P.O. Box 614
Augusta, ME 04332-0614
info@metruninstitute.org

2 Tunxis Road, Suite 112
Tariffville, CT 06081
Phone: 860.735.7043
Fax: 860.760.6014

Prototype PKPD Model Library for WinBUGS

User Manual: Version 1.2

BUGSModelLibrary version 1.2: Changes from version 1.1.1

- Added the model function OneCptKaModel that, unlike OneCptModel, is parametrized in terms of k_a instead of $k_a - \frac{CL}{V}$.
- Added the model function TwoCptKaModel that, unlike TwoCptModel, is parametrized in terms of k_a instead of $k_a - \lambda_1$.

BUGSModelLibrary version 1.1.1: Changes from version 1.1

- Corrected a bug (in Pmetrics/PKModeks.odc) that resulted in incorrect calculations when a dosing event with RATE ¿ 0 and an observation event occurred at the same time.

BUGSModelLibrary version 1.1: Changes from version 1.0

- Added the option of using LSODA as the ODE solver. The Runge-Kutta solver remains as the alternative option.
- Corrected omission that prevented user access to the one compartment model.
- Example script and model files renamed to be more descriptive.
- Miscellaneous minor changes to examples scripts.
- Added examples illustrating use of the one compartment model and the LSODA ODE solver.
- Revised the user manual to reflect above changes.

Introduction

Metrum Institute has developed a prototype PKPD model library for use with WinBUGS 1.4.3. The current version includes:

- Specific linear compartmental models:
  - One compartment model with first order absorption
  - Two compartment model with elimination from and first order absorption into central compartment

- General linear compartmental model described by a matrix exponential
- General compartmental model described by a system of first order ODEs

The models and data format are based on NONMEM®[1]/NMTRAN/PREDPP conventions including:

- Recursive calculation of model predictions
  - This permits piecewise constant covariate values
- Bolus or constant rate inputs into any compartment
- Handles single dose, multiple dose and steady-state dosing histories
- Implemented NMTRAN data items include:
  - TIME, EVID, CMT, AMT, RATE, ADDL, II, SS

The library provides BUGS language functions that calculate amounts in each compartment.

Implementation details.

- The BUGS language interface is implemented using WBDev (http://www.winbugs-development.org.uk/wbdev.html).
- An extensible object-oriented programming approach is used to facilitate development of additional models.
- The core procedures of the library are programmed in Component Pascal
- One and two compartment models:
  - Analytical calculations programmed in Component Pascal.
- General linear compartmental models:
  - Component Pascal wrapper procedures that call DLL's compiled from FORTRAN.
  - The DLL's are generated from Expokit (http://www.maths.uq.edu.au/expokit/) for matrix exponential calculations and LAPACK (http://www.netlib.org/lapack/) for matrix calculations.
- General compartmental models:
  - Numerical solution of ODE's. The user may chose between the following methods:
    - A Runge-Kutta 4th/5th order method
      - Uses ODE solver code provided (but not documented) in WinBUGS.
      - Programmed in Component Pascal.
    - LSODA: Livermore Solver for Ordinary Differential Equations, with Automatic method switching for stiff and nonstiff problems. It uses Adams methods (predictor-corrector) in the nonstiff case, and Backward Differentiation Formula (BDF) methods (the Gear methods) in the stiff case.
      - Component Pascal wrapper procedure that calls a DLL compiled from FORTRAN.
  - Steady-state calculations based on solution of a boundary value problem. Requires numerical solution of algebraic equations. Uses procedures from LibSolve (http://www.zinnamturm.eu/downloadsIN.htm#Lib).
  - Programmed in Component Pascal.

WARNING: The current version of the WinBUGS PKPD model library is a *prototype*. It is being released for review and comment, and to support limited research applications. It has not been rigorously tested and should not be used for critical applications without further testing or cross-checking by comparison with other methods.

[1]NONMEM® is licensed and distributed by ICON Development Solutions.

Development plans. Our current plans for further development of the BUGS PKPD library include the following:

- Revise prototype library using better programming practices to enhance extensibility, maintainability, reliability and efficiency. Specific changes include:
  - Use of factory design pattern.
  - Adaptive allocation of arrays to eliminate limits imposed by fixed size arrays while also avoiding inefficiencies due to overuse of dynamic allocation.
  - Implement additional ODE solvers, e.g., LSODES, CVODE, etc..
  - Improve method used for general compartmental model steady-state calculations.
- Implement additional models to be determined in collaboration with potential users.
- Develop documentation for both users and developers.
- Develop OpenBUGS interface for the model library.
- Execute rigorous and well-documented testing of the model library components.
- Develop software for installation and qualification of WinBUGS and the model library
- Conduct beta testing

For additional information please contact Bill Gillespie (billg@metruminstitute.org).

Installation

Install WinBUGS 1.4.3. Installation of WinBUGS 1.4.3 involves 3 steps: (1) installation of WinBUGS 1.4, (2) installation of the WinBUGS 1.4.3 patch and (3) registration to obtain a license for use until the end of the year.

(1) Install WinBUGS 1.4
  (a) Download WinBUGS14.exe from `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/WinBUGS14.exe`.
  (b) Double click on WinBUGS14.exe. This installs WinBUGS 1.4 in the "c:\Program Files" directory.
  (c) Go to "c:\Program Files\WinBUGS14". Create a shortcut to the file "c:\Program Files\WinBUGS14\WinBUGS14.exe". Move the shortcut to the desktop.

(2) Install the WinBUGS 1.4.3 patch
  (a) Download the WinBUGS 1.4.3 patch from `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/WinBUGS14_cumulative_patch_No3_06_08_07_RELEASE.txt`.
  (b) Launch WinBUGS by double clicking on the WinBUGS shortcut.
  (c) Drag and drop the WinBUGS 1.4.3 patch into the WinBUGS window.
  (d) Select "Decode" from the "Tools" menu; the "Decode" dialogue box should appear.
  (e) Click on the "Decode All" button to install the upgrade patch. You may be prompted to create two new directories during the installation process – click on "OK" for each one.
  (f) Exit WinBUGS to complete the installation.
  (g) When you launch WinBUGS the "About WinBUGS" command under the Help menu should produce a dialog box showing the WinBUGS version is 1.4.3.

(3) Install key for unrestricted use
  (a) Download the key for unrestricted use from `http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/WinBUGS14_immortality_key.txt`.
  (b) Launch WinBUGS and drag and drop the file containing the key into the WinBUGS window.
  (c) Select "Decode" from the "Tools" menu; the "Decode" dialogue box should appear.

(d) Click on the "Decode All" button to install the upgrade patch. After following the instructions given in the key, check that the Keys.ocf file in ..\WinBUGS14\Bugs\Code has been updated.

Install WBDev.

(1) Download wbdev.exe from `http://www.winbugs-development.org.uk/files/wbdev.exe`.
(2) Double click on wbdev.exe. This will deposit a file called wbdev_01_09_04.txt into your c:\Program Files\WinBUGS14 directory.
(3) Launch WinBUGS by double clicking on the WinBUGS shortcut.
(4) Drag and drop wbdev_01_09_04.txt into the WinBUGS window.
(5) Select "Decode" from the "Tools" menu; the "Decode" dialogue box should appear.
(6) Click on the "Decode All" button to install WBDev. You may be prompted to create new directories during the installation process – click on "OK" for each one.
(7) Exit WinBUGS to complete the installation.

Install BlackBox 1.5 compiler.

(1) Download SetupBlackBox15.exe from `http://www.oberon.ch/zip/SetupBlackBox15.exe`.
(2) Double click on SetupBlackBox15.exe to launch set up program and complete the set up accepting all defaults.

Add WinBUGS to BlackBox. The following steps install WinBUGS capabilities within the BlackBox compiler. This permits development, compilation, debugging and use of new WinBUGS functions within a single software environment. This capability is also used by the WinBUGS model library for implementing new models.

(1) Navigate to the c:\Program Files\WinBUGS14 directory.
(2) Select and copy all files and subdirectories within the WinBUGS14 directory (press Ctrl+A or select Select All from the Edit menu then press Ctrl+C or select Copy from the Edit menu).
(3) Navigate to the "c:\Program Files\BlackBox Component Builder 1.5" directory.
(4) Paste the WinBUGS files and subdirectories to that directory (press Ctrl+V or select Paste from the Edit menu). Select Yes to All if prompted about replacing existing files.
(5) Rename the "BlackBox Component Builder 1.5" directory as "BlackBoxWinBUGS". This is not necessary, but may be desirable to distinguish it from a standard BlackBox Component Builder installation. In addition the examples provided with the BUGSModelLibrary distribution assume that the directory is named BlackBoxWinBUGS.

Install WinBUGS PKPD model library and examples.

(1) The zip file BUGSModelLibrary.zip contains the WinBUGS PKPD model library materials including the software, computer scripts and data for running the examples described in subsequent sections. If you unzip the contents of BUGSModelLibrary.zip to the top level of your C: drive, you should now have the folder C:\BUGSModelLibrary. If you prefer to place the BUGSModelLibrary directory elsewhere that is OK, but you will have to edit the example computer scripts to indicate the path to the course directory.
(2) Launch BlackBox by double clicking on the BlackBox shortcut on the desktop.
(3) Drag and drop BUGSModelLibraryInstallPatch.txt from the BUGSModelLibrary directory into the BlackBox window.
(4) Select "Decode" from the "Tools" menu; the "Decode" dialogue box should appear.

(5) Click on the "Decode All" button to install the model library. You may be prompted to create new directories during the installation process – click on "OK" for each one.
(6) Exit BlackBox to complete the installation.

Using the WinBUGS model library

All WinBUGS model library functions have the form:

<modelName>(`time, amt, rate, ii, evid, cmt, addl, ss, theta`)

where `time`, `amt`, `rate`, `ii`, `evid`, `cmt`, `addl` and `ss` are equal length vectors and are defined identically to the NONMEM variables of the same name. `theta` may be either a vector of parameters or a matrix with a number of rows equal to the length of the time vector. In the latter case the i^{th} row contains a vector of model parameters for the time interval (`time[i-1]`, `time[i]`]. <modelName> is the name of a built-in model, such as `OneCptModel` or `TwoCptModel`, or a user-defined name in the case of models constructed by user specification of a system of differential equations.

Simulated examples. To describe and illustrate the use of the model library, simulated results for the following scenario have been generated. The 3 different simulated outcomes (plasma drug concentrations and 2 different PD measurements) exercise the 3 main components of the prototype model library.

- Phase 1 study in healthy volunteers
  - Parallel dose-escalation design
  - 25 subjects per dose arm
  - Single doses
    - Placebo, 5, 10, 20 and 40 mg
  - PK: plasma concentrations of parent drug (c)
  - PD: 2 different responses
    - Response 1: Emax function of effect compartment concentration (R_1)
    - Response 2: Indirect effect model with drug effect on kout (inhibitory Emax) (R_2)
  - PK and PD measured at 0, 0.125, 0.25, 0.5, 0.75, 1, 2, 3, 4, 6, 8, 12, 18 and 24 hours after dose.
- Phase IIa trial in patients:
  - Multiple doses
    - 20 mg bid (q12h) x 7 days
  - 100 patients per treatment arm
  - Sparse PK data (3-6 samples/patient)

The plasma drug concentrations (c) and the two PD responses (R_1 and R_2) were simulated according to the following equations.

$$\begin{aligned}
\log\left(c_{ij}\right) &\sim N\left(\log\left(\widehat{c}_{ij}\right), \sigma^2\right) \\
\widehat{c}_{ij} &= f_{2cpt}\left(t_{ij}, D_j, \tau_j, CL_j, Q_j, V_{1j}, V_{2j}, k_{aj}\right) \\
\log\left(CL_j, Q_j, V_{ssj}, k_{aj}\right) &\sim \\
& N\left(\log\left(\widehat{CL}\left(\frac{bw_j}{70}\right)^{0.75}, \widehat{Q}\left(\frac{bw_j}{70}\right)^{0.75}, \widehat{V}_{ss}\left(\frac{bw_j}{70}\right), \widehat{k}_a\right), \Omega\right) \\
V_{1j} &= f_{V_1} V_{ssj} \quad V_{2j} = \left(1 - f_{V_1}\right) V_{ssj} \\
\left(\widehat{CL}, \widehat{Q}, \widehat{V}_{ss}, \widehat{k}_a, f_{V_1}\right) &= \left(10 \text{ L/h}, 15 \text{ L/h}, 140 \text{ L}, 2 \text{ h}^{-1}, 0.25\right) \\
\Omega &= \begin{pmatrix} 0.25^2 & 0 & 0 & 0 \\ 0 & 0.25^2 & 0 & 0 \\ 0 & 0 & 0.25^2 & 0 \\ 0 & 0 & 0 & 0.25^2 \end{pmatrix} \quad \sigma = 0.1
\end{aligned}$$

$$\begin{aligned}
R_{1ij} &\sim N\left(\widehat{R}_{1ij}, \sigma^2_{R_1}\right) \\
\widehat{R}_{1ij} &= \frac{E_{max} c_{eij}^{\gamma}}{EC_{50j}^{\gamma} + c_{eij}^{\gamma}} \\
c'_{e\cdot j} &= k_{e0j}\left(c_{\cdot j} - c_{e\cdot j}\right) \\
\log\left(EC_{50j}, k_{e0j}\right) &\sim N\left(\log\left(\widehat{EC}_{50}, \widehat{k}_{e0}\right), \Omega_{R_1}\right) \\
\left(E_{max}, \widehat{EC}_{50}, \gamma, \widehat{k}_{e0}\right) &= \left(100, 100.7, 1.07, 1\right) \\
\Omega_{R_1} &= \begin{pmatrix} 0.2^2 & 0 \\ 0 & 0.25^2 \end{pmatrix} \quad \sigma_{R_1} = 10
\end{aligned}$$

$$\begin{aligned}
\log\left(R_{2ij}\right) &\sim N\left(\log\left(\widehat{R}_{2ij}\right), \sigma^2_{R_2}\right) \\
\widehat{R}'_{2\cdot j} &= k_{out,j}\left(R_{20j} - \left(1 - \frac{\widehat{c}_{\cdot j}}{EC_{50j}^{R_2} + \widehat{c}_{\cdot j}}\widehat{R}_{2\cdot j}\right)\right) \\
\log\left(R_{20j}, EC_{50j}^{R_2}, k_{out,j}\right) &\sim N\left(\log\left(\widehat{R}_{20}, \widehat{EC}_{50}^{R_2}, \widehat{k}_{out}\right), \Omega_{R_2}\right) \\
\left(\widehat{R}_{20}, \widehat{EC}_{50}^{R_2}, \widehat{k}_{out}\right) &= \left(100, 200, 0.5\right) \\
\Omega_{R_2} &= \begin{pmatrix} 0.25^2 & 0 & 0 \\ 0 & 0.25^2 & 0 \\ 0 & 0 & 0.25^2 \end{pmatrix} \quad \sigma_{R_2} = 0.15
\end{aligned}$$

The simulations were generated using R 2.6.1. Numerical solution of the differential equations required for simulating Response 2 was done using the R package odesolve. For each of the three examples the simulated data were analyzed using WinBUGS 1.4.3 modified by addition of the prototype PKPD model library. Data management, launching WinBUGS, and analysis of the MCMC samples were done using R 2.6.1 with the R packages R2WinBUGS and coda. Three MCMC chains of 10000 iterations were simulated. The first 4000

iterations of each chain were discarded and every 10th sample was retained. Thus 1800 MCMC samples were used for subsequent analyses.

Linear one and two compartment models. Linear one and two compartment models with first order absorption are pre-compiled models in the WinBUGS model library (see Figure 1). As with NONMEM's PREDPP models, bolus or constant rate inputs may be put into any compartment. The arguments of the pre-compiled model functions are described in Table 1.

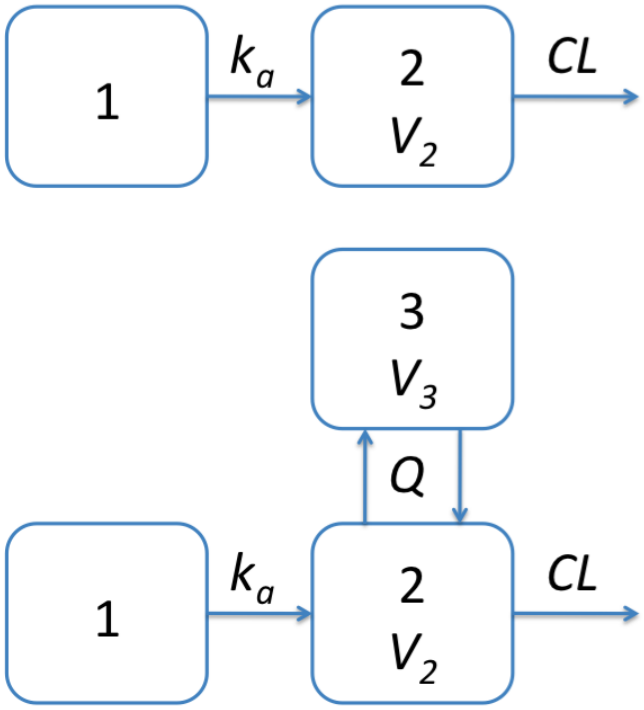

FIGURE 1. One and two compartment models with first order absorption implemented in the WinBUGS PKPD model library.

Example 1: Two compartment model with 1st order absorption. Example 1 involves fitting a 2 compartment model with first order absorption to the combined Phase I and II PK data. This illustrates the use of the precompiled 2 compartment model from the library. Figure 2 is a BUGS language model demonstrating the use of TwoCptModel. Correct use of the library model functions requires that the user pass the entire event history (observation and dosing events ordered by time) for an individual to the function. Thus the BUGS model shows the call to TwoCptModel within a loop over the individual subjects rather than over the individual observations. The BUGS model file (`twoCptExample.txt`) as well as an R script (`twoCptExample.R`) and data files (`fxa.data.csv` and `fxa.data2.csv`) implementing this example are contained in the BUGSModelLibrary directory. They implement a relatively general strategy using the model library functions given a NONMEM formatted data set. To run the examples the user must also install R and the R package R2WinBUGS.

Results. The MCMC history plots (Figure 3) suggest that the 3 chains have converged to common distributions for all of the key model parameters. The individual fits to the plasma concentration data (Figure 4) are in close agreement with the data—not surprising since the fitted model is identical to the one used to simulate the data. Similarly the parameter estimates summarized in Table 2 are consistent with the values used for simulation.

FIGURE 2. Example 1: BUGS language model for fitting a two compartment model.

```
model
{

   for(i in 1:nsub){

      # Inter-patient variation
      logtheta[i, 1:5] ~ dmnorm(logthetaMean[i, 1:5], omega.inv[1:5, 1:5])
      logthetaMean[i, 1] <- logCLHat + 0.75*log(weight[start[i]]/70) # CL
      logthetaMean[i, 2] <- logQHat + 0.75*log(weight[start[i]]/70)  # Q
      logthetaMean[i, 3] <- logV1Hat + log(weight[start[i]]/70)         # V1
      logthetaMean[i, 4] <- logV2Hat + log(weight[start[i]]/70)         # V2
      logthetaMean[i, 5] <- logDkaHat                                     # ka - alpha1
      theta[i,6] <- 1 # F1
      theta[i,7] <- 1 # F2
      theta[i,8] <- 1 # F3
      theta[i,9] <- 0 # tlag1
      theta[i,10] <- 0 # tlag2
      theta[i,11] <- 0 # tlag3

      for(j in 1:5){
         log(theta[i,j]) <- logtheta[i,j]
      }

      # Call to PK model library to calculate amount in each compartment at each time
      xhat[start[i]:end[i],1:3] <- TwoCptModel(time[start[i]:end[i]], amt[start[i]:end[i]],
         rate[start[i]:end[i]], ii[start[i]:end[i]], evid[start[i]:end[i]],
         cmt[start[i]:end[i]], addl[start[i]:end[i]], ss[start[i]:end[i]], theta[i,])

   }

   for(i in 1:nobs){

      logCobs[i] ~ dnorm(logCHat[i],tau)
      CHat[i] <- 1000*xhat[i,2]/theta[subject[i],3]
      logCHat[i] <- log(max(CHat[i],eps))

   }

   # Prior distributions
   logCLHat ~ dnorm(0,1.0E-6)
   logQHat ~ dnorm(0,1.0E-6)
   logV1Hat ~ dnorm(0,1.0E-6)
   logV2Hat ~ dnorm(0,1.0E-6)
   logDkaHat ~ dnorm(0,1.0E-6)
   log(CLHat) <- logCLHat
   log(QHat) <- logQHat
   log(V1Hat) <- logV1Hat
   log(V2Hat) <- logV2Hat
   log(DkaHat) <- logDkaHat
   tau <- 1/(sigma*sigma)
   sigma ~ dunif(0,1000)
   omega.inv[1:5, 1:5] ~ dwish(omega.inv.prior[1:5, 1:5], 5)
   omega[1:5, 1:5] <- inverse(omega.inv[1:5, 1:5])
   eps <- 1.0E-6

}
```

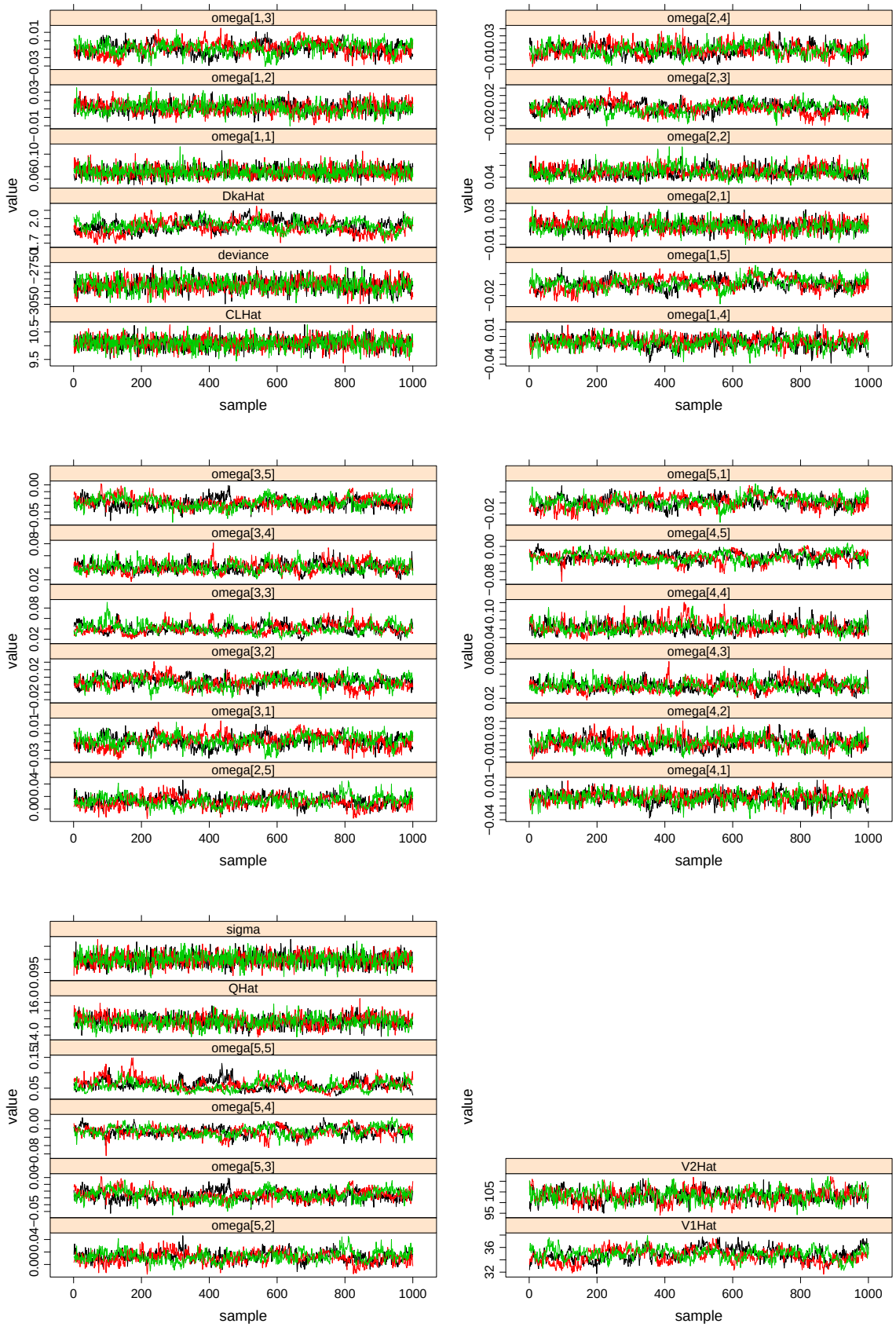

FIGURE 3. Example 1: MCMC history plots for the parameters of a two compartment model with first order absorption.

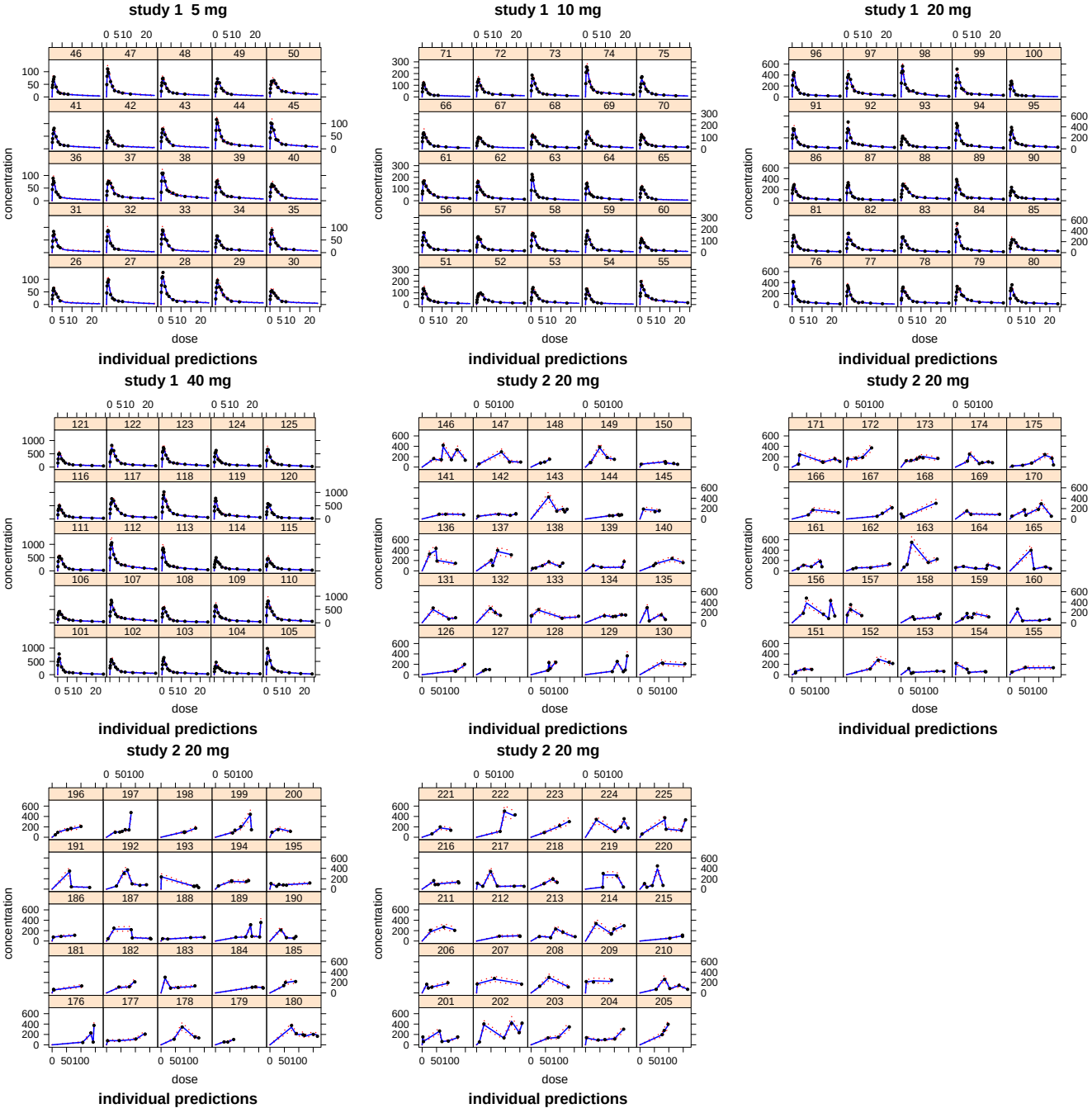

FIGURE 4. Example 1: Predicted (posterior median and 90% credible intervals) and observed plasma drug concentrations.

TABLE 1. Arguments of the pre-compiled model functions in the WinBUGS PKPD model library.

model	WinBUGS function name	argument names	model parameters in `theta`
one compartment model with first order absorption $\left(k_a > \frac{CL}{V_2}\right)$	`OneCptModel`	`time`, `amt`, `rate`, `ii`, `evid`, `cmt`, `addl`, `ss`	CL, V_2, $k_a - \frac{CL}{V_2}$, F_1, F_2, t_{lag1}, t_{lag2}
two compartment model with first order absorption $(k_a > \lambda_1)$	`TwoCptModel`	`time`, `amt`, `rate`, `ii`, `evid`, `cmt`, `addl`, `ss`	$CL, Q, V_2, V_3, k_a - \lambda_1$ [a], F_1, F_2, F_3, t_{lag1}, t_{lag2}, t_{lag2}
one compartment model with first order absorption $(k_a > 0)$	`OneCptKaModel`	`time`, `amt`, `rate`, `ii`, `evid`, `cmt`, `addl`, `ss`	CL, V_2, k_a, F_1, F_2, t_{lag1}, t_{lag2}
two compartment model with first order absorption $(k_a > 0)$	`TwoCptKaModel`	`time`, `amt`, `rate`, `ii`, `evid`, `cmt`, `addl`, `ss`	CL, Q, V_2, V_3, k_a, F_1, F_2, F_3, t_{lag1}, t_{lag2}, t_{lag2}

[a] $\lambda_1 = \frac{k_{10}+k_{12}+k_{21}-\sqrt{(k_{10}+k_{12}+k_{21})^2-4k_{10}k_{21}}}{2}$ where $k_{10} = \frac{CL}{V_2}$, $k_{12} = \frac{Q}{V_2}$ and $k_{21} = \frac{Q}{V_3}$.

TABLE 2. Example 1: Summary of the MCMC simulations of the marginal posterior distributions of the model parameters.

	Mean	SD	Naive SE	Time-series SE	2.50%	25%	50%	75%	97.50%	Effective N
deviance	-2900	47.1	0.86	1.72	-2990	-2930	-2900	-2870	-2810	902
CLHat	10.1	0.201	0.00367	0.00531	9.69	9.96	10.1	10.2	10.5	1740
QHat	14.8	0.331	0.00604	0.0128	14.2	14.6	14.8	15.1	15.5	647
V1Hat	34.8	0.983	0.0179	0.0622	32.9	34.1	34.8	35.5	36.7	130
V2Hat	103	2.86	0.0521	0.149	97.7	101	103	105	109	345
DkaHat	1.91	0.0704	0.00129	0.00445	1.77	1.86	1.91	1.96	2.05	109
sigma	0.1	0.00226	4.13E-05	6.40E-05	0.0957	0.0984	0.0999	0.102	0.104	1640
omega[1,1]	0.0713	0.00804	0.000147	0.000226	0.0569	0.0658	0.0706	0.0765	0.0884	1550
omega[1,2]	0.0116	0.00627	0.000115	0.000258	-0.000457	0.00739	0.0113	0.0157	0.0243	624
omega[1,3]	-0.00889	0.00699	0.000128	0.000412	-0.0234	-0.0133	-0.00865	-0.0042	0.00402	172
omega[1,4]	-0.00803	0.00782	0.000143	0.000387	-0.0244	-0.013	-0.0078	-0.00272	0.00632	445
omega[1,5]	0.00167	0.00986	0.00018	6.00E-04	-0.0179	-0.00497	0.00182	0.00836	0.0209	142
omega[2,2]	0.0489	0.00876	0.00016	0.000384	0.0344	0.0425	0.0479	0.0542	0.0681	474
omega[2,3]	0.00383	0.00687	0.000126	0.00041	-0.0102	-0.000687	0.00406	0.00856	0.0164	182
omega[2,4]	0.0103	0.00805	0.000147	0.000436	-0.00519	0.00466	0.0101	0.0157	0.0266	300
omega[2,5]	0.0123	0.00825	0.000151	0.000461	-0.00405	0.00686	0.0122	0.0177	0.0286	188
omega[3,3]	0.0409	0.00914	0.000167	0.000614	0.0256	0.0345	0.0397	0.0464	0.0614	154
omega[3,4]	0.041	0.00815	0.000149	0.000461	0.0268	0.0354	0.0406	0.046	0.0587	316
omega[3,5]	-0.0256	0.00725	0.000132	0.000404	-0.04	-0.0305	-0.0258	-0.0208	-0.011	198
omega[4,4]	0.0634	0.0125	0.000229	0.000675	0.0424	0.0545	0.0621	0.0706	0.091	388
omega[4,5]	-0.0258	0.0108	0.000197	0.000697	-0.0475	-0.0327	-0.0254	-0.0186	-0.00532	178
omega[5,5]	0.0604	0.0166	0.000303	0.00113	0.0342	0.0484	0.0585	0.0703	0.0969	97.6

General linear compartmental models. A general linear compartment model refers to a model that may be described in terms of a system of first order linear differential equations with (piecewise) constant coefficients, i.e., a differential equation of the form:

$$x'(t) = Kx(t)$$

where K is a matrix. For example K for a two compartment model with first order absorption is:

$$K = \begin{bmatrix} -k_a & 0 & 0 \\ k_a & -(k_{10}+k_{12}) & k_{21} \\ 0 & k_{12} & -k_{21} \end{bmatrix}$$

Implementation of a new BUGS language model for a general linear compartment model involves the following steps:

(1) Create a new Component Pascal model library component
   (a) Navigate to "Program Files\BlackBox Component Builder 1.5\Pmetrics\Mod".
   (b) Create a copy of TwoCptEffCptModel.odc within the same directory and rename the copy to something descriptive of the model you are creating. This file serves as a template that you can modify to create your own model.
   (c) Launch BlackBox Component Builder 1.5.
   (d) Drag and drop your model component into BlackBox (or use File >> Open in the usual way).
   (e) Replace the red text in the template with code implementing your model.
      - The red text in the first and last lines of the file should be changed to match the name of the file (excluding the .odc extension).
      - Nonzero elements of K should be assigned to the appropriate elements of `kMatrix` inside the `UserKMatrix` procedure. Note that indices of vectors and matrices in Component Pascal start at 0 instead of 1. All values will be passed to your model from WinBUGS via the vector `theta`. What each element of `theta` represents is up to the user.
      - Add/delete variable name declarations in the section following `VAR` inside the `UserKMatrix` procedure.
      - Edit the parameters in the `InitModel` procedure.
         - `m.nParameter`: total number of parameters (elements of `theta`). This should include an F and a t_{lag} for each compartment.
         - `m.F1Index`: index of F_1 in `theta`. The model library assumes that the F's are adjacent and ordered by compartment number in `theta`.
         - `m.tlag1Index`: index of t_{lag1} in `theta`. The model library assumes that the t_{lag}'s are adjacent and ordered by compartment number in `theta`.
         - `m.nCmt`: number of compartments or, equivalently, the number of rows (or columns) in K.
   (f) Save the revised model component.
   (g) Compile the model component. From the Dev menu choose "Compile and Unload". If there are no compilation errors then "Ok" will appear at the bottom of the BlackBox window. Otherwise you will get an error message usually directing you to the offending lines of code. Modify and recompile as necessary.

(2) Create an interface between WinBUGS and your new model using WBDev.
   (a) Navigate to "Program Files\BlackBox Component Builder 1.5\WBDev\Mod".
   (b) Create a copy of TwoCptEffCptModel.odc within the same directory and rename the copy to the same name as the model component file you created.
   (c) Drag and drop the new WBDev file (the file you just created) into BlackBox (or use File >> Open in the usual way).
   (d) Replace all 4 instances of `TwoCptEffCptModel` with the name of your new model component.
   (e) Compile the new WBDev file.
   (f) Navigate to "Program Files\BlackBox Component Builder 1.5\WBDev\Rsrc".
   (g) Drag and drop Functions.odc into BlackBox.
   (h) Duplicate the line

```
v <- "TwoCptEffCptModel"(v,v,v,v,v,v,v,v,v) "WBDevTwoCptEffCptModel.Install"
```

   (i) Edit the copied line, replacing `TwoCptEffCptModel` with the name of your new model component.
      - Actually it is only necessary to use the model component name on the right hand side. The name on the left hand side is the name of the BUGS language function you are creating and it may be different from the model component name.

(3) Exit BlackBox. The new function will only be available after you restart BlackBox.

Example 2: General linear compartmental model. In example 2 both the PK and response 1 from Phases I and II data were fit simultaneously. The PK data were described by a 2 compartment model with first order absorption, and the response 1 data were described by an Emax function of the effect compartment concentrations. This illustrates the use of the library component for general linear compartmental models described by a matrix exponential. The BUGS language model is shown in Figure 5; the non-zero elements of K are specified in the Component Pascal file TwoCptEffCptModel.odc as shown in Figure 6. The BUGS model file (`effectCptExample.txt`) as well as an R script (`effectCptExample.R`) and data files (`fxa.data.csv` and `fxa.data2.csv`) implementing this example are contained in the BUGSModelLibrary directory.

Results. The MCMC history plots (Figure 7) suggest that the 3 chains converge to a common distribution. However a relatively large amount of autocorrelation is evident some parameters, e.g., $\widehat{k}_{e0}$, $\widehat{EC}_{50}$, $\omega_{k_{e0}}$ and $\omega_{EC_{50}}$ indicating that a larger number of iterations would be desirable, particularly if precise inferences regarding those parameters are desired. The model fit to the plasma concentration data was comparable to that in example 1 (not shown). The model described the response 1 data quite well (Figure 8), and the parameter estimates (Table 3) were similar to the values used for simulation.

TABLE 3. Example 2: Summary of the MCMC simulations of the marginal posterior distributions of the model parameters.

	Mean	SD	Naive SE	Time-series SE	2.50%	25%	50%	75%	97.50%	Effective N
deviance	13400	49.9	1.18	2.05	13300	13300	13400	13400	13500	507
CLHat	10.1	0.211	0.00498	0.00645	9.68	9.95	10.1	10.2	10.5	1020
QHat	14.9	0.341	0.00804	0.0143	14.2	14.6	14.8	15.1	15.5	595
V1Hat	34.7	0.879	0.0207	0.0548	33	34.1	34.7	35.3	36.4	109
V2Hat	103	2.96	0.0698	0.158	97.8	101	103	105	109	308
DkaHat	1.9	0.0695	0.00164	0.00414	1.77	1.85	1.9	1.95	2.04	117
ke0Hat	1.06	0.056	0.00132	0.00399	0.937	1.03	1.07	1.1	1.16	77.8
EC50Hat	104	2.02	0.0476	0.133	100	103	104	106	109	210
omegaKe0	0.106	0.0581	0.00137	0.00433	0.034	0.0594	0.0893	0.14	0.243	23.8
omegaEC50	0.0715	0.0407	0.00096	0.00254	0.0103	0.036	0.0718	0.101	0.151	41.3
sigmaC	0.0996	0.00221	5.20E-05	7.07E-05	0.0956	0.0981	0.0996	0.101	0.104	1130
sigmaFxa	16.3	0.262	0.00618	0.00588	15.8	16.2	16.4	16.5	16.9	1880
omega[1,1]	0.0718	0.0083	0.000196	0.000274	0.0575	0.066	0.0713	0.0768	0.0901	1010
omega[1,2]	0.0124	0.00642	0.000151	0.000275	0.000229	0.00794	0.0123	0.0168	0.0256	328
omega[1,3]	-0.00907	0.0072	0.00017	0.000423	-0.0236	-0.0137	-0.00893	-0.00441	0.00485	125
omega[1,4]	-0.00815	0.00856	0.000202	0.000439	-0.0247	-0.014	-0.00796	-0.00235	0.00793	356
omega[1,5]	0.00075	0.0107	0.000253	0.000659	-0.0196	-0.00666	0.00071	0.00792	0.0219	115
omega[2,2]	0.0502	0.00896	0.000211	0.000389	0.035	0.0437	0.0496	0.0556	0.0697	419
omega[2,3]	0.00484	0.00634	0.00015	0.000355	-0.00799	0.000658	0.00487	0.00933	0.0164	271
omega[2,4]	0.0115	0.00818	0.000193	0.000442	-0.00385	0.00582	0.0112	0.0168	0.0281	341
omega[2,5]	0.0103	0.00901	0.000212	0.000506	-0.00765	0.00442	0.0105	0.0164	0.0278	139
omega[3,3]	0.0395	0.00935	0.00022	0.000636	0.0235	0.0327	0.0385	0.0455	0.0605	155
omega[3,4]	0.0408	0.00903	0.000213	0.000522	0.0249	0.0344	0.04	0.0464	0.0599	193
omega[3,5]	-0.0249	0.00882	0.000208	0.000516	-0.0397	-0.0301	-0.0261	-0.0207	-0.00283	100
omega[4,4]	0.0659	0.0134	0.000316	0.000734	0.0428	0.0563	0.0652	0.0747	0.0949	290
omega[4,5]	-0.0275	0.0116	0.000274	0.00069	-0.0504	-0.0349	-0.0277	-0.0206	-0.00318	140
omega[5,5]	0.066	0.018	0.000425	0.00119	0.0377	0.0528	0.0641	0.0758	0.105	115

General compartmental models. The WinBUGS model library may be used to fit models described by a system of first order ordinary differential equations (ODE's), i.e., differential equations of the form:

$$x'(t) = f(t, x(t))$$

where x and f are vector-valued functions.

Implementation of a new BUGS language model for a general compartment model involves the following steps:

(1) Create a new Component Pascal model library component
   (a) Navigate to "Program Files\BlackBox Component Builder 1.5\Pmetrics\Mod".

FIGURE 5. Example 2: BUGS language model for fitting a PKPD model consisting of a two compartment PK submodel and a PD submodel with an effect compartment.

```
model
{

   for(i in 1:nsub){

      # Inter-patient variation
      logtheta[i, 1:5] ~ dmnorm(logthetaMean[i, 1:5], omega.inv[1:5, 1:5])
      logthetaMean[i, 1] <- logCLHat + 0.75*log(weight[start[i]]/70) # CL
      logthetaMean[i, 2] <- logQHat + 0.75*log(weight[start[i]]/70)  # Q
      logthetaMean[i, 3] <- logV1Hat + log(weight[start[i]]/70)         # V1
      logthetaMean[i, 4] <- logV2Hat + log(weight[start[i]]/70)         # V2
      logthetaMean[i, 5] <- logDkaHat                                     # ka - alpha1
      logtheta[i,6] ~ dnorm(logKe0Hat,tauKe0)                                    # ke0
      theta[i,7] <- 1 # F1
      theta[i,8] <- 1 # F2
      theta[i,9] <- 1 # F3
      theta[i,10] <- 0 # tlag1
      theta[i,11] <- 0 # tlag2
      theta[i,12] <- 0 # tlag3

      logEC50[i] ~ dnorm(logEC50Hat,tauEC50)
      log(EC50[i]) <- logEC50[i]

      for(j in 1:6){
         log(theta[i,j]) <- logtheta[i,j]
      }

      # Call to PK model library to calculate amount in each compartment at each time
      xhat[start[i]:end[i],1:4] <- TwoCptEffCptModel(time[start[i]:end[i]],
         amt[start[i]:end[i]], rate[start[i]:end[i]], ii[start[i]:end[i]], evid[start[i]:end[i]],
         cmt[start[i]:end[i]], addl[start[i]:end[i]], ss[start[i]:end[i]], theta[i,])

   }

   for(i in 1:nobs){

      logCobs[i] ~ dnorm(logCHat[i],tauC)
      CHat[i] <- 1000*xhat[i,2]/theta[subject[i],3]
      logCHat[i] <- log(max(CHat[i],eps))

      fxa[i] ~ dnorm(fxaMean[i],tauFxa)
      fxaMean[i] <- Emax*CeHat[i]/(EC50[subject[i]]+CeHat[i])
      CeHat[i] <- 1000*xhat[i,4]/theta[subject[i],3]
      logCeHat[i] <- log(max(CeHat[i],eps))

   }

   # Prior distributions
   logCLHat ~ dnorm(0,1.0E-6)
   logQHat ~ dnorm(0,1.0E-6)
   logV1Hat ~ dnorm(0,1.0E-6)
   logV2Hat ~ dnorm(0,1.0E-6)
   logDkaHat ~ dnorm(0,1.0E-6)
   log(CLHat) <- logCLHat
   log(QHat) <- logQHat
   log(V1Hat) <- logV1Hat
   log(V2Hat) <- logV2Hat
   log(DkaHat) <- logDkaHat
   logKe0Hat ~ dnorm(0,1.0E-6)
   logEC50Hat ~ dnorm(0,1.0E-6)
   log(ke0Hat) <- logKe0Hat
   log(EC50Hat) <- logEC50Hat
   tauC <- 1/(sigmaC*sigmaC)
   sigmaC ~ dunif(0,1000)
   tauFxa <- 1/(sigmaFxa*sigmaFxa)
   sigmaFxa ~ dunif(0,1000)
   omega.inv[1:5, 1:5] ~ dwish(omega.inv.prior[1:5, 1:5], 5)
   omega[1:5, 1:5] <- inverse(omega.inv[1:5, 1:5])
   omegaKe0 ~ dunif(0,1000)
   tauKe0 <- 1/(omegaKe0*omegaKe0)
   omegaEC50 ~ dunif(0,1000)
   tauEC50 <- 1/(omegaEC50*omegaEC50)
   eps <- 1.0E-6
   Emax <- 100

}
```

FIGURE 6. Example 2: Specification of the non-zero elements of K in the Component Pascal file TwoCptEffCptModel.odc.

```
PROCEDURE UserKMatrix(IN theta: ARRAY OF REAL; nCmt: INTEGER):
    POINTER TO ARRAY OF ARRAY OF REAL;
 VAR
    kMatrix: POINTER TO ARRAY OF ARRAY OF REAL;
    i, j: INTEGER;
    CL, Q, V2, V3, dka, ke0, k10, k12, k21, ksum, ka: REAL;
BEGIN
    NEW(kMatrix,nCmt,nCmt);
    (* Initialize to all zeros *)
    FOR i := 0 TO nCmt-1 DO;
    FOR j := 0 TO nCmt-1 DO;
       kMatrix[i,j] := 0;
    END; END;
    (* 2 cpt model with 1st order absorption *)
    CL := theta[0];
    Q := theta[1];
    V2 := theta[2];
    V3 := theta[3];
    dka := theta[4];
    ke0 := theta[5];
    k10 := CL/V2;
    k12 := Q/V2;
    k21 := Q/V3;
    ksum := k10 + k12 + k21;
    ka := dka + (ksum - Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;
    (* Assign nonzero rate constants *)
    kMatrix[0,0] := -ka;
    kMatrix[1,0] := ka;
    kMatrix[1,1] := -(k10+k12);
    kMatrix[1,2] := k21;
    kMatrix[2,1] := k12;
    kMatrix[2,2] := -k21;
    kMatrix[3,1] := ke0;
    kMatrix[3,3] := -ke0;

    RETURN kMatrix;

 END UserKMatrix;

 PROCEDURE (m: MatExpModel) InitModel*;
 BEGIN
    m.nParameter := 12;
    m.F1Index := 6;
    m.tlag1Index := 10;
    m.nCmt := 4;
 END InitModel;
```

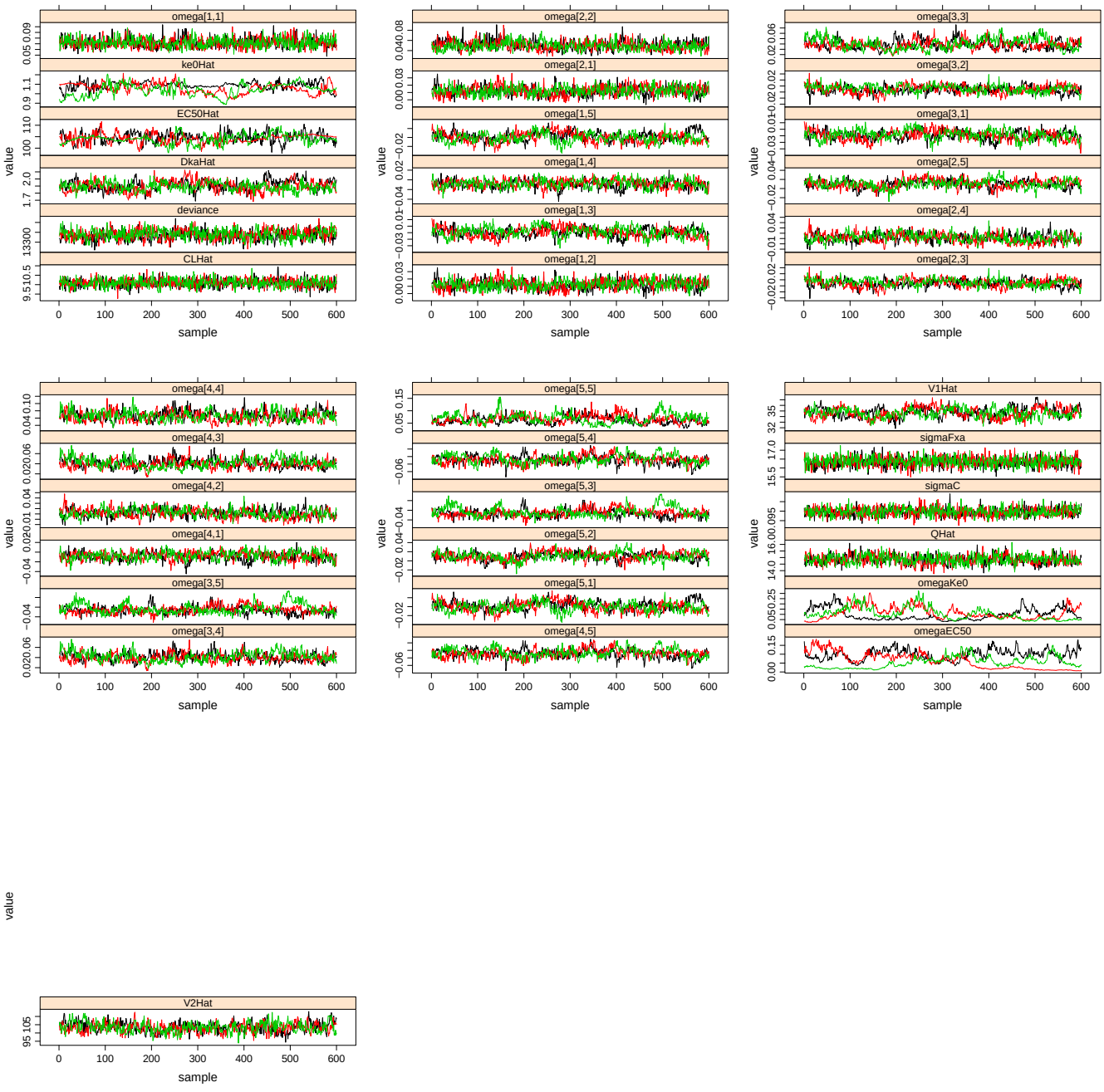

FIGURE 7. Example 2: MCMC history plots for the parameters of a PKPD model where the PK is described by a two compartment model with first order absorption and the PD by an Emax function of the effect compartment concentration.

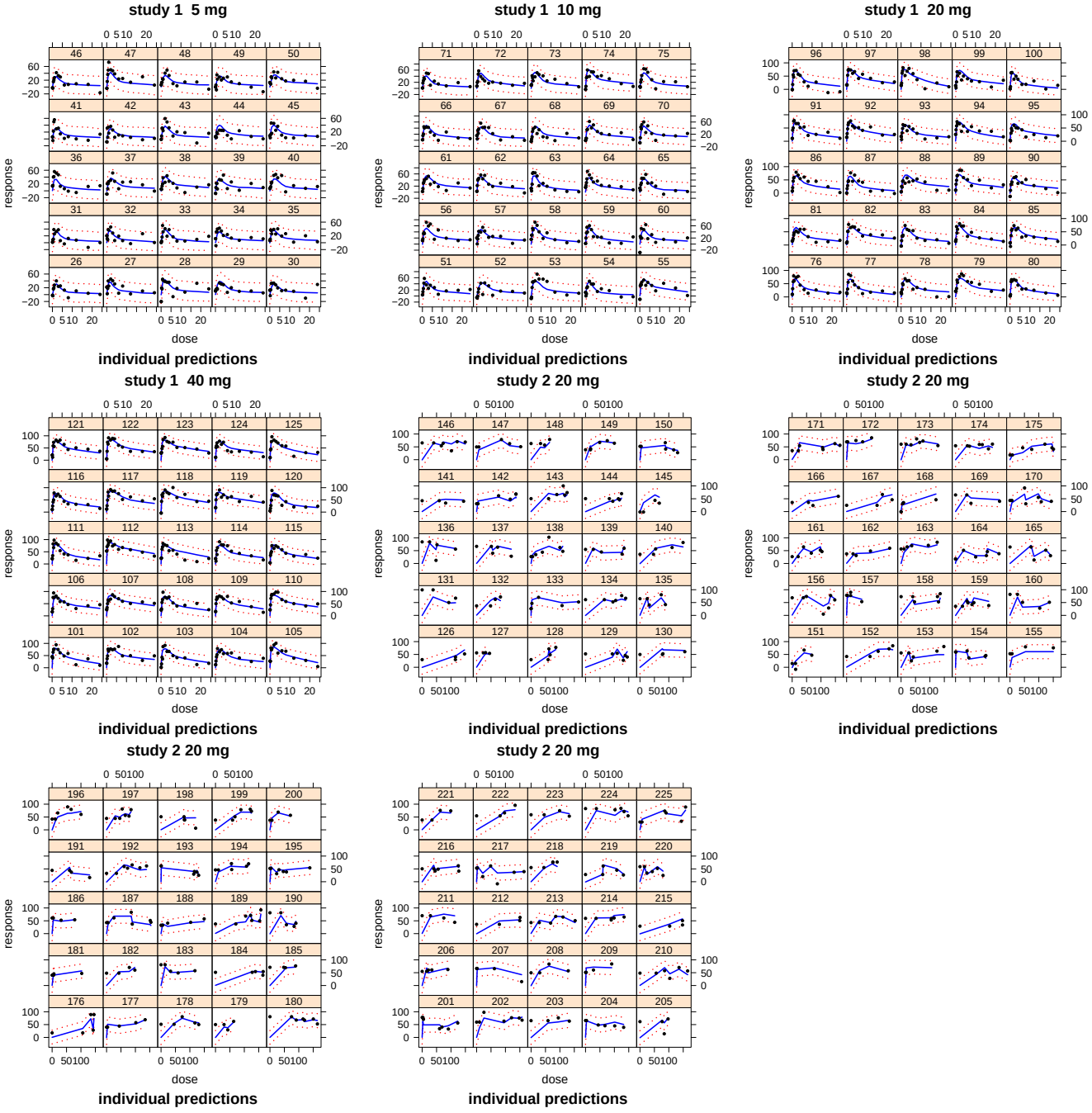

FIGURE 8. Example 2: Predicted (posterior median and 90% credible intervals) and observed response 1 measurements.

(b) Create a copy of TwoCptIndEff1ModelRK45.odc (or TwoCptIndEff1ModelLSODA.odc) within the same directory and rename the copy to something descriptive of the model you are creating. This file serves as a template that you can modify to create your own model.
- Model components derived from TwoCptIndEff1ModelRK45.odc use the Runge-Kutta ODE solver. Those derived from TwoCptIndEff1ModelLSODA.odc use the Livermore solver (LSODA).

(c) Launch BlackBox Component Builder 1.5.

(d) Drag and drop your model component into BlackBox (or use File >> Open in the usual way).

(e) Replace the red text in the template with code implementing your model.
- The red text in the first and last lines of the file should be changed to match the name of the file (excluding the .odc extension).
- Specify the number of differential equations (usually the number of compartments) by editing the line
  ```
  numEq = 4;
  ```
- Write the differential equations specifying the model inside the `UserDerivatives` procedure. Follow the syntax illustrated in the `TwoCptIndEff1ModelRK45` example. Note that indices of vectors and matrices in Component Pascal start at 0 instead of 1. All values will be passed to your model from WinBUGS via the vector `theta`. What each element of `theta` represents is up to the user.
- Add/delete variable name declarations in the section following `VAR` inside the `UserDerivatives` procedure.
- Edit the parameters in the `InitModel` procedure.
  - `m.nParameter`: total number of parameters (elements of `theta`). This should include an F and a t_{lag} for each compartment.
  - `m.F1Index`: index of F_1 in `theta`. The model library assumes that the F's are adjacent and ordered by compartment number in `theta`.
  - `m.tlag1Index`: index of t_{lag1} in `theta`. The model library assumes that the t_{lag}'s are adjacent and ordered by compartment number in `theta`.
  - `m.nCmt`: number of compartments or, equivalently, the number of rows (or columns) in K.

(f) Save the revised model component.

(g) Compile the model component. From the Dev menu choose "Compile and Unload". If there are no compilation errors then "Ok" will appear at the bottom of the BlackBox window. Otherwise you will get an error message usually directing you to the offending lines of code. Modify and recompile as necessary.

(2) Create an interface between WinBUGS and your new model using WBDev.

(a) Navigate to "Program Files\BlackBox Component Builder 1.5\WBDev\Mod".

(b) Create a copy of TwoCptIndEff1ModelRK45.odc (or TwoCptIndEff1ModelLSODA.odc) within the same directory and rename the copy to the same name as the model component file you created.

(c) Drag and drop the new WBDev file (the file you just created) into BlackBox (or use File >> Open in the usual way).

(d) Replace all 4 instances of `TwoCptIndEff1ModelRK45` with the name of your new model component.

(e) Compile the new WBDev file.

(f) Navigate to "Program Files\BlackBox Component Builder 1.5\WBDev\Rsrc".

(g) Drag and drop Functions.odc into BlackBox.

(h) Duplicate the line
```
v <- "TwoCptIndEff1ModelRK45"(v,v,v,v,v,v,v,v,v) "WBDevTwoCptIndEff1ModelRK45.Install"
```

(i) Edit the copied line, replacing `TwoCptIndEff1ModelRK45` with the name of your new model component.

(3) Exit BlackBox. The new function will only be available after you restart BlackBox.

Example 3: General compartmental model. Example 3 involved simultaneous fitting of PK and response 2 data from Phases I and II. As before the PK was described by a two compartment with 1st order absorption. The response 2 data was described by an indirect effect model with drug effect on kout (inhibitory Emax). This example illustrates use of the library component for general compartmental models described by a system of ordinary differential equations. The BUGS language model is shown in Figure 9; the differential equations of the model are specified in the Component Pascal file TwoCptIndEff1ModelRK45.odc as shown in Figure 10. The BUGS model file (`indirectActionRK45Example.txt`) as well as an R script (`indirectActionRK45Example.R`) and data files (`fxa.data.csv` and `fxa.data2.csv`) implementing this example are contained in the BUGSModelLibrary directory.

Results. The MCMC history plots (Figure 11) indicate that the chains for some of the parameters, e.g., $\widehat{k}_{out}$, $\widehat{EC}_{50}$, $\widehat{\Delta k}_a$ and $\widehat{V}_1$, did not achieve convergence. Longer chains, reparameterization or other model revision should be considered. The model fit to the plasma concentration data was comparable to that in example 1 (not shown) in spite of this. The model also described the response 2 data reasonably well (Figure 12). Though the model fit is less than optimal, the example still successfully illustrates implementation of a nonlinear model described by user-specified ODE's.

TABLE 4. Example 3: Summary of the MCMC simulations of the marginal posterior distributions of the model parameters.

	Mean	SD	Naive SE	Time-series SE	2.50%	25%	50%	75%	97.50%	Effective N
deviance	-5000	98.7	2.33	5.59	-5130	-5060	-5020	-4950	-4740	56.5
CLHat	10.1	0.212	0.00499	0.0081	9.68	9.95	10.1	10.2	10.5	204
QHat	14.9	0.397	0.00935	0.0141	14.1	14.6	14.9	15.2	15.6	82.1
V1Hat	34.1	2.95	0.0694	NA	30.5	31.8	32.9	36.9	39.9	15.9
V2Hat	106	2.83	0.0667	0.13	100	104	106	108	111	88
DkaHat	1.86	0.235	0.00553	0.00506	1.58	1.69	1.76	2.07	2.33	20.7
koutHat	0.576	0.089	0.0021	NA	0.451	0.471	0.587	0.671	0.696	52.9
yeff0Hat	111	2.91	0.0686	0.0884	105	109	111	113	117	684
EC50Hat	0.323	0.0523	0.00123	0.00115	0.243	0.285	0.304	0.384	0.411	21.6
sigmaC	0.0992	0.00239	5.62E-05	0.000102	0.0948	0.0976	0.0991	0.101	0.104	303
sigmaYeff	0.157	0.00296	6.98E-05	0.000122	0.152	0.155	0.157	0.159	0.163	135
omega[1,1]	0.0729	0.00775	0.000183	0.000282	0.0592	0.0674	0.0725	0.0778	0.0898	378
omega[1,2]	0.0122	0.00676	0.000159	0.000263	-0.00148	0.00787	0.0123	0.0167	0.0256	121
omega[1,3]	-0.0103	0.0107	0.000252	0.000239	-0.0316	-0.0183	-0.00928	-0.00201	0.00773	181
omega[1,4]	-0.00863	0.00754	0.000178	0.000369	-0.0234	-0.0141	-0.00845	-0.0032	0.00539	117
omega[1,5]	-0.00365	0.00903	0.000213	NA	-0.0245	-0.00851	-0.00257	0.00254	0.0108	185
omega[2,2]	0.0559	0.0144	0.00034	NA	0.0332	0.0445	0.054	0.0654	0.0869	29.4
omega[2,3]	-0.00843	0.0146	0.000345	NA	-0.034	-0.0203	-0.00896	0.00399	0.0163	40.6
omega[2,4]	0.00915	0.00769	0.000181	0.000438	-0.00351	0.00364	0.00803	0.0141	0.0261	25.3
omega[2,5]	-0.00103	0.0136	0.00032	0.000444	-0.0299	-0.00992	-0.000248	0.00913	0.0222	18.2
omega[3,3]	0.0693	0.0226	0.000532	0.000471	0.0336	0.0469	0.0714	0.0872	0.108	49.8
omega[3,4]	0.0427	0.0095	0.000224	0.000498	0.0258	0.0364	0.0419	0.0481	0.0647	98.6
omega[3,5]	0.00544	0.0194	0.000456	0.00045	-0.0252	-0.0149	0.00909	0.0204	0.0385	43.4
omega[4,4]	0.0645	0.0117	0.000275	0.000639	0.0451	0.0562	0.0635	0.0718	0.09	21.9
omega[4,5]	-0.0177	0.0114	0.000268	0.00028	-0.0364	-0.0262	-0.0203	-0.00821	0.00456	182
omega[5,5]	0.0853	0.0212	5.00E-04	NA	0.0488	0.0713	0.0825	0.0977	0.132	30
omegaPD[1,1]	0.0584	0.0225	0.000529	0.000596	0.0206	0.0387	0.0594	0.0778	0.0959	42.9
omegaPD[1,2]	0.00172	0.0107	0.000251	NA	-0.025	-0.00346	0.00327	0.00901	0.0181	9.9
omegaPD[1,3]	0.0149	0.0388	0.000914	NA	-0.0655	-0.006	0.0172	0.0469	0.0759	17.4
omegaPD[2,2]	0.0866	0.00928	0.000219	0.000297	0.0703	0.0799	0.0863	0.0924	0.106	589
omegaPD[2,3]	0.0406	0.0149	0.000351	NA	0.0176	0.029	0.0382	0.0515	0.0704	74.1
omegaPD[3,3]	0.146	0.0439	0.00103	0.00201	0.0781	0.11	0.143	0.178	0.237	14.9

Additional examples. The BUGSModelLibrary distribution also contains two additional examples. oneCptExample.R and oneCptExample.txt are the R script and BUGS model files that attempt to fit a one compartment model to the same data use for Example 1. indirectActionLSODAExample.R and indirectActionLSODAExample repeat Example 3 using the LSODA ODE solver.

FIGURE 9. Example 3: BUGS language model for fitting a PKPD model consisting of a two compartment PK submodel and an indirect action PD submodel.

```
model
{

   for(i in 1:nsub){

    # Inter-patient variation
      logtheta[i, 1:5] ~ dmnorm(logthetaMean[i, 1:5], omega.inv[1:5, 1:5])
      logthetaMean[i, 1] <- logCLHat + 0.75*log(weight[start[i]]/70) # CL
      logthetaMean[i, 2] <- logQHat + 0.75*log(weight[start[i]]/70)  # Q
      logthetaMean[i, 3] <- logV1Hat + log(weight[start[i]]/70)         # V1
      logthetaMean[i, 4] <- logV2Hat + log(weight[start[i]]/70)         # V2
      logthetaMean[i, 5] <- logDkaHat                                     # ka - alpha1

      logtheta[i, 6:8] ~ dmnorm(logthetaMean[i, 6:8], omegaPD.inv[1:3, 1:3])
      logthetaMean[i, 6] <- logKoutHat                       # kout
      logthetaMean[i, 7] <- logYeff0Hat                      # yeff0
      logthetaMean[i, 8] <- logEC50Hat                     # EC50

      theta[i,9] <- 1 # F1
      theta[i,10] <- 1 # F2
      theta[i,11] <- 1 # F3
      theta[i,12] <- 1 # F4
      theta[i,13] <- 0 # tlag1
      theta[i,14] <- 0 # tlag2
      theta[i,15] <- 0 # tlag3
      theta[i,16] <- 0 # tlag4

      for(j in 1:8){
         log(theta[i,j]) <- logtheta[i,j]
         log(thetaPred[i,j]) <- logthetaPred[i,j]
      }

      # Call to PK model library to calculate amount in each compartment at each time
      xhat[start[i]:end[i],1:4] <- TwoCptIndEff1ModelRK45(time[start[i]:end[i]],
         amt[start[i]:end[i]], rate[start[i]:end[i]], ii[start[i]:end[i]], evid[start[i]:end[i]],
         cmt[start[i]:end[i]], addl[start[i]:end[i]], ss[start[i]:end[i]], theta[i,])

   }

   for(i in 1:nobs){

      logCobs[i] ~ dnorm(logCHat[i],tauC)
      CHat[i] <- 1000*xhat[i,2]/theta[subject[i],3]
      logCHat[i] <- log(max(CHat[i],eps))

      logYeff[i] ~ dnorm(logYeffHat[i],tauYeff)
      YeffHat[i] <- 1000*(xhat[i,4] + theta[subject[i],7])
      logYeffHat[i] <- log(max(YeffHat[i],eps))

   }

   logCLHat ~ dnorm(0,1.0E-6)
   logQHat ~ dnorm(0,1.0E-6)
   logV1Hat ~ dnorm(0,1.0E-6)
   logV2Hat ~ dnorm(0,1.0E-6)
   logDkaHat ~ dnorm(0,1.0E-6)
   log(CLHat) <- logCLHat
   log(QHat) <- logQHat
   log(V1Hat) <- logV1Hat
   log(V2Hat) <- logV2Hat
   log(DkaHat) <- logDkaHat

   logKoutHat ~ dnorm(0,1.0E-6)
   logYeff0Hat ~ dnorm(0,1.0E-6)
   logEC50Hat ~ dnorm(0,1.0E-6)
   log(koutHat) <- logKoutHat
   log(yeff0Hat) <- logYeff0Hat
   log(EC50Hat) <- logEC50Hat
   tauC <- 1/(sigmaC*sigmaC)
   sigmaC ~ dunif(0,1000)
   tauYeff <- 1/(sigmaYeff*sigmaYeff)
   sigmaYeff ~ dunif(0,1000)
   omega.inv[1:5, 1:5] ~ dwish(omega.inv.prior[1:5, 1:5], 5)
   omega[1:5, 1:5] <- inverse(omega.inv[1:5, 1:5])
   omegaPD.inv[1:3, 1:3] ~ dwish(omegaPD.inv.prior[1:3, 1:3], 3)
   omegaPD[1:3, 1:3] <- inverse(omegaPD.inv[1:3, 1:3])
   eps <- 1.0E-6

}
```

FIGURE 10. Example 3: Specification of differential equations in the Component Pascal file TwoCptIndEff1ModelRK45.odc.

```
PROCEDURE UserDerivatives(IN theta, x: ARRAY OF REAL;
numEq: INTEGER; t: REAL; OUT dxdt: ARRAY OF REAL) ;
VAR
CL, Q, V2, V3, dka, kout, yeff0, EC50, ka, k10, k12, k21, ksum, conc: REAL;
BEGIN
CL := theta[0];
Q := theta[1];
V2 := theta[2];
V3 := theta[3];
dka := theta[4];
kout := theta[5];
yeff0 := theta[6];
EC50 := theta[7];
k10 := CL/V2;
k12 := Q/V2;
k21 := Q/V3;
ksum := k10 + k12 + k21;
ka := dka + (ksum - Math.Sqrt(ksum*ksum-4.0*k10*k21))/2.0;

(* Differential equations for the model excluding piecewise *)
(* constant  input rates provided in the data set *)

dxdt[0] := -ka * x[0];
dxdt[1] := ka * x[0] - (k10 + k12) * x[1] + k21 * x[2];
dxdt[2] := k12 * x[1] - k21 * x[2];
conc := x[1]/V2;
dxdt[3] := kout * (yeff0 - (1 - conc/(EC50+conc))*(x[3]+yeff0))
(* x[3] = yeff - yeff0 *)

END UserDerivatives;
```

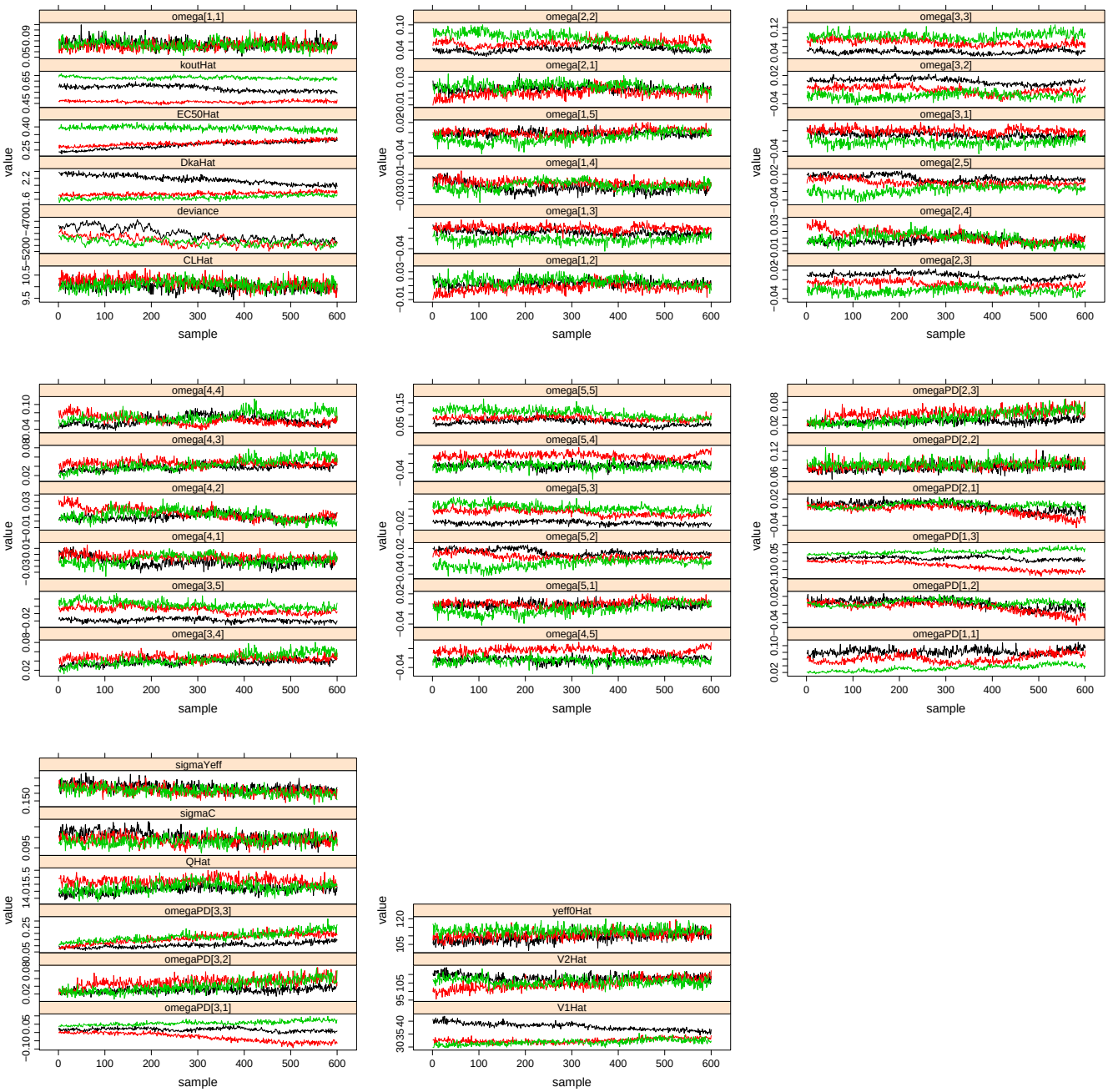

FIGURE 11. Example 3: MCMC history plots for the parameters of a PKPD model where the PK is described by a two compartment model with first order absorption and the PD by an indirect effect model with a drug effect on k_{out} (inhibitory Emax).

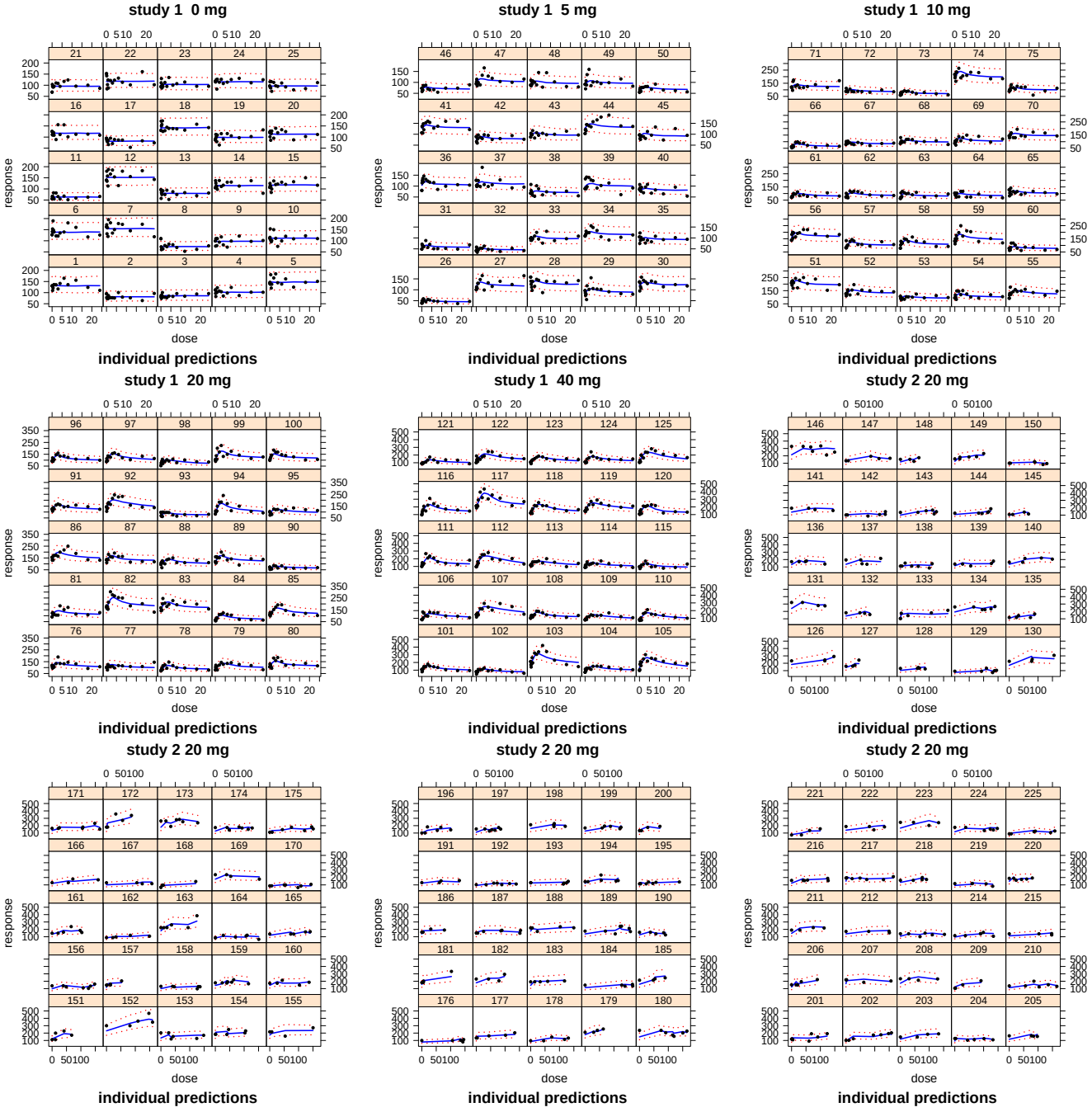

FIGURE 12. Example 3: Predicted (posterior median and 90% credible intervals) and observed response 1 measurements.