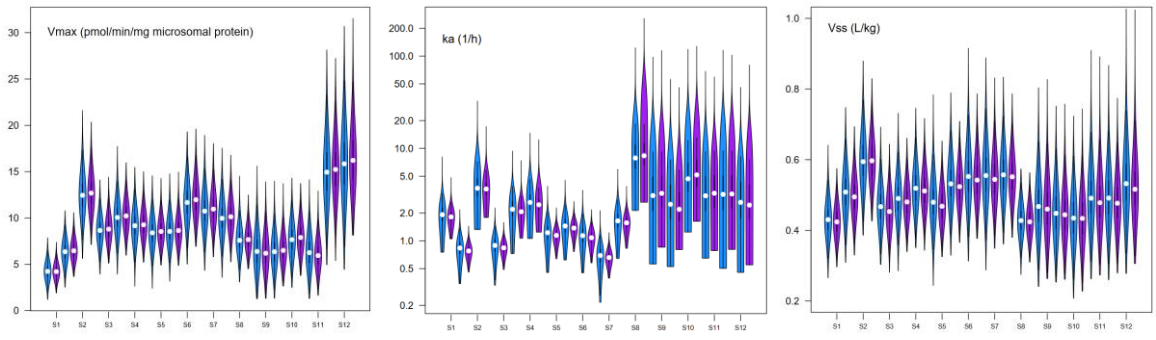

Figure S11: Comparison of violin plots of the individual parameter estimates of V_{max} (left panel) K_a (center) and V_{ss} (right panel) with model *D* (using block stationary sampling; in dark blue) and model *E* (using fully Bayesian block sampling; in purple).

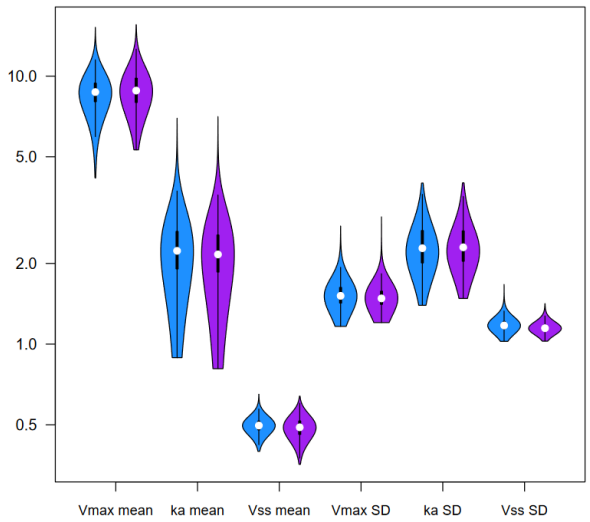

Figure S12: Comparison of model *D* (using block stationary sampling; in dark blue) and model *E* (using fully Bayesian block sampling; in purple) V_{max}, K_a, and V_{ss} population parameter estimates.

2. Posterior distribution summaries

Table S1: Model *B* summaries of the posterior parameter distributions for the parameters of interest. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Vmax[1]	4.71	0.54	3.61	4.36	4.71	5.07	5.72
Vmax[2]	7.05	0.64	5.71	6.65	7.07	7.49	8.26
Vmax[3]	13.80	1.09	11.61	13.11	13.82	14.49	15.93
Vmax[4]	9.60	0.71	8.19	9.13	9.60	10.08	10.94
Vmax[5]	11.24	0.84	9.58	10.67	11.23	11.81	12.88
Vmax[6]	10.17	0.84	8.50	9.62	10.18	10.74	11.84
Vmax[7]	9.42	0.79	7.82	8.90	9.41	9.95	10.93
Vmax[8]	9.40	0.71	7.96	8.93	9.40	9.87	10.79
Vmax[9]	13.22	1.13	10.98	12.46	13.20	13.97	15.43
Vmax[10]	11.97	1.00	10.02	11.29	11.97	12.66	13.93
Vmax[11]	11.13	0.97	9.22	10.47	11.12	11.78	13.07
Vmax[12]	8.56	0.77	7.03	8.07	8.56	9.05	10.11
Vmax[13]	6.24	1.34	3.58	5.34	6.24	7.17	8.80
Vmax[14]	6.15	0.87	4.38	5.58	6.17	6.75	7.81
Vmax[15]	7.44	0.80	5.84	6.94	7.49	7.99	8.93
Vmax[16]	6.32	1.10	4.11	5.60	6.33	7.07	8.50
Vmax[17]	17.73	1.75	14.23	16.57	17.74	18.91	21.16
Vmax[18]	19.75	2.21	15.34	18.25	19.76	21.19	24.09
Vmax_pop	9.63	1.03	7.75	8.93	9.57	10.29	11.75
Vmax_popSD	1.55	0.15	1.34	1.45	1.53	1.63	1.92
Vss[1]	0.46	0.03	0.41	0.44	0.46	0.48	0.52
Vss[2]	0.55	0.04	0.47	0.52	0.55	0.58	0.64
Vss[3]	0.67	0.05	0.58	0.64	0.67	0.70	0.77
Vss[4]	0.49	0.04	0.42	0.46	0.49	0.52	0.57
Vss[5]	0.53	0.04	0.46	0.50	0.53	0.55	0.61
Vss[6]	0.57	0.04	0.49	0.54	0.57	0.59	0.65
Vss[7]	0.51	0.04	0.43	0.48	0.51	0.53	0.59
Vss[8]	0.58	0.04	0.51	0.55	0.58	0.61	0.66
Vss[9]	0.60	0.05	0.51	0.57	0.60	0.64	0.70
Vss[10]	0.61	0.06	0.51	0.57	0.61	0.65	0.73
Vss[11]	0.62	0.05	0.53	0.58	0.61	0.65	0.71
Vss[12]	0.46	0.03	0.41	0.44	0.46	0.48	0.51
Vss[13]	0.46	0.04	0.38	0.43	0.46	0.48	0.53
Vss[14]	0.40	0.03	0.34	0.38	0.40	0.42	0.47
Vss[15]	0.38	0.03	0.33	0.36	0.38	0.40	0.44
Vss[16]	0.51	0.04	0.43	0.48	0.51	0.54	0.59
Vss[17]	0.55	0.05	0.46	0.51	0.54	0.58	0.65
Vss[18]	0.63	0.07	0.51	0.59	0.63	0.68	0.77
Vss_pop	0.53	0.03	0.48	0.51	0.53	0.54	0.58
Vss_popSD	1.21	0.05	1.13	1.17	1.20	1.24	1.33
ka[1]	1.80	0.30	1.32	1.60	1.76	1.98	2.50
ka[2]	0.79	0.13	0.58	0.71	0.78	0.87	1.07
ka[3]	3.90	0.90	2.55	3.29	3.78	4.33	6.02

ka[4]	0.83	0.12	0.62	0.74	0.82	0.90	1.08
ka[5]	2.12	0.42	1.48	1.83	2.06	2.34	3.09
ka[6]	2.55	0.56	1.74	2.17	2.47	2.83	3.82
ka[7]	1.14	0.19	0.83	1.01	1.12	1.25	1.57
ka[8]	1.41	0.22	1.04	1.25	1.39	1.54	1.90
ka[9]	1.12	0.20	0.78	0.98	1.10	1.24	1.56
ka[10]	0.68	0.11	0.49	0.60	0.66	0.75	0.93
ka[11]	1.61	0.29	1.15	1.41	1.58	1.78	2.25
ka[12]	10.06	7.08	4.52	6.46	8.02	11.04	27.95
ka[13]	4.52	3.93	1.43	2.36	3.35	5.12	14.70
ka[14]	3.17	2.72	1.18	1.73	2.28	3.42	11.26
ka[15]	6.50	4.60	2.50	3.86	5.17	7.33	19.45
ka[16]	4.49	3.80	1.41	2.29	3.31	5.16	15.13
ka[17]	4.55	4.39	1.30	2.15	3.13	5.18	16.73
ka[18]	3.22	3.28	0.94	1.57	2.26	3.56	11.02
ka_pop	2.24	0.59	1.42	1.85	2.15	2.51	3.72
ka_popSD	2.37	0.45	1.71	2.04	2.29	2.62	3.48
b0	0.98	0.13	0.75	0.89	0.96	1.06	1.25
b1	0.10	0.01	0.08	0.09	0.10	0.11	0.12

Table S2: Model *C* summaries of the posterior parameter distributions for the parameters of interest. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Age[1]	26.11	2.88	21.27	23.62	26.17	28.62	30.76
Age[2]	26.03	2.89	21.26	23.52	26.05	28.52	30.75
Age[3]	26.00	2.89	21.25	23.50	25.99	28.51	30.75
Age[4]	25.99	2.89	21.24	23.50	25.97	28.51	30.74
Age[5]	25.99	2.88	21.27	23.50	25.98	28.49	30.74
Age[6]	25.99	2.89	21.23	23.49	25.98	28.50	30.76
Age[7]	25.99	2.89	21.24	23.49	25.98	28.48	30.76
Age[8]	26.04	2.89	21.26	23.55	26.03	28.56	30.76
Age[9]	26.04	2.89	21.26	23.54	26.05	28.54	30.76
Age[10]	26.05	2.89	21.25	23.55	26.08	28.58	30.76
Age[11]	25.99	2.88	21.25	23.51	26.00	28.46	30.75
Age[12]	25.98	2.89	21.24	23.49	25.97	28.48	30.74
Age[13]	26.03	2.89	21.25	23.53	26.04	28.54	30.76
Age[14]	26.05	2.88	21.27	23.57	26.06	28.56	30.77
Age[15]	25.98	2.88	21.25	23.51	25.97	28.44	30.74
Age[16]	26.05	2.87	21.26	23.59	26.07	28.53	30.74
Age[17]	25.95	2.88	21.25	23.47	25.94	28.45	30.74
Age[18]	25.95	2.89	21.26	23.45	25.92	28.43	30.75
BM[1]	79.56	0.50	78.58	79.22	79.56	79.89	80.54
BM[2]	58.23	0.50	57.26	57.89	58.24	58.57	59.21
BM[3]	65.07	0.50	64.09	64.74	65.08	65.41	66.06
BM[4]	60.51	0.50	59.53	60.17	60.52	60.85	61.50
BM[5]	72.41	0.50	71.43	72.07	72.41	72.75	73.37
BM[6]	70.53	0.50	69.56	70.20	70.53	70.87	71.51
BM[7]	72.70	0.50	71.72	72.36	72.70	73.04	73.68
BM[8]	54.66	0.50	53.68	54.32	54.66	55.00	55.65
BM[9]	80.02	0.50	79.03	79.68	80.02	80.36	81.00
BM[10]	64.65	0.50	63.68	64.31	64.65	64.99	65.63
BM[11]	70.54	0.50	69.55	70.20	70.54	70.88	71.52
BM[12]	86.36	0.50	85.38	86.02	86.36	86.70	87.34
BM[13]	71.03	7.78	55.68	65.79	70.85	76.09	86.88
BM[14]	65.05	8.37	50.02	59.35	64.65	70.32	82.78
BM[15]	64.92	8.98	49.00	58.64	64.20	70.89	84.02
BM[16]	76.34	8.61	60.09	70.44	76.16	82.24	93.94
BM[17]	84.59	10.10	65.42	77.91	84.16	90.96	105.87
BM[18]	93.89	11.16	74.22	85.72	93.29	101.13	117.16
Vmax[1]	4.73	0.56	3.60	4.36	4.74	5.10	5.81
Vmax[2]	7.21	0.69	5.83	6.76	7.23	7.68	8.53
Vmax[3]	13.99	1.17	11.74	13.20	13.98	14.76	16.31
Vmax[4]	9.73	0.78	8.22	9.22	9.73	10.25	11.27
Vmax[5]	11.29	0.92	9.49	10.67	11.28	11.91	13.10
Vmax[6]	10.28	0.89	8.53	9.69	10.28	10.87	12.03
Vmax[7]	9.46	0.83	7.85	8.91	9.47	10.01	11.10
Vmax[8]	9.59	0.77	8.09	9.07	9.58	10.10	11.12
Vmax[9]	13.21	1.22	10.83	12.39	13.19	14.01	15.65
Vmax[10]	12.13	1.09	10.00	11.40	12.13	12.85	14.29
Vmax[11]	11.24	1.01	9.25	10.56	11.23	11.91	13.22

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Vmax[12]	8.52	0.79	6.96	7.99	8.51	9.05	10.10
Vmax[13]	7.00	1.54	4.07	5.95	6.97	8.00	10.16
Vmax[14]	7.27	1.23	4.94	6.43	7.25	8.07	9.81
Vmax[15]	8.67	1.21	6.40	7.84	8.63	9.45	11.12
Vmax[16]	6.75	1.31	4.25	5.85	6.73	7.62	9.39
Vmax[17]	17.12	2.29	12.85	15.56	17.05	18.59	21.84
Vmax[18]	17.75	2.69	12.73	15.91	17.67	19.51	23.23
Vmax_pop	9.86	1.00	8.02	9.19	9.81	10.46	11.96
Vmax_popSD	1.49	0.14	1.30	1.40	1.47	1.56	1.84
Vss[1]	0.47	0.03	0.41	0.45	0.47	0.49	0.53
Vss[2]	0.55	0.04	0.47	0.52	0.55	0.57	0.63
Vss[3]	0.66	0.05	0.57	0.62	0.66	0.69	0.76
Vss[4]	0.50	0.04	0.43	0.48	0.50	0.53	0.58
Vss[5]	0.53	0.04	0.46	0.51	0.53	0.56	0.60
Vss[6]	0.57	0.04	0.49	0.54	0.56	0.59	0.64
Vss[7]	0.52	0.04	0.45	0.49	0.52	0.54	0.59
Vss[8]	0.58	0.04	0.50	0.55	0.58	0.60	0.66
Vss[9]	0.60	0.05	0.51	0.57	0.60	0.63	0.70
Vss[10]	0.60	0.05	0.51	0.57	0.60	0.64	0.71
Vss[11]	0.61	0.05	0.52	0.58	0.61	0.64	0.70
Vss[12]	0.47	0.03	0.42	0.45	0.47	0.49	0.52
Vss[13]	0.52	0.06	0.41	0.48	0.51	0.55	0.64
Vss[14]	0.49	0.06	0.38	0.45	0.49	0.53	0.61
Vss[15]	0.47	0.06	0.36	0.43	0.47	0.51	0.60
Vss[16]	0.54	0.06	0.43	0.50	0.54	0.58	0.67
Vss[17]	0.54	0.06	0.42	0.49	0.53	0.57	0.67
Vss[18]	0.57	0.07	0.45	0.53	0.57	0.61	0.72
Vss_pop	0.54	0.02	0.49	0.52	0.54	0.55	0.59
Vss_popSD	1.15	0.05	1.08	1.12	1.15	1.18	1.27
ka[1]	1.85	0.31	1.35	1.63	1.81	2.03	2.56
ka[2]	0.79	0.12	0.59	0.71	0.78	0.86	1.05
ka[3]	3.78	0.95	2.46	3.14	3.62	4.23	6.06
ka[4]	0.85	0.12	0.64	0.77	0.84	0.93	1.12
ka[5]	2.13	0.41	1.51	1.85	2.08	2.35	3.07
ka[6]	2.54	0.53	1.75	2.18	2.47	2.82	3.79
ka[7]	1.16	0.19	0.85	1.03	1.15	1.27	1.59
ka[8]	1.39	0.22	1.03	1.24	1.37	1.52	1.89
ka[9]	1.11	0.19	0.79	0.97	1.09	1.22	1.55
ka[10]	0.67	0.11	0.48	0.59	0.66	0.73	0.91
ka[11]	1.59	0.28	1.13	1.39	1.56	1.75	2.23
ka[12]	10.44	7.71	4.76	6.70	8.41	11.34	28.77
ka[13]	4.68	4.74	1.41	2.29	3.24	5.14	17.61
ka[14]	3.13	3.08	1.13	1.66	2.19	3.30	11.38
ka[15]	7.15	7.95	2.51	3.88	5.23	7.65	22.14
ka[16]	4.95	6.17	1.42	2.30	3.28	5.21	19.18
ka[17]	4.82	7.37	1.34	2.22	3.18	5.09	17.10
ka[18]	3.89	4.74	1.06	1.72	2.49	4.17	15.31
ka_pop	2.28	0.61	1.43	1.86	2.17	2.57	3.79
ka_popSD	2.40	0.48	1.71	2.04	2.31	2.67	3.60

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
b0	0.97	0.13	0.75	0.88	0.96	1.06	1.27
b1	0.10	0.01	0.08	0.09	0.10	0.11	0.12

Table S3: Model *D* summaries of the posterior parameter distributions for the parameters of interest. One parameter at a time stationary sampling was used. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Age[1]	26.1	2.89	21.3	23.5	26.1	28.6	30.8
Age[2]	26.1	2.88	21.3	23.6	26.1	28.5	30.8
Age[3]	26.0	2.89	21.3	23.5	26.0	28.6	30.7
Age[4]	26.0	2.88	21.3	23.6	26.0	28.5	30.7
Age[5]	26.0	2.90	21.2	23.5	26.0	28.4	30.8
Age[6]	26.0	2.88	21.3	23.5	25.9	28.4	30.8
Age[7]	26.1	2.89	21.3	23.6	26.1	28.6	30.8
Age[8]	26.0	2.89	21.3	23.5	26.1	28.6	30.8
Age[9]	26.0	2.89	21.3	23.5	26.0	28.6	30.7
Age[10]	26.0	2.91	21.2	23.4	26.0	28.5	30.8
Age[11]	26.0	2.90	21.2	23.5	26.0	28.5	30.8
Age[12]	25.9	2.88	21.2	23.4	25.9	28.4	30.7
Age[13]	26.0	2.89	21.3	23.5	26.1	28.5	30.8
Age[14]	26.1	2.89	21.3	23.6	26.1	28.6	30.8
Age[15]	26.0	2.87	21.2	23.5	26.0	28.5	30.7
Age[16]	26.0	2.87	21.3	23.5	26.0	28.5	30.7
Age[17]	26.0	2.89	21.2	23.5	25.9	28.5	30.7
Age[18]	26.0	2.86	21.3	23.5	25.9	28.4	30.7
BM[1]	79.6	0.50	78.6	79.2	79.6	79.9	80.5
BM[2]	58.2	0.50	57.3	57.9	58.2	58.6	59.2
BM[3]	65.1	0.51	64.1	64.7	65.0	65.4	66.1
BM[4]	60.5	0.50	59.5	60.2	60.5	60.8	61.5
BM[5]	72.4	0.50	71.4	72.1	72.4	72.7	73.4
BM[6]	70.5	0.50	69.5	70.2	70.5	70.9	71.5
BM[7]	72.7	0.50	71.7	72.4	72.7	73.0	73.7
BM[8]	54.6	0.51	53.7	54.3	54.6	55.0	55.6
BM[9]	80.0	0.50	79.0	79.7	80.0	80.4	81.0
BM[10]	64.6	0.51	63.6	64.3	64.6	65.0	65.6
BM[11]	70.5	0.50	69.5	70.2	70.5	70.9	71.5
BM[12]	86.4	0.50	85.4	86.0	86.4	86.7	87.3
BM[13]	71.0	9.48	53.2	64.6	70.5	77.2	90.8
BM[14]	65.2	9.80	47.7	58.4	64.8	71.5	86.0
BM[15]	63.7	8.65	47.7	57.7	63.6	69.2	81.8
BM[16]	73.7	9.43	56.5	67.6	73.5	79.3	94.5
BM[17]	84.0	12.0	61.5	75.9	83.3	91.3	111
BM[18]	91.2	11.0	70.8	84.0	91.0	97.7	115
Vmax[1]	4.24	0.83	2.63	3.70	4.23	4.76	5.92
Vmax[2]	6.40	1.13	4.17	5.66	6.40	7.13	8.65
Vmax[3]	12.5	1.93	8.84	11.2	12.4	13.7	16.5
Vmax[4]	8.69	1.30	6.22	7.84	8.65	9.49	11.4
Vmax[5]	10.1	1.53	7.15	9.10	10.1	11.1	13.1
Vmax[6]	9.19	1.47	6.43	8.19	9.17	10.1	12.2
Vmax[7]	8.45	1.34	5.83	7.58	8.41	9.31	11.2
Vmax[8]	8.57	1.29	6.12	7.70	8.53	9.37	11.2
Vmax[9]	11.7	1.91	8.14	10.4	11.7	12.9	15.7
Vmax[10]	10.8	1.72	7.54	9.66	10.7	11.9	14.3

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Vmax[11]	10.0	1.62	6.89	8.93	9.98	11.1	13.4
Vmax[12]	7.65	1.24	5.28	6.81	7.62	8.44	10.2
Vmax[13]	6.49	1.86	3.06	5.19	6.42	7.69	10.4
Vmax[14]	6.48	1.59	3.49	5.41	6.43	7.51	9.76
Vmax[15]	7.78	1.58	4.75	6.73	7.74	8.81	11.0
Vmax[16]	6.34	1.65	3.26	5.21	6.30	7.40	9.76
Vmax[17]	15.3	3.09	9.60	13.2	15.1	17.3	21.8
Vmax[18]	15.9	3.37	9.51	13.7	15.8	18.1	22.9
Vmax_pop	8.81	1.09	6.83	8.07	8.74	9.48	11.1
Vmax_popSD	1.54	0.17	1.30	1.42	1.51	1.62	1.96
Vss[1]	0.43	0.04	0.35	0.40	0.43	0.46	0.52
Vss[2]	0.51	0.06	0.40	0.47	0.51	0.55	0.63
Vss[3]	0.60	0.07	0.48	0.55	0.59	0.64	0.74
Vss[4]	0.47	0.05	0.37	0.44	0.47	0.50	0.58
Vss[5]	0.49	0.05	0.40	0.46	0.49	0.52	0.60
Vss[6]	0.52	0.05	0.42	0.49	0.52	0.56	0.63
Vss[7]	0.48	0.05	0.38	0.45	0.48	0.51	0.59
Vss[8]	0.53	0.06	0.43	0.50	0.53	0.57	0.65
Vss[9]	0.56	0.06	0.44	0.51	0.55	0.60	0.69
Vss[10]	0.56	0.07	0.43	0.51	0.56	0.60	0.71
Vss[11]	0.56	0.06	0.45	0.52	0.56	0.60	0.69
Vss[12]	0.43	0.04	0.35	0.40	0.43	0.45	0.51
Vss[13]	0.47	0.07	0.35	0.42	0.47	0.51	0.62
Vss[14]	0.45	0.07	0.33	0.41	0.45	0.50	0.60
Vss[15]	0.44	0.06	0.32	0.40	0.44	0.48	0.58
Vss[16]	0.50	0.07	0.37	0.45	0.50	0.55	0.66
Vss[17]	0.50	0.07	0.37	0.45	0.50	0.54	0.67
Vss[18]	0.54	0.08	0.41	0.48	0.53	0.58	0.71
Vss_pop	0.50	0.03	0.44	0.48	0.50	0.52	0.56
Vss_popSD	1.18	0.06	1.09	1.14	1.17	1.21	1.32
ka[1]	1.99	0.54	1.18	1.62	1.91	2.26	3.28
ka[2]	0.86	0.21	0.53	0.71	0.83	0.97	1.35
ka[3]	4.14	1.97	2.12	3.04	3.73	4.70	8.53
ka[4]	0.92	0.21	0.57	0.77	0.89	1.04	1.41
ka[5]	2.36	0.95	1.34	1.85	2.21	2.67	4.10
ka[6]	2.81	1.01	1.54	2.17	2.64	3.20	5.09
ka[7]	1.26	0.33	0.77	1.03	1.22	1.43	2.02
ka[8]	1.50	0.39	0.91	1.23	1.44	1.70	2.39
ka[9]	1.19	0.32	0.70	0.97	1.14	1.36	1.96
ka[10]	0.71	0.17	0.43	0.59	0.69	0.81	1.11
ka[11]	1.72	0.48	1.03	1.39	1.64	1.96	2.86
ka[12]	9.86	7.49	3.82	6.01	7.88	11.03	28.23
ka[13]	4.45	5.01	1.20	2.12	3.10	4.91	15.54
ka[14]	3.54	3.70	1.04	1.71	2.42	3.83	12.96
ka[15]	6.70	9.52	2.06	3.37	4.61	6.97	20.96
ka[16]	4.40	4.45	1.19	2.11	3.09	4.96	15.88
ka[17]	4.81	6.37	1.22	2.18	3.21	5.23	17.31
ka[18]	4.01	4.97	0.98	1.74	2.59	4.35	15.79
ka_pop	2.34	0.64	1.44	1.91	2.23	2.64	3.91

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
ka_popSD	2.37	0.49	1.67	2.01	2.27	2.65	3.58
CLrh	0.31	0.01	0.28	0.30	0.31	0.32	0.34
Creat	76.5	4.0	68.7	73.8	76.5	79.1	84.3
Km	377	20	338	364	377	390	415
Kpliver	1.00	0.05	0.90	0.97	1.00	1.04	1.10
LAb_CYP1A2	52.0	2.60	46.9	50.2	52.0	53.8	57.1
MPPGL	39.7	2.01	35.8	38.4	39.7	41.1	43.6
bp	0.83	0.04	0.75	0.79	0.83	0.86	0.90
fa	0.95	0.03	0.90	0.93	0.95	0.97	1.00
fg	0.95	0.03	0.90	0.93	0.95	0.98	1.00
fup	0.50	0.03	0.45	0.48	0.50	0.52	0.55
b0	1.11	0.26	0.74	0.92	1.06	1.25	1.72
b1	0.14	0.03	0.10	0.12	0.14	0.15	0.20

Table S4: Model *D* summaries of the posterior parameter distributions for the parameters of interest. Block stationary sampling was used. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Age[1]	26.1	2.91	21.3	23.5	26.1	28.6	30.8
Age[2]	26.0	2.85	21.2	23.6	26.1	28.5	30.8
Age[3]	26.0	2.90	21.2	23.5	26.0	28.5	30.8
Age[4]	26.0	2.89	21.3	23.5	26.0	28.5	30.8
Age[5]	26.0	2.88	21.3	23.5	25.9	28.5	30.8
Age[6]	26.0	2.88	21.2	23.5	26.0	28.5	30.7
Age[7]	26.1	2.86	21.3	23.6	26.1	28.5	30.8
Age[8]	26.0	2.89	21.3	23.4	25.9	28.5	30.7
Age[9]	26.0	2.90	21.3	23.5	26.0	28.5	30.7
Age[10]	25.9	2.88	21.2	23.4	25.9	28.4	30.7
Age[11]	26.0	2.87	21.3	23.5	26.0	28.5	30.7
Age[12]	26.0	2.88	21.3	23.5	26.0	28.5	30.8
Age[13]	26.0	2.88	21.3	23.5	26.1	28.5	30.7
Age[14]	26.0	2.90	21.3	23.5	26.1	28.6	30.8
Age[15]	26.0	2.89	21.2	23.5	26.0	28.5	30.8
Age[16]	26.0	2.87	21.3	23.6	26.0	28.5	30.8
Age[17]	26.0	2.88	21.2	23.5	26.0	28.5	30.7
Age[18]	25.9	2.90	21.2	23.4	25.9	28.5	30.7
BM[1]	79.6	0.49	78.6	79.2	79.6	79.9	80.5
BM[2]	58.2	0.50	57.2	57.9	58.2	58.6	59.2
BM[3]	65.1	0.50	64.1	64.7	65.1	65.4	66.0
BM[4]	60.5	0.50	59.5	60.2	60.5	60.8	61.5
BM[5]	72.4	0.51	71.4	72.1	72.4	72.7	73.4
BM[6]	70.5	0.51	69.5	70.2	70.5	70.9	71.5
BM[7]	72.7	0.50	71.7	72.4	72.7	73.0	73.7
BM[8]	54.7	0.51	53.7	54.3	54.7	55.0	55.6
BM[9]	80.0	0.50	79.1	79.7	80.0	80.3	81.0
BM[10]	64.6	0.51	63.7	64.3	64.6	65.0	65.6
BM[11]	70.5	0.50	69.6	70.2	70.5	70.9	71.5
BM[12]	86.4	0.50	85.4	86.0	86.4	86.7	87.3
BM[13]	70.8	9.35	53.4	64.3	70.4	76.7	90.2
BM[14]	65.5	8.97	48.8	59.0	65.1	71.5	84.2
BM[15]	64.9	9.05	48.6	58.6	64.3	70.7	84.0
BM[16]	74.7	9.40	57.8	67.8	74.8	80.9	94.1
BM[17]	85.4	11.6	65.8	77.3	84.0	92.5	112
BM[18]	91.1	11.3	70.1	83.0	91.2	98.8	113
Vmax[1]	4.24	0.84	2.62	3.68	4.24	4.79	5.96
Vmax[2]	6.37	1.10	4.25	5.63	6.36	7.09	8.56
Vmax[3]	12.5	1.89	8.96	11.3	12.5	13.7	16.4
Vmax[4]	8.68	1.31	6.14	7.82	8.66	9.52	11.4
Vmax[5]	10.1	1.52	7.25	9.11	10.1	11.1	13.3
Vmax[6]	9.19	1.48	6.32	8.18	9.17	10.1	12.2
Vmax[7]	8.42	1.35	5.79	7.53	8.40	9.29	11.1
Vmax[8]	8.57	1.29	6.10	7.72	8.55	9.39	11.2
Vmax[9]	11.7	1.88	8.11	10.5	11.7	12.9	15.6
Vmax[10]	10.8	1.72	7.56	9.68	10.7	11.9	14.4

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Vmax[11]	10.0	1.60	6.96	8.97	9.97	11.0	13.3
Vmax[12]	7.64	1.26	5.23	6.82	7.59	8.42	10.3
Vmax[13]	6.45	1.88	2.99	5.16	6.39	7.67	10.4
Vmax[14]	6.41	1.53	3.54	5.35	6.36	7.38	9.59
Vmax[15]	7.70	1.59	4.71	6.62	7.66	8.73	11.0
Vmax[16]	6.29	1.66	3.11	5.17	6.26	7.39	9.72
Vmax[17]	15.1	3.06	9.39	12.9	14.9	17.1	21.4
Vmax[18]	15.9	3.47	9.40	13.6	15.8	18.1	23.1
Vmax_pop	8.76	1.11	6.73	8.03	8.72	9.42	11.1
Vmax_popSD	1.54	0.17	1.30	1.42	1.51	1.63	1.96
Vss[1]	0.43	0.04	0.35	0.40	0.43	0.46	0.52
Vss[2]	0.51	0.06	0.41	0.47	0.51	0.55	0.63
Vss[3]	0.60	0.06	0.48	0.55	0.59	0.64	0.74
Vss[4]	0.47	0.05	0.37	0.43	0.47	0.50	0.57
Vss[5]	0.49	0.05	0.40	0.46	0.49	0.52	0.60
Vss[6]	0.52	0.05	0.42	0.48	0.52	0.56	0.63
Vss[7]	0.48	0.05	0.38	0.45	0.48	0.51	0.59
Vss[8]	0.53	0.06	0.43	0.50	0.53	0.57	0.65
Vss[9]	0.56	0.07	0.44	0.51	0.55	0.60	0.69
Vss[10]	0.56	0.07	0.43	0.51	0.56	0.60	0.71
Vss[11]	0.56	0.06	0.45	0.52	0.56	0.60	0.69
Vss[12]	0.43	0.04	0.35	0.40	0.43	0.45	0.51
Vss[13]	0.47	0.07	0.34	0.42	0.47	0.51	0.62
Vss[14]	0.45	0.07	0.33	0.40	0.45	0.49	0.59
Vss[15]	0.44	0.06	0.31	0.39	0.43	0.48	0.57
Vss[16]	0.50	0.07	0.37	0.45	0.49	0.54	0.65
Vss[17]	0.50	0.07	0.37	0.45	0.49	0.54	0.65
Vss[18]	0.54	0.08	0.41	0.49	0.53	0.59	0.72
Vss_pop	0.50	0.03	0.44	0.48	0.50	0.52	0.56
Vss_popSD	1.18	0.06	1.09	1.14	1.17	1.22	1.33
ka[1]	2.00	0.54	1.20	1.63	1.92	2.27	3.24
ka[2]	0.86	0.20	0.53	0.72	0.84	0.97	1.33
ka[3]	4.14	1.94	2.13	3.03	3.71	4.69	8.45
ka[4]	0.92	0.21	0.58	0.77	0.89	1.03	1.39
ka[5]	2.33	0.72	1.34	1.86	2.19	2.67	4.04
ka[6]	2.79	0.97	1.56	2.17	2.60	3.17	5.13
ka[7]	1.27	0.33	0.76	1.04	1.23	1.45	2.02
ka[8]	1.51	0.39	0.92	1.23	1.45	1.71	2.46
ka[9]	1.19	0.32	0.70	0.97	1.14	1.36	1.95
ka[10]	0.71	0.17	0.44	0.59	0.69	0.81	1.12
ka[11]	1.72	0.47	1.02	1.40	1.64	1.95	2.84
ka[12]	9.79	7.02	3.92	6.06	7.88	11.01	28.25
ka[13]	4.58	5.36	1.21	2.12	3.09	4.95	17.37
ka[14]	3.72	4.06	1.03	1.74	2.48	4.06	14.65
ka[15]	6.33	6.21	2.10	3.43	4.70	6.94	20.79
ka[16]	4.43	4.50	1.18	2.10	3.07	4.97	16.12
ka[17]	4.65	5.41	1.22	2.16	3.18	5.07	17.01
ka[18]	3.87	4.01	0.97	1.75	2.61	4.33	14.63
ka_pop	2.34	0.63	1.43	1.91	2.23	2.63	3.87

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
ka_popSD	2.37	0.49	1.68	2.01	2.28	2.65	3.55
CLrh	0.31	0.02	0.28	0.30	0.31	0.32	0.34
Creat	76.5	4.0	68.8	73.8	76.6	79.3	84.3
Km	377	20	337	364	377	390	416
Kpliver	1.00	0.05	0.90	0.97	1.00	1.03	1.10
LAb_CYP1A2	52.0	2.60	46.9	50.3	52.0	53.7	57.0
MPPGL	39.8	2.00	35.8	38.4	39.8	41.1	43.7
bp	0.82	0.04	0.75	0.79	0.83	0.86	0.90
fa	0.95	0.03	0.90	0.93	0.95	0.97	1.00
fg	0.95	0.03	0.90	0.93	0.95	0.98	1.00
fup	0.50	0.02	0.45	0.48	0.50	0.52	0.55
b0	1.12	0.26	0.74	0.93	1.07	1.26	1.75
b1	0.14	0.03	0.09	0.12	0.14	0.15	0.19

Table S5: Model *E* summaries of the posterior parameter distributions for the parameters of interest. One parameter at a time random walk sampling was used. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Age[1]	26.2	2.90	21.3	23.7	26.3	28.7	30.8
Age[2]	26.0	2.87	21.3	23.6	26.1	28.5	30.7
Age[3]	25.9	2.89	21.3	23.4	25.9	28.4	30.7
Age[4]	26.0	2.88	21.2	23.5	26.0	28.5	30.7
Age[5]	26.1	2.88	21.3	23.6	26.1	28.6	30.7
Age[6]	26.0	2.91	21.2	23.4	26.0	28.5	30.8
Age[7]	26.1	2.86	21.3	23.6	26.1	28.5	30.7
Age[8]	26.0	2.89	21.2	23.5	26.0	28.5	30.8
Age[9]	26.0	2.90	21.2	23.5	26.0	28.5	30.7
Age[10]	26.0	2.87	21.3	23.5	26.0	28.5	30.7
Age[11]	26.0	2.89	21.2	23.5	26.0	28.4	30.7
Age[12]	26.0	2.88	21.2	23.5	26.0	28.5	30.7
Age[13]	26.0	2.88	21.2	23.5	26.1	28.4	30.7
Age[14]	26.0	2.88	21.3	23.5	26.0	28.5	30.8
Age[15]	26.0	2.87	21.3	23.5	26.0	28.5	30.7
Age[16]	26.0	2.90	21.2	23.5	26.1	28.5	30.8
Age[17]	26.0	2.89	21.2	23.4	26.0	28.4	30.8
Age[18]	26.0	2.88	21.2	23.5	26.0	28.5	30.8
BM[1]	79.6	0.50	78.6	79.2	79.6	79.9	80.5
BM[2]	58.2	0.50	57.3	57.9	58.2	58.6	59.2
BM[3]	65.1	0.50	64.1	64.7	65.1	65.4	66.0
BM[4]	60.5	0.50	59.5	60.2	60.5	60.9	61.5
BM[5]	72.4	0.49	71.4	72.1	72.4	72.7	73.4
BM[6]	70.5	0.50	69.5	70.2	70.5	70.9	71.5
BM[7]	72.7	0.50	71.7	72.4	72.7	73.1	73.7
BM[8]	54.7	0.49	53.7	54.3	54.7	55.0	55.6
BM[9]	80.0	0.50	79.0	79.7	80.0	80.3	81.0
BM[10]	64.7	0.50	63.7	64.3	64.6	65.0	65.6
BM[11]	70.5	0.50	69.6	70.2	70.5	70.9	71.5
BM[12]	86.4	0.50	85.4	86.0	86.4	86.7	87.4
BM[13]	71.7	8.65	57.0	65.4	71.1	77.2	90.6
BM[14]	64.5	8.05	49.9	58.8	64.1	69.8	81.6
BM[15]	62.0	8.52	46.8	56.1	61.5	67.7	80.1
BM[16]	76.6	8.58	60.2	70.9	76.2	81.9	94.0
BM[17]	84.4	9.6	67.4	77.7	83.9	90.5	106
BM[18]	93.7	9.8	74.1	87.1	94.0	100	113
Vmax[1]	4.25	0.71	2.95	3.75	4.21	4.71	5.73
Vmax[2]	6.54	0.97	4.81	5.86	6.48	7.14	8.62
Vmax[3]	12.8	1.78	9.63	11.6	12.7	13.9	16.6
Vmax[4]	8.85	1.22	6.66	7.99	8.77	9.61	11.5
Vmax[5]	10.3	1.44	7.79	9.33	10.2	11.2	13.4
Vmax[6]	9.35	1.31	7.00	8.44	9.29	10.2	12.1
Vmax[7]	8.64	1.24	6.46	7.78	8.58	9.42	11.3
Vmax[8]	8.75	1.20	6.62	7.91	8.67	9.51	11.4
Vmax[9]	12.1	1.76	9.02	10.9	12.0	13.2	15.8
Vmax[10]	11.1	1.60	8.21	9.95	11.0	12.1	14.5

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Vmax[11]	10.3	1.49	7.64	9.24	10.2	11.2	13.5
Vmax[12]	7.72	1.14	5.67	6.93	7.67	8.46	10.1
Vmax[13]	6.30	1.58	3.45	5.21	6.19	7.28	9.7
Vmax[14]	6.65	1.35	4.33	5.69	6.57	7.49	9.62
Vmax[15]	8.13	1.46	5.61	7.10	8.01	9.04	11.3
Vmax[16]	6.10	1.38	3.63	5.13	6.02	6.97	9.09
Vmax[17]	15.7	2.70	10.88	13.8	15.5	17.4	21.4
Vmax[18]	16.3	3.08	11.06	14.1	16.1	18.2	23.0
Vmax_pop	9.00	1.37	6.61	8.06	8.89	9.85	11.9
Vmax_popSD	1.50	0.14	1.30	1.41	1.48	1.57	1.86
Vss[1]	0.43	0.04	0.36	0.40	0.43	0.45	0.51
Vss[2]	0.50	0.05	0.41	0.47	0.50	0.53	0.60
Vss[3]	0.60	0.06	0.49	0.56	0.60	0.64	0.72
Vss[4]	0.46	0.05	0.37	0.43	0.46	0.49	0.55
Vss[5]	0.49	0.05	0.40	0.45	0.48	0.52	0.58
Vss[6]	0.52	0.05	0.43	0.48	0.52	0.55	0.62
Vss[7]	0.47	0.05	0.39	0.44	0.47	0.50	0.57
Vss[8]	0.53	0.05	0.44	0.49	0.53	0.56	0.63
Vss[9]	0.55	0.06	0.44	0.51	0.55	0.59	0.67
Vss[10]	0.55	0.06	0.44	0.51	0.55	0.59	0.68
Vss[11]	0.56	0.05	0.45	0.52	0.55	0.59	0.67
Vss[12]	0.43	0.04	0.36	0.40	0.43	0.45	0.50
Vss[13]	0.47	0.06	0.35	0.43	0.47	0.51	0.60
Vss[14]	0.45	0.06	0.34	0.41	0.45	0.49	0.58
Vss[15]	0.45	0.06	0.33	0.40	0.45	0.49	0.58
Vss[16]	0.49	0.06	0.38	0.45	0.49	0.53	0.63
Vss[17]	0.49	0.06	0.38	0.45	0.49	0.53	0.62
Vss[18]	0.52	0.07	0.40	0.48	0.52	0.57	0.68
Vss_pop	0.49	0.04	0.42	0.47	0.49	0.52	0.57
Vss_popSD	1.15	0.05	1.07	1.12	1.14	1.17	1.26
ka[1]	1.85	0.30	1.36	1.64	1.82	2.02	2.53
ka[2]	0.79	0.12	0.60	0.71	0.78	0.86	1.06
ka[3]	3.79	0.99	2.45	3.15	3.63	4.23	6.02
ka[4]	0.86	0.12	0.65	0.77	0.85	0.93	1.12
ka[5]	2.13	0.41	1.51	1.85	2.07	2.34	3.09
ka[6]	2.55	0.52	1.75	2.19	2.47	2.82	3.79
ka[7]	1.16	0.19	0.86	1.03	1.14	1.27	1.60
ka[8]	1.40	0.22	1.04	1.25	1.38	1.53	1.91
ka[9]	1.11	0.20	0.79	0.97	1.09	1.22	1.56
ka[10]	0.67	0.11	0.48	0.59	0.66	0.73	0.90
ka[11]	1.59	0.28	1.14	1.39	1.56	1.75	2.21
ka[12]	10.3	7.01	4.76	6.74	8.48	11.3	28.3
ka[13]	4.80	6.10	1.39	2.29	3.26	5.10	17.7
ka[14]	3.14	3.34	1.12	1.65	2.17	3.19	11.9
ka[15]	6.83	5.71	2.47	3.90	5.25	7.65	21.5
ka[16]	4.65	4.42	1.40	2.30	3.31	5.17	16.9
ka[17]	4.73	5.29	1.37	2.24	3.22	5.14	18.1
ka[18]	3.81	4.44	1.04	1.72	2.48	4.09	15.0
ka_pop	2.28	0.60	1.44	1.87	2.17	2.56	3.76

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
ka_popSD	2.39	0.46	1.72	2.04	2.31	2.65	3.51
CLrh	0.31	0.01	0.28	0.30	0.31	0.32	0.34
Creat	76.5	4.0	68.6	73.8	76.5	79.3	84.4
Km	378	20	339	364	378	391	417
Kpliver	1.01	0.05	0.91	0.98	1.01	1.04	1.11
LAb_CYP1A2	51.9	2.58	46.8	50.1	51.9	53.5	57.0
MPPGL	39.7	2.01	35.7	38.3	39.7	41.0	43.6
bp	0.81	0.04	0.75	0.78	0.81	0.85	0.89
fa	0.95	0.03	0.90	0.93	0.95	0.98	1.00
fg	0.95	0.03	0.90	0.93	0.95	0.98	1.00
fup	0.50	0.02	0.45	0.48	0.50	0.52	0.55
b0	0.98	0.13	0.75	0.88	0.96	1.06	1.27
b1	0.10	0.01	0.08	0.09	0.10	0.11	0.12

Table S6: Model *E* summaries of the posterior parameter distributions for the parameters of interest. Block random sampling was used. The population parameters are geometric means and SDs.

Parameter	Mean	SD	2.5%tile	25%tile	50%tile	75%tile	97.5%tile
Age[1]	26.1	2.91	21.3	23.6	26.2	28.6	30.8
Age[2]	26.0	2.90	21.3	23.5	26.1	28.6	30.7
Age[3]	26.0	2.89	21.3	23.5	26.1	28.5	30.7
Age[4]	26.0	2.88	21.3	23.5	26.0	28.6	30.7
Age[5]	26.0	2.87	21.3	23.5	26.1	28.5	30.7
Age[6]	26.0	2.87	21.2	23.5	26.0	28.5	30.7
Age[7]	26.0	2.89	21.3	23.5	25.9	28.5	30.7
Age[8]	26.0	2.88	21.2	23.5	26.0	28.5	30.8
Age[9]	26.0	2.89	21.2	23.5	26.0	28.5	30.8
Age[10]	26.0	2.88	21.3	23.5	25.9	28.5	30.7
Age[11]	26.0	2.87	21.2	23.5	26.0	28.4	30.8
Age[12]	26.0	2.90	21.3	23.5	26.1	28.6	30.8
Age[13]	26.0	2.88	21.2	23.5	26.0	28.5	30.7
Age[14]	25.9	2.88	21.2	23.4	25.9	28.4	30.7
Age[15]	26.0	2.89	21.2	23.5	26.0	28.5	30.8
Age[16]	26.0	2.90	21.2	23.5	26.0	28.6	30.8
Age[17]	26.0	2.87	21.3	23.5	26.0	28.5	30.7
Age[18]	26.0	2.86	21.3	23.6	26.0	28.5	30.7
BM[1]	79.6	0.49	78.6	79.2	79.6	79.9	80.5
BM[2]	58.2	0.50	57.2	57.9	58.2	58.6	59.2
BM[3]	65.1	0.50	64.1	64.7	65.1	65.4	66.0
BM[4]	60.5	0.50	59.5	60.2	60.5	60.8	61.5
BM[5]	72.4	0.49	71.4	72.1	72.4	72.7	73.4
BM[6]	70.5	0.50	69.5	70.2	70.5	70.9	71.5
BM[7]	72.7	0.49	71.7	72.4	72.7	73.0	73.7
BM[8]	54.7	0.49	53.7	54.3	54.7	55.0	55.6
BM[9]	80.0	0.50	79.1	79.7	80.0	80.3	81.0
BM[10]	64.7	0.50	63.7	64.3	64.7	65.0	65.6
BM[11]	70.5	0.50	69.6	70.2	70.5	70.9	71.5
BM[12]	86.4	0.50	85.4	86.0	86.4	86.7	87.4
BM[13]	72.0	9.30	55.3	65.7	71.1	77.9	91.3
BM[14]	64.9	8.36	49.3	59.2	64.7	70.0	83.3
BM[15]	63.8	9.32	48.1	57.6	63.0	68.8	85.4
BM[16]	77.8	9.17	60.4	71.4	77.4	84.0	96.0
BM[17]	86.6	9.6	67.7	80.1	86.2	92.9	106
BM[18]	93.3	11.4	71.9	85.6	92.4	100	119
Vmax[1]	4.25	0.72	2.94	3.76	4.21	4.71	5.75
Vmax[2]	6.55	1.00	4.78	5.84	6.47	7.18	8.66
Vmax[3]	12.8	1.80	9.58	11.5	12.7	13.9	16.5
Vmax[4]	8.87	1.23	6.69	8.01	8.79	9.66	11.4
Vmax[5]	10.3	1.43	7.77	9.32	10.2	11.2	13.4
Vmax[6]	9.36	1.32	6.98	8.42	9.27	10.2	12.1
Vmax[7]	8.62	1.24	6.40	7.75	8.54	9.42	11.3
Vmax[8]	8.73	1.23	6.54	7.85	8.65	9.52	11.4
Vmax[9]	12.1	1.77	8.97	10.8	12.0	13.2	15.8
Vmax[10]	11.1	1.61	8.23	9.97	11.0	12.1	14.6

Vmax[11]	10.2	1.49	7.62	9.17	10.1	11.2	13.4
Vmax[12]	7.73	1.13	5.71	6.94	7.67	8.46	10.1
Vmax[13]	6.31	1.66	3.44	5.14	6.19	7.35	9.8
Vmax[14]	6.59	1.34	4.18	5.66	6.50	7.42	9.47
Vmax[15]	7.99	1.48	5.38	6.95	7.89	8.92	11.3
Vmax[16]	6.03	1.38	3.59	5.05	5.95	6.92	8.98
Vmax[17]	15.4	2.71	10.6	13.5	15.2	17.1	21.2
Vmax[18]	16.4	3.18	10.9	14.2	16.2	18.4	23.4
Vmax_pop	8.96	1.37	6.63	7.97	8.83	9.83	11.9
Vmax_popSD	1.51	0.14	1.30	1.41	1.48	1.58	1.84
Vss[1]	0.42	0.04	0.35	0.40	0.42	0.45	0.50
Vss[2]	0.50	0.05	0.40	0.46	0.49	0.53	0.60
Vss[3]	0.60	0.06	0.49	0.56	0.60	0.64	0.72
Vss[4]	0.45	0.05	0.37	0.42	0.45	0.49	0.55
Vss[5]	0.48	0.05	0.40	0.45	0.48	0.51	0.58
Vss[6]	0.51	0.05	0.42	0.48	0.51	0.55	0.61
Vss[7]	0.47	0.05	0.38	0.44	0.47	0.50	0.56
Vss[8]	0.53	0.05	0.43	0.49	0.52	0.56	0.63
Vss[9]	0.54	0.06	0.44	0.50	0.54	0.58	0.67
Vss[10]	0.55	0.06	0.44	0.51	0.54	0.59	0.67
Vss[11]	0.55	0.06	0.45	0.52	0.55	0.59	0.67
Vss[12]	0.43	0.04	0.35	0.40	0.42	0.45	0.50
Vss[13]	0.46	0.07	0.34	0.42	0.46	0.51	0.61
Vss[14]	0.45	0.06	0.33	0.40	0.44	0.49	0.58
Vss[15]	0.43	0.07	0.30	0.39	0.43	0.48	0.57
Vss[16]	0.48	0.07	0.36	0.44	0.48	0.52	0.63
Vss[17]	0.48	0.06	0.37	0.44	0.48	0.52	0.62
Vss[18]	0.52	0.07	0.39	0.47	0.52	0.57	0.68
Vss_pop	0.49	0.04	0.41	0.46	0.49	0.52	0.57
Vss_popSD	1.15	0.05	1.08	1.12	1.15	1.18	1.27
ka[1]	1.85	0.31	1.35	1.63	1.81	2.02	2.55
ka[2]	0.79	0.12	0.59	0.71	0.78	0.85	1.05
ka[3]	3.81	1.01	2.47	3.17	3.64	4.24	6.16
ka[4]	0.85	0.12	0.64	0.77	0.84	0.93	1.13
ka[5]	2.12	0.41	1.50	1.84	2.07	2.33	3.05
ka[6]	2.54	0.56	1.74	2.17	2.46	2.81	3.79
ka[7]	1.16	0.19	0.85	1.03	1.14	1.27	1.59
ka[8]	1.40	0.22	1.04	1.24	1.37	1.53	1.90
ka[9]	1.10	0.19	0.78	0.97	1.08	1.22	1.54
ka[10]	0.67	0.11	0.49	0.59	0.66	0.73	0.90
ka[11]	1.60	0.29	1.14	1.40	1.57	1.76	2.27
ka[12]	10.6	10.56	4.72	6.63	8.35	11.3	29.4
ka[13]	4.61	4.81	1.40	2.30	3.26	5.07	15.9
ka[14]	3.27	3.54	1.13	1.66	2.19	3.32	13.0
ka[15]	6.76	5.95	2.51	3.83	5.18	7.52	21.1
ka[16]	4.61	4.42	1.40	2.29	3.27	5.13	16.6
ka[17]	4.88	6.17	1.37	2.26	3.22	5.12	18.7
ka[18]	3.70	4.13	1.04	1.72	2.44	4.06	13.8
ka_pop	2.27	0.60	1.42	1.86	2.16	2.56	3.75
ka_popSD	2.39	0.48	1.71	2.04	2.30	2.65	3.58

CLrh	0.31	0.01	0.28	0.30	0.31	0.32	0.34
Creat	76.3	4.00	68.6	73.7	76.2	78.9	84.1
Km	379	20.0	339	366	379	393	419
Kpliver	1.01	0.05	0.91	0.97	1.01	1.04	1.11
LAb_CYP1A2	51.9	2.57	47.0	50.2	51.9	53.7	57.0
MPPGL	39.7	2.04	35.7	38.3	39.7	41.1	43.6
bp	0.82	0.04	0.75	0.78	0.81	0.85	0.89
fa	0.95	0.03	0.90	0.92	0.95	0.98	1.00
fg	0.95	0.03	0.90	0.93	0.95	0.98	1.00
fup	0.50	0.02	0.45	0.48	0.50	0.51	0.55
b0	0.97	0.13	0.75	0.88	0.96	1.06	1.26
b1	0.10	0.01	0.08	0.09	0.10	0.11	0.12

3. Model Codes

3.1. Code for minimal PBPK ODEs

```
/* ------------------------------------------------------------------------
   C-coded minimal PBPK model structural equations - file minPBPK.c
*/
#include <R.h>

#define Nparms 15
static double parms[Nparms];

#define dose         parms[0]
#define ka           parms[1]
#define fa           parms[2]
#define fg           parms[3]
#define fup          parms[4]
#define bp           parms[5]
#define Kpliver      parms[6]
#define Vh           parms[7]
#define Vsys         parms[8]
#define Qpv          parms[9]
#define Qh           parms[10]
#define Qha          parms[11]
#define CLr          parms[12]
#define Vmax_scaled  parms[13]
#define Km           parms[14]

/* initializer */
void minPBPK(void (* odeparms)(int *, double *))
{ int N = Nparms;
  odeparms(&N, parms); }

/* derivatives */
void derivs(int *neq, double *t, double *y, double *ydot, double *yout, int*ip)
{
  /* State variables
     Csys <- y[0] # systemic blood concentration, microM
     Cliv <- y[1] # liver tissue   ...                                    */
  double Qgut = dose * exp(-ka * (*t));            /* micromols          */
  double Cpv = y[0] + (ka * fa * fg * Qgut) / Qpv; /* portal vein conc., microM */
  double Cliv_VP = y[1] / Kpliver; /* plasma conc. in vein leaving liver   */
  double Cliv_VB = Cliv_VP * bp;   /* blood  conc. in vein leaving liver   */

  double CluintH = Vmax_scaled / (Km + Cliv_VP * fup);

  /* ODEs
  /* systemic blood concentration, microM */
  ydot[0] = (Qh * Cliv_VB - Qh * y[0] - CLr * y[0] / bp) / Vsys;
  /* liver tissue concentration, microM */
  ydot[1] = (Qpv * Cpv  - Qh * Cliv_VB + Qha * y[0] -
             CluintH * fup * Cliv_VB / bp) / Vh;
}

/* End */
```

3.2. Code for prior Monte Carlo simulations (model A)

```
## ------------------------------------------------------------------------
## Minimal PBPK Model for theophylline, parameter dependencies only, no ODEs
## V7b
library(nimble)

## ------------------------------------------------------------------------
## Core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## default fixed (top parent) parameters
  dose       <- 1776    # micromols
  fup        <- 0.5     # fraction unbound in plasma
  bp         <- 0.815   # blood to plasma ratio
  fg         <- 1       # fraction escaped from gut metabolism
  Kpliver    <- 1       # liver tissue:plasma part coef.
  density    <- 1080    # liver density (g/L)
  Km         <- 377     # Michaelis-Menten constant (microM)
  CLrh       <- 0.31    # renal clearance in average healthy subject (L/h)
  fa         <- 1       # fraction absorbed
  LAb_CYP1A2 <- 52      # pmol/mg protein
  MPPGL      <- 39.79   # mg microsomal proteins / g liver
  Creat      <- 76.5    # serum creatinine, male < 61 year-old (micromol/L)
  ##
  ## sampled top parent parameters, dnorm uses inverse variance...
  Age  ~ dunif(21, 31)         # from Trembath (years)
  Vss  ~ dnorm(0.5, sd=0.05)   # L per kg
  ka   ~ dnorm(2,   sd=0.2)    # first order absorption rate (1/h)
  Vmax ~ dnorm(10,  sd=1)      # pmol/min/mg microsomal protein
  ##
  ## dependent parameters, constant or sampled
  tmp1  <- 175.32 + 0.113 * Age - 0.0025 * Age * Age
  Ht    ~ dnorm(tmp1, sd=5)                          # body height (cm)
  tmp2  <- exp(2.643 + 0.0099 * Ht)
  BM    ~ dnorm(tmp2, sd=0.15 * tmp2)                # body mass (kg)
  BSA   <- 0.00718 * pow(BM, 0.425) * pow(Ht, 0.725) # body surface area (m^2)
  tmp3  <- 0.722 * pow(BSA, 1.176)
  Vh    ~  dnorm(tmp3, sd=0.025)    # liver tissue volume (L)
  LW    <- Vh * density             # liver weight (g)
  Vsys  <- Vss * BM - Vh * Kpliver / bp
  GFR   <- (140 - Age) * BM / (0.814 * Creat) * 1.73 / BSA # (ml/min)
  RF    <- GFR / 130.0              # renal function (unitless);
  CLr   <- RF * CLrh                # plasma renal clearance (L/h)
  CO    <- BSA * 60 * (3 - 0.01 * (Age - 20))        # cardiac output (L/h)
  Qha   <- 0.196 * CO               # hep art. blood flow (L/h)
  Qpv   <- 0.065 * CO               # portal vein blood flow (L/h)
  Qh    <- Qha + Qpv                # hepatic blood flow (L/h)
  Vmax_scaled <- Vmax * LAb_CYP1A2 * MPPGL * LW * 6e-5
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model
Rmodel = nimbleModel(myNimbleCode, constants= list())
Node.names = Rmodel$getNodeNames(includeData=T)
N.nodes = length(Node.names)

## ------------------------------------------------------------------------
## Monte Carlo simulations
N = 10000
prior.sample = matrix(NA, nrow=N, ncol=N.nodes)
for (i in 1:N) {
  Rmodel$simulate(nodes=Node.names)
  prior.sample[i,] = values(Rmodel, Node.names)
}
colnames(prior.sample) = Node.names

## End.
```

3.3. Code for MCMC simulations of model B

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v7d: Population model replicating Wedagedera et al. paper.

library(nimble)

## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]
## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
  return(result[,2] / parms[6])
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Core Nimble (BUGS)covariate model
myNimbleCode <- nimbleCode({
  ## population means
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ## default fixed (top parent) parameters
  Age        <- 27.637  # Trembath group, assumed (years)
  fup        <- 0.5     # fraction unbound in plasma
  bp         <- 0.815   # blood to plasma ratio
```

```
  fa         <- 1       # absorption fraction
  fg         <- 1       # fraction escaped from gut metabolism
  Kpliver    <- 1       # liver tissue:plasma part coef.
  density    <- 1080    # liver density (g/L)
  LAb_CYP1A2 <- 52      # pmol/mg protein
  MPPGL      <- 39.79   # mg microsomal proteins / g liver
  Km         <- 377     # Michaelis-Menten constant (microM)
  CLrh       <- 0.31    # renal clearance in average healthy subject (L/h)
  Creat      <- 76.5    # serum creatinine, male < 61 year-old (micromol/L)
  ##
  ## dependent parameters, constant but computed from others
  Ht    <- 175.32 + 0.113 * Age - 0.0025 * Age * Age # body height (cm)
  ##
  ## measurement error model
  b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
  b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
  ## for each subject
  for (i in 1:nsubjects) {
    Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
    ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
    Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
    BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht, 0.725) # (m^2)
    CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age - 20))   # cardiac output (L/h)
    Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
    Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
    Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
    Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
    LW[i]    <- Vh[i] * density             # liver weight (g)
    Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
    GFR[i]   <- (140 - Age)*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # (ml/min)
    RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
    CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
    Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, microM
    Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates], times[i, 1:ntimes[i]],
                                       c(dose[i], ka[i], fa,  fg, fup,
                                         bp, Kpliver, Vh[i], Vsys[i],
                                         Qpv[i], Qh[i], Qha[i],
                                         CLr[i], Vmax_scaled[i], Km))
   for (j in 1:ntimes[i]) {
     ## measurement error SD from PLASMA concentration, microM
     sigma[i,j] <- b0 + b1 * Conc[i,j]
     ## data likelihood
     C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
   }
  }
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model for predictions

## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj))))
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
constants <- list(nsubjects = nsubjects,
```

```
                  BM = misc$BM,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))
data <- list(C_plasma_obs = C_plasma_obs)
inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits) # interpreted

Node.names = Rmodel$getNodeNames(includeData=T)

conf <- configureMCMC(Rmodel, thin=5, useConjugacy=F,
                      monitors=c("Vmax_pop", "ka_pop", "Vss_pop",
                                 "Vmax_popSD", "ka_popSD", "Vss_popSD",
                                 "Vmax", "ka", "Vss", "b0", "b1",
                                 "Vh", "Vsys", "Qpv", "Qh", "Qha", "CLr",
                                 "Vmax_scaled"))
Rmcmc <- buildMCMC(conf)

## compile
Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

Mysamples = runMCMC(Cmcmc, niter=10000, nburnin=5000, nchains=4,
samplesAsCodaMCMC=T)

## End.
```

3.4. Code for MCMC simulations of model C

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v8a: Model uncertainty in age and body mass rigorously.

library(nimble)

## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]
## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
```

```
  return(result[,2] / parms[6]) # watch for correct index of bp
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall.
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## population mean
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ##
  ## default fixed (top parent) parameters
  fup        <- 0.5     # fraction unbound in plasma
  bp         <- 0.815   # blood to plasma ratio
  fa         <- 1       # absorption fraction
  fg         <- 1       # fraction escaped from gut metabolism
  Kpliver    <- 1       # liver tissue:plasma part coef.
  density    <- 1080    # liver density (g/L)
  LAb_CYP1A2 <- 52      # pmol/mg protein
  MPPGL      <- 39.79   # mg microsomal proteins / g liver
  Km         <- 377     # Michaelis-Menten constant (microM)
  CLrh       <- 0.31    # renal clearance in average healthy subject (L/h)
  Creat      <- 76.5    # serum creatinine, male < 61 year-old (micromol/L)
  ##
  ## measurement error model
  b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
  b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
  ## for each subject
  for (i in 1:nsubjects) {
    Age[i]  ~ dunif(21, 31)  # Trembath group age range (years)
    Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
    ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
    Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
    ##
    ## dependent parameters, constant but computed from others
    tmp1[i]  <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    Ht[i]    ~ dnorm(tmp1[i], sd = 0.039 * tmp1[i])  # body height (cm)
    tmp2[i]  <- exp(2.643 + 0.0099 * Ht[i])
    BM[i]    ~ dnorm(tmp2[i], sd = 0.150 * tmp2[i])  # body mass (kg)
    ##
    BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht[i], 0.725) # (m^2)
    CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age[i] - 20)) # cardiac output (L/h)
    Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
    Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
    Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
    Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
    LW[i]    <- Vh[i] * density             # liver weight (g)
    Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
    GFR[i]   <- (140 - Age[i])*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # ml/min
    RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
    CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
    Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
```

```
    ##
    ## likelihood for BM measurement
    BM_obs[i] ~ dnorm(BM[i], sd=0.5)    # body mass (kg)
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, microM
    Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates],times[i, 1:ntimes[i]],
                                       c(dose[i], ka[i], fa,  fg, fup,
                                         bp, Kpliver, Vh[i], Vsys[i],
                                         Qpv[i], Qh[i], Qha[i],
                                         CLr[i], Vmax_scaled[i], Km))
    for (j in 1:ntimes[i]) {
      ## measurement error SD from PLASMA concentration, microM
      sigma[i,j] <- b0 + b1 * Conc[i,j]
      ## data likelihood
      C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
    }
  }
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model for predictions

## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj))))
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
BM_obs = misc$BM
BM_obs[13:18] = NA

constants <- list(nsubjects = nsubjects,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))
data <- list(BM_obs = BM_obs,
             C_plasma_obs = C_plasma_obs)
inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits)
Node.names = Rmodel$getNodeNames(includeData=T)

conf <- configureMCMC(Rmodel, thin=5, useConjugacy=F,
                      monitors=c("Age", "BM", "Vmax_pop", "ka_pop", "Vss_pop",
                                 "Vmax_popSD", "ka_popSD", "Vss_popSD",
                                 "Vmax", "ka", "Vss", "b0", "b1",
                                 "Vh", "Vsys", "Qpv", "Qh", "Qha", "CLr",
                                 "Vmax_scaled"))
Rmcmc <- buildMCMC(conf)
Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

mysamples = runMCMC(Cmcmc, niter=55000, nburnin=5000, nchains=4,
samplesAsCodaMCMC=T)

## End.
```

3.5. Code for MCMC simulations of model D

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v9b: Use a stationary sampler for most top parent parameters
##      previously fixed at default values.

library(nimble)

## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]
## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
  return(result[,2] / parms[6]) # watch for correct index of bp
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall.
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## population mean
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ##
  ## parameters sampled with a stationary (non-updating) sampler
  fup        ~ dnorm(mean=0.5, sd=0.025)     # fraction unbound in plasma
```

```
  bp         ~ dunif(0.75, 0.9)              # blood / plasma ratio
  fa         ~ dunif(0.9, 1)                 # absorption fraction
  fg         ~ dunif(0.9, 1)                 # fraction escaping gut metabolism
  Kpliver    ~ dnorm(mean=1,     sd=0.05)    # liver tissue:plasma part coef.
  LAb_CYP1A2 ~ dnorm(mean=52,    sd=2.6)     # pmol/mg protein
  MPPGL      ~ dnorm(mean=39.79, sd=2)       # mg microsomal proteins / g liver
  Km         ~ dnorm(mean=377,   sd=20)      # M-M constant (microM)
  CLrh       ~ dnorm(mean=0.31,  sd=0.015)   # renal clear healthy subj (L/h)
  Creat      ~ dnorm(mean=76.5,  sd=4)       # creat, male<61 y (micromol/L)
  ##
  ## default fixed (top parent) parameters
  density    <- 1080    # liver density (g/L)
  ##
  ## measurement error model
  b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
  b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
  ## for each subject
  for (i in 1:nsubjects) {
    Age[i]  ~ dunif(21, 31)  # Trembath group age range (years)
    Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
    ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
    Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
    ##
    ## dependent parameters, constant but computed from others
    tmp1[i]  <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    Ht[i]    ~ dnorm(tmp1[i], sd = 0.039 * tmp1[i])  # body height (cm)
    ##Ht[i]    <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    tmp2[i]  <- exp(2.643 + 0.0099 * Ht[i])
    BM[i]    ~ dnorm(tmp2[i], sd = 0.150 * tmp2[i])  # body mass (kg)
    ##BM[i]    <- exp(2.643 + 0.0099 * Ht[i])
    ##
    BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht[i], 0.725) # (m^2)
    CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age[i] - 20)) # cardiac output (L/h)
    Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
    Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
    Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
    Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
    LW[i]    <- Vh[i] * density             # liver weight (g)
    Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
    GFR[i]   <- (140 - Age[i])*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # ml/min
    RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
    CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
    Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
    ##
    ## likelihood for BM measurement
    BM_obs[i] ~ dnorm(BM[i], sd=0.5)    # body mass (kg)
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, microM
    Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates],
                                       times[i, 1:ntimes[i]],
                                       c(dose[i], ka[i], fa,  fg, fup,
                                         bp, Kpliver, Vh[i], Vsys[i],
                                         Qpv[i], Qh[i], Qha[i],
                                         CLr[i], Vmax_scaled[i], Km))
    for (j in 1:ntimes[i]) {
      ## measurement error SD from PLASMA concentration, microM
      sigma[i,j] <- b0 + b1 * Conc[i,j]
      ## data likelihood
      C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
    }
  }
}) # End myNimbleCode


## ------------------------------------------------------------------------
## Build and compile the model for predictions
```

```
## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj))))
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
BM_obs = misc$BM
BM_obs[13:18] = NA

constants <- list(nsubjects = nsubjects,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))

data <- list(BM_obs = BM_obs,
             C_plasma_obs = C_plasma_obs)

inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits)  # interpreted

Node.names = Rmodel$getNodeNames(includeData=T)

conf <- configureMCMC(Rmodel, thin=25, useConjugacy=F,
                      monitors=c("Age", "BM", "Vmax_pop", "ka_pop",
                                 "Vss_pop", "Vmax_popSD", "ka_popSD",
                                 "Vss_popSD",
                                 "fup", "bp", "fa", "fg",
                                 "Kpliver", "LAb_CYP1A2", "MPPGL",
                                 "Km", "CLrh", "Creat",
                                 "Vmax", "ka",
                                 "Vss", "b0", "b1", "Vh", "Vsys",
                                 "Qpv", "Qh", "Qha", "CLr", "Vmax_scaled",
                                 "logProb_Vmax_pop", "logProb_ka_pop",
                                 "logProb_Vss_pop",
                                 "logProb_Vmax_popSD", "logProb_ka_popSD",
                                 "logProb_Vss_popSD",
                                 "logProb_b0", "logProb_b1",
                                 "logProb_BM_obs", "logProb_C_plasma_obs"))

## ------------------------------------------------------------------------
## define stationary sampler
stationary_RW <- nimbleFunction(
  ##
  contains = sampler_BASE,
  ##
  setup = function(model, mvSaved, target, control) {
    # proposal standard deviation
    type      <- if(!is.null(control$type))     control$type     else "N"
    location  <- if(!is.null(control$location)) control$location else 1
    scale     <- if(!is.null(control$scale))    control$scale    else 1
    inf       <- if(!is.null(control$inf))      control$inf      else 1
    sup       <- if(!is.null(control$sup))      control$sup      else 1
    calcNodes <- model$getDependencies(target)
  },
  ##
  run = function() {
```

```
    ## generate proposal
    if (type == "N") {
      proposal <- rnorm(1, mean=location, sd=scale)
    } else {
      proposal <- runif(1, inf, sup)
    }
    ## store proposal into model
    model[[target]] <<- proposal
    model$calculate(calcNodes)
    ##
    ## jump <- TRUE
    ## keep the model and mvSaved objects consistent
    copy(from = model, to = mvSaved, row = 1,
         nodes = calcNodes, logProb = TRUE)
  },
  methods = list(reset = function () {})
)

## impose stationary sampler
conf$removeSamplers("fup")
conf$addSampler(target = "fup", type = 'stationary_RW',
                control = list(type="N", location=0.5, scale=0.025))
conf$removeSamplers("bp")
conf$addSampler(target = "bp", type = 'stationary_RW',
                control = list(type="U", location=0.815, scale=0.04,
                               inf=0.75, sup=0.9))
conf$removeSamplers("fa")
conf$addSampler(target = "fa", type = 'stationary_RW',
                control = list(type="U", location=1, scale=0.05,
                               inf=0.9, sup=1))
conf$removeSamplers("fg")
conf$addSampler(target = "fg", type = 'stationary_RW',
                control = list(type="U", location=1, scale=0.05,
                               inf=0.9, sup=1))
conf$removeSamplers("Kpliver")
conf$addSampler(target = "Kpliver", type = 'stationary_RW',
                control = list(type="N", location=1, scale=0.05))
conf$removeSamplers("LAb_CYP1A2")
conf$addSampler(target = "LAb_CYP1A2", type = 'stationary_RW',
                control = list(type="N", location=52, scale=2.6))
conf$removeSamplers("MPPGL")
conf$addSampler(target = "MPPGL", type = 'stationary_RW',
                control = list(type="N", location=39.79, scale=2))
conf$removeSamplers("Km")
conf$addSampler(target = "Km", type = 'stationary_RW',
                control = list(type="N", location=377, scale=20))
conf$removeSamplers("CLrh")
conf$addSampler(target = "CLrh", type = 'stationary_RW',
                control = list(type="N", location=0.31, scale=0.015))
conf$removeSamplers("Creat")
conf$addSampler(target = "Creat", type = 'stationary_RW',
                control = list(type="N", location=76.5, scale=4))

Rmcmc <- buildMCMC(conf)

## compile
Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

mysamples <- runMCMC(Cmcmc, niter=55000, nburnin=5000, nchains=4,
samplesAsCodaMCMC=T)

## End.
```

3.6. Code for model D with block sampling

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v9c: Extend the stationary sampler to do block sampling for
##      most top parent parameters previously fixed at default values.

library(nimble)

## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
dat # in mg/L
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]

## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver:
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
  return(result[,2] / parms[6]) # watch for correct index of bp
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall.
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## population mean
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ##
```

```
## parameters sampled with a stationary (non-updating) sampler
fup        ~ dnorm(mean=0.5, sd=0.025)     # fraction unbound in plasma
bp         ~ dunif(0.75, 0.9)              # blood / plasma ratio
## bp        ~ T(dnorm(0.815, 0.04), 0, 1)
fa         ~ dunif(0.9, 1)                 # absorption fraction
## fa        ~ T(dnorm(1, 0.05),     0, 1)
fg         ~ dunif(0.9, 1)                 # fraction escaping gut metabolism
## fg        ~ T(dnorm(1, 0.05),     0, 1)
Kpliver    ~ dnorm(mean=1,     sd=0.05)    # liver tissue:plasma part coef.
LAb_CYP1A2 ~ dnorm(mean=52,    sd=2.6)     # pmol/mg protein
MPPGL      ~ dnorm(mean=39.79, sd=2)       # mg microsomal proteins / g liver
Km         ~ dnorm(mean=377,   sd=20)      # M-M constant (microM)
CLrh       ~ dnorm(mean=0.31,  sd=0.015)   # renal clear healthy subj (L/h)
Creat      ~ dnorm(mean=76.5,  sd=4)       # creat, male<61 y (micromol/L)
##
## default fixed (top parent) parameters
density    <- 1080    # liver density (g/L)
##
## measurement error model
b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
## for each subject
for (i in 1:nsubjects) {
  Age[i]  ~ dunif(21, 31)  # Trembath group age range (years)
  Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
  ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
  Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
  ##
  ## dependent parameters, constant but computed from others
  tmp1[i]  <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
  Ht[i]    ~ dnorm(tmp1[i], sd = 0.039 * tmp1[i])  # body height (cm)
  ##Ht[i]    <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
  tmp2[i]  <- exp(2.643 + 0.0099 * Ht[i])
  BM[i]    ~ dnorm(tmp2[i], sd = 0.150 * tmp2[i])  # body mass (kg)
  ##BM[i]    <- exp(2.643 + 0.0099 * Ht[i])
  ##
  BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht[i], 0.725) # (m^2)
  CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age[i] - 20)) # cardiac output (L/h)
  Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
  Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
  Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
  Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
  LW[i]    <- Vh[i] * density             # liver weight (g)
  Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
  GFR[i]   <- (140 - Age[i])*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # ml/min
  RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
  CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
  Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
  ##
  ## likelihood for BM measurement
  BM_obs[i] ~ dnorm(BM[i], sd=0.5)    # body mass (kg)
  ##
  ## likelihood for concentration measurements
  ## Call the ODE solver to get predictions for each subject
  ## predictions are plasma concentrations, microM
  Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates],
                                     times[i, 1:ntimes[i]],
                                     c(dose[i], ka[i], fa,  fg, fup,
                                       bp, Kpliver, Vh[i], Vsys[i],
                                       Qpv[i], Qh[i], Qha[i],
                                       CLr[i], Vmax_scaled[i], Km))
  for (j in 1:ntimes[i]) {
    ## measurement error SD from PLASMA concentration, microM
    sigma[i,j] <- b0 + b1 * Conc[i,j]
    ## data likelihood
    C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
  }
}
```

```
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model for predictions:

## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj))))
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
BM_obs = misc$BM
BM_obs[13:18] = NA

constants <- list(nsubjects = nsubjects,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))

data <- list(BM_obs = BM_obs,
             C_plasma_obs = C_plasma_obs)

inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits)

Node.names = Rmodel$getNodeNames(includeData=T)
Rmodel$simulate(nodes=Node.names)
values(Rmodel, Node.names)

stationary.p = c("fup", "bp", "fa", "fg",
                 "Kpliver", "LAb_CYP1A2", "MPPGL", "Km", "CLrh", "Creat")

conf <- configureMCMC(Rmodel, thin=25, useConjugacy=F,
                      monitors=c("Age", "BM", "Vmax_pop", "ka_pop",
                                 "Vss_pop", "Vmax_popSD", "ka_popSD",
                                 "Vss_popSD",
                                 stationary.p,
                                 "Vmax", "ka",
                                 "Vss", "b0", "b1", "Vh", "Vsys",
                                 "Qpv", "Qh", "Qha", "CLr", "Vmax_scaled",
                                 "logProb_Vmax_pop", "logProb_ka_pop",
                                 "logProb_Vss_pop",
                                 "logProb_Vmax_popSD", "logProb_ka_popSD",
                                 "logProb_Vss_popSD",
                                 "logProb_b0", "logProb_b1",
                                 "logProb_BM_obs", "logProb_C_plasma_obs"))

## ------------------------------------------------------------------------
## define stationary sampler with block sampling of a vector,
stationary_RW_block <- nimbleFunction(
  ##
  contains = sampler_BASE,
  ##
  setup = function(model, mvSaved, target, control) {
    # proposal standard deviation
    n.target  <- length(target)
    distrib   <- if(!is.null(control$distrib))  control$distrib
```

```
                else rep("N", n.target)
    location  <- if(!is.null(control$location)) control$location
                else rep(1, n.target)
    scale     <- if(!is.null(control$scale))    control$scale
                else rep(1, n.target)
    inf       <- if(!is.null(control$inf))      control$inf
                else rep(1, n.target)
    sup       <- if(!is.null(control$sup))      control$sup
                else rep(1, n.target)
    proposal  <- numeric(n.target)
    calcNodes <- model$getDependencies(target) # target is a vector of labels
  },
  ##
  run = function() {
    ## generate proposal
    for (i in 1:n.target) {
      if (distrib[i] == "N") {
        proposal[i] <<- rnorm(1, mean=location[i], sd=scale[i])
      } else {
        proposal[i] <<- runif(1, inf[i], sup[i])
      }
    } # end for i
    ## store proposal into model
    values(model, target) <<- proposal
    model$calculate(calcNodes)
    ##
    ## jump <- TRUE
    ## keep the model and mvSaved objects consistent
    copy(from = model, to = mvSaved, row = 1,
         nodes = calcNodes, logProb = TRUE)
  },
  ##
  methods = list(reset = function () {})
) # end stationary_RW_block

## impose stationary sampler
conf$removeSamplers(stationary.p)

distribs = c("N", "U", "U", "U", "N", "N", "N", "N", "N", "N")
locations = c(0.5, 0.815, 1, 1, 1, 52, 39.79, 377, 0.31, 76.5)
scales = c(0.025, 0.04, 0.05, 0.05, 0.05, 2.6, 2, 20, 0.015, 4)
infs = c(0.025, 0.75, 0.9, 0.9, 0, 0, 0, 0, 0, 0)
sups = c(0.025, 0.9, 1, 1, 10, 100, 100, 1000, 1000, 1000)

conf$addSampler(target=stationary.p, type='stationary_RW_block',
                control=list(distrib=distribs, location=locations,
                             scale=scales, inf=infs, sup=sups))

Rmcmc <- buildMCMC(conf)

Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

mysamples <- runMCMC(Cmcmc, niter=55000, nburnin=5000, nchains=4,
                     samplesAsCodaMCMC=T)

## End.
```

3.7. Code for MCMC simulations of model E

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v10a: Full Bayesian inference on all appropriate nodes.

library(nimble)
```

```
## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]
## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
  return(result[,2] / parms[6]) # watch for correct index of bp
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## population mean
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ##
  ## parameters sampled with a stationary (non-updating) sampler
  fup        ~ dnorm(mean=0.5, sd=0.025)     # fraction unbound in plasma
  bp         ~ dunif(0.75, 0.9)              # blood / plasma ratio
  fa         ~ dunif(0.9, 1)                 # absorption fraction
  fg         ~ dunif(0.9, 1)                 # fraction escaping gut metabolism
  Kpliver    ~ dnorm(mean=1,     sd=0.05)    # liver tissue:plasma part coef.
  LAb_CYP1A2 ~ dnorm(mean=52,    sd=2.6)     # pmol/mg protein
  MPPGL      ~ dnorm(mean=39.79, sd=2)       # mg microsomal proteins / g liver
  Km         ~ dnorm(mean=377,   sd=20)      # M-M constant (microM)
  CLrh       ~ dnorm(mean=0.31,  sd=0.015)   # renal clear healthy subj (L/h)
  Creat      ~ dnorm(mean=76.5,  sd=4)       # creat, male<61 y (micromol/L)
  ##
```

```
  ## default fixed (top parent) parameters
  density    <- 1080    # liver density (g/L)
  ##
  ## measurement error model
  b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
  b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
  ## for each subject
  for (i in 1:nsubjects) {
    Age[i]  ~ dunif(21, 31)  # Trembath group age range (years)
    Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
    ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
    Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
    ##
    ## dependent parameters, constant but computed from others
    tmp1[i]  <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    Ht[i]    ~ dnorm(tmp1[i], sd = 0.039 * tmp1[i])  # body height (cm)
    tmp2[i]  <- exp(2.643 + 0.0099 * Ht[i])
    BM[i]    ~ dnorm(tmp2[i], sd = 0.150 * tmp2[i])  # body mass (kg)
    ##
    BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht[i], 0.725) # (m^2)
    CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age[i] - 20)) # cardiac output (L/h)
    Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
    Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
    Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
    Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
    LW[i]    <- Vh[i] * density             # liver weight (g)
    Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
    GFR[i]   <- (140 - Age[i])*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # ml/min
    RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
    CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
    Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
    ##
    ## likelihood for BM measurement
    BM_obs[i] ~ dnorm(BM[i], sd=0.5)    # body mass (kg)
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, microM
    Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates],
                                       times[i, 1:ntimes[i]],
                                       c(dose[i], ka[i], fa,  fg, fup,
                                         bp, Kpliver, Vh[i], Vsys[i],
                                         Qpv[i], Qh[i], Qha[i],
                                         CLr[i], Vmax_scaled[i], Km))
    for (j in 1:ntimes[i]) {
      ## measurement error SD from PLASMA concentration, microM
      sigma[i,j] <- b0 + b1 * Conc[i,j]
      ## data likelihood
      C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
    }
  }
}) # End myNimbleCode

## ------------------------------------------------------------------------
## Build and compile the model for predictions

## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj)))) ## too big...
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
```

```
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
BM_obs = misc$BM
BM_obs[13:18] = NA

constants <- list(nsubjects = nsubjects,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))

data <- list(BM_obs = BM_obs,
             C_plasma_obs = C_plasma_obs)

inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits)  # interpreted

Node.names = Rmodel$getNodeNames(includeData=T)

conf <- configureMCMC(Rmodel, thin=25, useConjugacy=F,
                      monitors=c("Age", "BM", "Vmax_pop", "ka_pop",
                                 "Vss_pop", "Vmax_popSD", "ka_popSD",
                                 "Vss_popSD",
                                 "fup", "bp", "fa", "fg",
                                 "Kpliver", "LAb_CYP1A2", "MPPGL",
                                 "Km", "CLrh", "Creat",
                                 "b0", "b1",
                                 "Vmax", "ka", "Vss",
                                 "Vh", "Vsys",
                                 "Qpv", "Qh", "Qha", "CLr", "Vmax_scaled",
                                 "logProb_Vmax_pop", "logProb_ka_pop",
                                 "logProb_Vss_pop",
                                 "logProb_Vmax_popSD", "logProb_ka_popSD",
                                 "logProb_Vss_popSD",
                                 "logProb_fup", "logProb_bp", "logProb_fa",
                                 "logProb_fg", "logProb_Kpliver",
                                 "logProb_LAb_CYP1A2", "logProb_MPPGL",
                                 "logProb_Km", "logProb_CLrh", "logProb_Creat",
                                 "logProb_b0", "logProb_b1",
                                 "logProb_BM_obs", "logProb_C_plasma_obs"))

Rmcmc <- buildMCMC(conf)

## compile
Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

mysamples <- runMCMC(Cmcmc, niter=55000, nburnin=5000, nchains=4,
samplesAsCodaMCMC=T)

## End.
```

3.8. Code for model E with block sampling

```
## ------------------------------------------------------------------------
## Nimble code
## Minimal PBPK Model for theophylline
## v10b: Full Bayesian inference on all appropriate nodes, with
##       blocking of the top parents previously fixed at default.

library(nimble)

## compile C ODE model for deSolve
system("R CMD SHLIB minPBPK.c")
```

```
dyn.load(paste("minPBPK", .Platform$dynlib.ext, sep = ""))

## ------------------------------------------------------------------------
## read the data
dat = read.csv("Theophylline all data.csv", as.is=F)
dat # in mg/L
Conc_microM = dat$Conc_mgL * 1E3 / 180.2 # in microM
dat = cbind(dat, Conc_microM)
## keep only the IR dosings
dat = dat[dat$Type=="IR",]
## remove the lines with NA data
dat = dat[which(!is.na(dat$Conc_microM)),]

## read body mass and dose data
misc = read.delim("Subjects_dose_BM.dat")

## ------------------------------------------------------------------------
## Define the model

## ------------------------------------------------------------------------
## ODE solver:
R_ode <- function(y, times, parms) {
  ## params = c(dose, ka, fa, fg, fup, bp, Kpliver, Vh, Vsys, Qpv, Qh, Qha,
  ##            CLr, Vmax_scaled, Km)
  result <- deSolve::ode(y, times, func="derivs", parms=parms,
                         rtol=1e-8, atol=1e-8, dllname="minPBPK",
                         initfunc = "minPBPK")
  ## extract systemic blood (microM) and convert to plasma (microM)
  return(result[,2] / parms[6]) # watch for correct index of bp
} # end of R_ode model solver

## ------------------------------------------------------------------------
## Nimble function with nimbleRcall:
nimble_ode <- nimbleRcall(
  prototype = function(
    y           = double(1), # vector
    times       = double(1), # vector
    parms       = double(1)  # vector
  ) {},
  returnType = double(1), # outcome is a vector
  Rfun = 'R_ode'
)

## ------------------------------------------------------------------------
## Hierarchical core Nimble (BUGS) code
myNimbleCode <- nimbleCode({
  ## population mean
  Vmax_pop ~ dunif(0.05, 50)  # (pmol/min/mg microsomal protein)
  ka_pop   ~ dunif(0.2,  40)  # first order absorption rate (1/h)
  Vss_pop  ~ dunif(0.05, 10)  # L per kg
  ## pop SDs
  Vmax_popSD ~ T(dnorm(1.1, 0.3), 1, 4)
  ka_popSD   ~ T(dnorm(1.1, 0.3), 1, 4)
  Vss_popSD  ~ T(dnorm(1.1, 0.3), 1, 4)
  ##
  ## parameters sampled with a stationary (non-updating) sampler
  fup        ~ dnorm(mean=0.5, sd=0.025)     # fraction unbound in plasma
  bp         ~ dunif(0.75, 0.9)              # blood / plasma ratio
  ## bp         ~ T(dnorm(0.815, 0.04), 0, 1)
  fa         ~ dunif(0.9, 1)                 # absorption fraction
  ## fa         ~ T(dnorm(1, 0.05),     0, 1)
  fg         ~ dunif(0.9, 1)                 # fraction escaping gut metabolism
  ## fg         ~ T(dnorm(1, 0.05),     0, 1)
  Kpliver    ~ dnorm(mean=1,     sd=0.05)    # liver tissue:plasma part coef.
  LAb_CYP1A2 ~ dnorm(mean=52,    sd=2.6)     # pmol/mg protein
  MPPGL      ~ dnorm(mean=39.79, sd=2)       # mg microsomal proteins / g liver
  Km         ~ dnorm(mean=377,   sd=20)      # M-M constant (microM)
  CLrh       ~ dnorm(mean=0.31,  sd=0.015)   # renal clear healthy subj (L/h)
```

```
  Creat      ~ dnorm(mean=76.5,  sd=4)       # creat, male<61 y (micromol/L)
  ##
  ## default fixed (top parent) parameters
  density    <- 1080    # liver density (g/L)
  ##
  ## measurement error model
  b0 ~ T(dnorm(0, 5.0), 0, ) # fixed    error part of sigma
  b1 ~ T(dnorm(0, 0.1), 0, ) # relative error part of sigma
  ## for each subject
  for (i in 1:nsubjects) {
    Age[i]  ~ dunif(21, 31)  # Trembath group age range (years)
    Vmax[i] ~ dlnorm(log(Vmax_pop), sdlog=log(Vmax_popSD))
    ka[i]   ~ dlnorm(log(ka_pop),   sdlog=log(ka_popSD))
    Vss[i]  ~ dlnorm(log(Vss_pop),  sdlog=log(Vss_popSD))
    ##
    ## dependent parameters, constant but computed from others
    tmp1[i]  <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    Ht[i]    ~ dnorm(tmp1[i], sd = 0.039 * tmp1[i])  # body height (cm)
    ##Ht[i]    <- 175.32 + 0.1113 * Age[i] - 0.0025 * Age[i] * Age[i]
    tmp2[i]  <- exp(2.643 + 0.0099 * Ht[i])
    BM[i]    ~ dnorm(tmp2[i], sd = 0.150 * tmp2[i])  # body mass (kg)
    ##BM[i]    <- exp(2.643 + 0.0099 * Ht[i])
    ##
    BSA[i]   <- 0.00718 * pow(BM[i], 0.425) * pow(Ht[i], 0.725) # (m^2)
    CO[i]    <- BSA[i] * 60 * (3 - 0.01 * (Age[i] - 20)) # cardiac output (L/h)
    Qha[i]   <- 0.196 * CO[i]               # hep art. blood flow (L/h)
    Qpv[i]   <- 0.065 * CO[i]               # portal vein blood flow (L/h)
    Qh[i]    <- Qha[i] + Qpv[i]             # hepatic blood flow (L/h)
    Vh[i]    <- 0.722 * pow(BSA[i], 1.176)  # liver tissue volume (L)
    LW[i]    <- Vh[i] * density             # liver weight (g)
    Vsys[i]  <- Vss[i] * BM[i] - Vh[i] * Kpliver / bp
    GFR[i]   <- (140 - Age[i])*BM[i] / (0.814 * Creat) * 1.73 / BSA[i] # ml/min
    RF[i]    <- GFR[i] / 130.0              # renal function (unitless);
    CLr[i]   <- RF[i] * CLrh                # plasma renal clearance (L/h)
    Vmax_scaled[i] <- Vmax[i] * LAb_CYP1A2 * MPPGL * LW[i] * 6e-5
    ##
    ## likelihood for BM measurement
    BM_obs[i] ~ dnorm(BM[i], sd=0.5)    # body mass (kg)
    ##
    ## likelihood for concentration measurements
    ## Call the ODE solver to get predictions for each subject
    ## predictions are plasma concentrations, microM
    Conc[i, 1:ntimes[i]] <- nimble_ode(y[1:nstates],
                                       times[i, 1:ntimes[i]],
                                       c(dose[i], ka[i], fa,  fg, fup,
                                         bp, Kpliver, Vh[i], Vsys[i],
                                         Qpv[i], Qh[i], Qha[i],
                                         CLr[i], Vmax_scaled[i], Km))
    for (j in 1:ntimes[i]) {
      ## measurement error SD from PLASMA concentration, microM
      sigma[i,j] <- b0 + b1 * Conc[i,j]
      ## data likelihood
      C_plasma_obs[i,j] ~ dnorm(Conc[i,j], sd=sigma[i,j])
    }
  }
}) # End myNimbleCode


## ------------------------------------------------------------------------
## Build and compile the model for predictions:

## initial values for state variables Csys and Cliv
y = c(Csys=0, Cliv=0)

## define time and plasma concentration values from dataset
subjects = unique(dat$Subject)
nsubjects = length(subjects)
times = array(0, dim=c(nsubjects, max(tabulate(dat$Subj))))
```

```
C_plasma_obs = times ## init
for (i in 1:nsubjects) {
  myindex = which(dat$Subj == subjects[i]) # rows for subject i
  tmp = dat$Time_h[myindex]                # times for subject i
  times[i,1:length(tmp)] = tmp
  tmp = dat$Conc_microM[myindex]           # plasma data for subject i
  C_plasma_obs[i,1:length(tmp)] = tmp
}
BM_obs = misc$BM
BM_obs[13:18] = NA

constants <- list(nsubjects = nsubjects,
                  dose = misc$Dose,
                  ntimes = tabulate(dat$Subj),
                  times = times,
                  nstates = length(y))

data <- list(BM_obs = BM_obs,
             C_plasma_obs = C_plasma_obs)

inits <- list(y = y)

Rmodel <- nimbleModel(myNimbleCode, constants, data, inits)

Node.names = Rmodel$getNodeNames(includeData=T)
Node.names

block.p = c("fup", "bp", "fa", "fg",
            "Kpliver", "LAb_CYP1A2", "MPPGL", "Km", "CLrh", "Creat")

conf <- configureMCMC(Rmodel, thin=25, useConjugacy=F,
                      monitors=c("Age", "BM", "Vmax_pop", "ka_pop",
                                 "Vss_pop", "Vmax_popSD", "ka_popSD",
                                 "Vss_popSD",
                                 block.p,
                                 "b0", "b1",
                                 "Vmax", "ka", "Vss",
                                 "Vh", "Vsys",
                                 "Qpv", "Qh", "Qha", "CLr", "Vmax_scaled",
                                 "logProb_Vmax_pop", "logProb_ka_pop",
                                 "logProb_Vss_pop",
                                 "logProb_Vmax_popSD", "logProb_ka_popSD",
                                 "logProb_Vss_popSD",
                                 "logProb_fup", "logProb_bp", "logProb_fa",
                                 "logProb_fg", "logProb_Kpliver",
                                 "logProb_LAb_CYP1A2", "logProb_MPPGL",
                                 "logProb_Km", "logProb_CLrh", "logProb_Creat",
                                 "logProb_b0", "logProb_b1",
                                 "logProb_BM_obs", "logProb_C_plasma_obs"))

## do block sampling for the block.p parameters
conf$removeSamplers(block.p)

## prepare a covariance matrix guess for block.p
vars = c(0.025, 0.04, 0.05, 0.05, 0.05, 2.6, 2, 20, 0.015, 4)^2
propCov = diag(vars, nrow=length(block.p))

conf$addSampler(target=block.p, type='RW_block', control=list(propCov=propCov))

Rmcmc <- buildMCMC(conf)

Cmodel = compileNimble(Rmodel, showCompilerOutput = F)
Cmcmc  = compileNimble(Rmcmc, project = Rmodel)

mysamples <- runMCMC(Cmcmc, niter=55000, nburnin=5000, nchains=4,
samplesAsCodaMCMC=T)

## End.
```

3.9. Data files

5.1.1 File Theophylline all data.csv

```
"Subject","Wt_kg","Dose_mgkg","Dose_mg","Time_h","Conc_mgL","Type"
"A",79.6,4.02,319.992,0,0.74,"IR"
"A",79.6,4.02,319.992,0.25,2.84,"IR"
"A",79.6,4.02,319.992,0.57,6.57,"IR"
"A",79.6,4.02,319.992,1.12,10.5,"IR"
"A",79.6,4.02,319.992,2.02,9.66,"IR"
"A",79.6,4.02,319.992,3.82,8.58,"IR"
"A",79.6,4.02,319.992,5.1,8.36,"IR"
"A",79.6,4.02,319.992,7.03,7.47,"IR"
"A",79.6,4.02,319.992,9.05,6.89,"IR"
"A",79.6,4.02,319.992,12.12,5.94,"IR"
"A",79.6,4.02,319.992,24.37,3.28,"IR"
"B",58.2,5.5,320.1,0,0.24,"IR"
"B",58.2,5.5,320.1,0.37,2.89,"IR"
"B",58.2,5.5,320.1,0.77,5.22,"IR"
"B",58.2,5.5,320.1,1.02,6.41,"IR"
"B",58.2,5.5,320.1,2.05,7.83,"IR"
"B",58.2,5.5,320.1,3.55,10.21,"IR"
"B",58.2,5.5,320.1,5.05,9.18,"IR"
"B",58.2,5.5,320.1,7.08,8.02,"IR"
"B",58.2,5.5,320.1,9.38,7.14,"IR"
"B",58.2,5.5,320.1,12.1,5.68,"IR"
"B",58.2,5.5,320.1,23.7,2.42,"IR"
"C",65,4.92,319.8,0,0,"IR"
"C",65,4.92,319.8,0.25,4.86,"IR"
"C",65,4.92,319.8,0.5,7.24,"IR"
"C",65,4.92,319.8,0.98,8,"IR"
"C",65,4.92,319.8,1.98,6.81,"IR"
"C",65,4.92,319.8,3.6,5.87,"IR"
"C",65,4.92,319.8,5.02,5.22,"IR"
"C",65,4.92,319.8,7.03,4.45,"IR"
"C",65,4.92,319.8,9.03,3.62,"IR"
"C",65,4.92,319.8,12.12,2.69,"IR"
"C",65,4.92,319.8,24.08,0.86,"IR"
"D",60.5,5.3,320.65,0,0,"IR"
"D",60.5,5.3,320.65,0.25,1.25,"IR"
"D",60.5,5.3,320.65,0.5,3.96,"IR"
"D",60.5,5.3,320.65,1,7.82,"IR"
"D",60.5,5.3,320.65,2,9.72,"IR"
"D",60.5,5.3,320.65,3.52,9.75,"IR"
"D",60.5,5.3,320.65,5.07,8.57,"IR"
"D",60.5,5.3,320.65,7.07,6.59,"IR"
"D",60.5,5.3,320.65,9.03,6.11,"IR"
"D",60.5,5.3,320.65,12.05,4.57,"IR"
"D",60.5,5.3,320.65,24.15,1.17,"IR"
"E",72.4,4.4,318.56,0,0,"IR"
"E",72.4,4.4,318.56,0.27,1.72,"IR"
"E",72.4,4.4,318.56,0.52,7.91,"IR"
"E",72.4,4.4,318.56,1,8.31,"IR"
"E",72.4,4.4,318.56,1.92,8.33,"IR"
"E",72.4,4.4,318.56,3.5,6.85,"IR"
"E",72.4,4.4,318.56,5.02,6.08,"IR"
"E",72.4,4.4,318.56,7.03,5.4,"IR"
"E",72.4,4.4,318.56,9,4.55,"IR"
"E",72.4,4.4,318.56,12,3.01,"IR"
"E",72.4,4.4,318.56,24.3,0.9,"IR"
"F",70.5,4.53,319.365,0,0,"IR"
"F",70.5,4.53,319.365,0.27,4.4,"IR"
"F",70.5,4.53,319.365,0.58,6.9,"IR"
"F",70.5,4.53,319.365,1.02,8.2,"IR"
"F",70.5,4.53,319.365,2.02,7.8,"IR"
```

```
"F",70.5,4.53,319.365,3.62,7.5,"IR"
"F",70.5,4.53,319.365,5.08,6.2,"IR"
"F",70.5,4.53,319.365,7.07,5.3,"IR"
"F",70.5,4.53,319.365,9,4.9,"IR"
"F",70.5,4.53,319.365,12.15,3.7,"IR"
"F",70.5,4.53,319.365,24.17,1.05,"IR"
"G",72.7,4.4,319.88,0,0,"IR"
"G",72.7,4.4,319.88,0.35,1.89,"IR"
"G",72.7,4.4,319.88,0.6,4.6,"IR"
"G",72.7,4.4,319.88,1.07,8.6,"IR"
"G",72.7,4.4,319.88,2.13,8.38,"IR"
"G",72.7,4.4,319.88,3.5,7.54,"IR"
"G",72.7,4.4,319.88,5.02,6.88,"IR"
"G",72.7,4.4,319.88,7.02,5.78,"IR"
"G",72.7,4.4,319.88,9.02,5.33,"IR"
"G",72.7,4.4,319.88,11.98,4.19,"IR"
"G",72.7,4.4,319.88,24.65,1.15,"IR"
"H",54.6,5.86,319.956,0,0,"IR"
"H",54.6,5.86,319.956,0.3,2.02,"IR"
"H",54.6,5.86,319.956,0.52,5.63,"IR"
"H",54.6,5.86,319.956,1,11.4,"IR"
"H",54.6,5.86,319.956,2.02,9.33,"IR"
"H",54.6,5.86,319.956,3.5,8.74,"IR"
"H",54.6,5.86,319.956,5.02,7.56,"IR"
"H",54.6,5.86,319.956,7.02,7.09,"IR"
"H",54.6,5.86,319.956,9.1,5.9,"IR"
"H",54.6,5.86,319.956,12,4.37,"IR"
"H",54.6,5.86,319.956,24.35,1.57,"IR"
"I",80,4,320,0,0,"IR"
"I",80,4,320,0.27,1.29,"IR"
"I",80,4,320,0.58,3.08,"IR"
"I",80,4,320,1.15,6.44,"IR"
"I",80,4,320,2.03,6.32,"IR"
"I",80,4,320,3.57,5.53,"IR"
"I",80,4,320,5,4.94,"IR"
"I",80,4,320,7,4.02,"IR"
"I",80,4,320,9.22,3.46,"IR"
"I",80,4,320,12.1,2.78,"IR"
"I",80,4,320,23.85,0.92,"IR"
"J",64.6,4.95,319.77,0,0.15,"IR"
"J",64.6,4.95,319.77,0.25,0.85,"IR"
"J",64.6,4.95,319.77,0.5,2.35,"IR"
"J",64.6,4.95,319.77,1.02,5.02,"IR"
"J",64.6,4.95,319.77,2.02,6.58,"IR"
"J",64.6,4.95,319.77,3.48,7.09,"IR"
"J",64.6,4.95,319.77,5,6.66,"IR"
"J",64.6,4.95,319.77,6.98,5.25,"IR"
"J",64.6,4.95,319.77,9,4.39,"IR"
"J",64.6,4.95,319.77,12.05,3.53,"IR"
"J",64.6,4.95,319.77,24.22,1.15,"IR"
"K",70.5,4.53,319.365,0,0,"IR"
"K",70.5,4.53,319.365,0.25,3.05,"IR"
"K",70.5,4.53,319.365,0.52,3.05,"IR"
"K",70.5,4.53,319.365,0.98,7.31,"IR"
"K",70.5,4.53,319.365,2.02,7.56,"IR"
"K",70.5,4.53,319.365,3.53,6.59,"IR"
"K",70.5,4.53,319.365,5.05,5.88,"IR"
"K",70.5,4.53,319.365,7.15,4.73,"IR"
"K",70.5,4.53,319.365,9.07,4.57,"IR"
"K",70.5,4.53,319.365,12.1,3,"IR"
"K",70.5,4.53,319.365,24.12,1.25,"IR"
"L",86.4,3.1,267.84,0,0,"IR"
"L",86.4,3.1,267.84,0.3,7.37,"IR"
"L",86.4,3.1,267.84,0.63,9.03,"IR"
"L",86.4,3.1,267.84,1.05,7.14,"IR"
"L",86.4,3.1,267.84,2.02,6.33,"IR"
"L",86.4,3.1,267.84,3.53,5.66,"IR"
"L",86.4,3.1,267.84,5.02,5.67,"IR"
```

```
"L",86.4,3.1,267.84,7.17,4.24,"IR"
"L",86.4,3.1,267.84,8.8,4.11,"IR"
"L",86.4,3.1,267.84,11.6,3.16,"IR"
"L",86.4,3.1,267.84,24.43,1.12,"IR"
"M",NA,NA,190,0,0,"IR"
"M",NA,NA,190,1,5.8,"IR"
"M",NA,NA,190,2,5.5,"IR"
"M",NA,NA,190,3,5.5,"IR"
"M",NA,NA,190,4,4.7,"IR"
"M",NA,NA,190,6,4.5,"IR"
"M",NA,NA,190,8,NA,"IR"
"M",NA,NA,190,10,3.4,"IR"
"M",NA,NA,190,12,NA,"IR"
"N",NA,NA,190,0,0,"IR"
"N",NA,NA,190,1,5.7,"IR"
"N",NA,NA,190,2,6.5,"IR"
"N",NA,NA,190,3,7.2,"IR"
"N",NA,NA,190,4,5.7,"IR"
"N",NA,NA,190,6,3.9,"IR"
"N",NA,NA,190,8,3.9,"IR"
"N",NA,NA,190,10,3.2,"IR"
"N",NA,NA,190,12,3.4,"IR"
"O",NA,NA,190,0,0,"IR"
"O",NA,NA,190,1,9,"IR"
"O",NA,NA,190,2,5.3,"IR"
"O",NA,NA,190,3,5.3,"IR"
"O",NA,NA,190,4,5.3,"IR"
"O",NA,NA,190,6,4.3,"IR"
"O",NA,NA,190,8,3.5,"IR"
"O",NA,NA,190,10,3.2,"IR"
"O",NA,NA,190,12,2.6,"IR"
"P",NA,NA,190,0,0,"IR"
"P",NA,NA,190,1,5.2,"IR"
"P",NA,NA,190,2,5.1,"IR"
"P",NA,NA,190,3,4.5,"IR"
"P",NA,NA,190,4,4.1,"IR"
"P",NA,NA,190,6,4.1,"IR"
"P",NA,NA,190,8,4.1,"IR"
"P",NA,NA,190,10,3.2,"IR"
"P",NA,NA,190,12,2.6,"IR"
"Q",NA,NA,190,0,0,"IR"
"Q",NA,NA,190,1,4.2,"IR"
"Q",NA,NA,190,2,4.1,"IR"
"Q",NA,NA,190,3,3.5,"IR"
"Q",NA,NA,190,4,2.9,"IR"
"Q",NA,NA,190,6,2.5,"IR"
"Q",NA,NA,190,8,1.4,"IR"
"Q",NA,NA,190,10,1.2,"IR"
"Q",NA,NA,190,12,1.1,"IR"
"R",NA,NA,190,0,0,"IR"
"R",NA,NA,190,1,3.2,"IR"
"R",NA,NA,190,2,3.3,"IR"
"R",NA,NA,190,3,3.2,"IR"
"R",NA,NA,190,4,2.8,"IR"
"R",NA,NA,190,6,1.8,"IR"
"R",NA,NA,190,8,1.4,"IR"
"R",NA,NA,190,10,1.1,"IR"
"R",NA,NA,190,12,1.1,"IR"
```

5.1.2 File Subjects_dose_BM.dat

BM_kg	Dose_microM
79.6	1776
58.2	1776
65	1775
60.5	1779
72.4	1768
70.5	1772
72.7	1775
54.6	1776
80	1776
64.6	1775
70.5	1772
86.4	1486
80.58	1054
80.58	1054
80.58	1054
80.58	1054
80.58	1054
80.58	1054