

Interactive population PK/PD Simulations in R



Tarjinder Sahota¹, Stefano Zamuner¹

¹Clinical Pharmacology Modelling and Simulation, GSK, London, UK

Introduction

Simulation of population PK/PD models enables untested hypotheses and intervention strategies to be explored quantitatively. In strategic drug development meetings, fast production of graphical outputs enables real time evidence based scenario planning where the breadth of clinical team expertise can be leveraged. Traditionally this has been difficult due to limitations in the flexibility and speed of simulation software.

Objectives

The desired workflow should consist of the following:

- Access to a fast and reliable differential equation solver
- Ability to specify mixed effects models
- Ability to specify complex dosing inputs
- Ability to create custom graphical output with an intuitive interface
- User-friendly and fast.

The R language has historically not been used for such a purpose. However, recent package development in R has provided users with the ability to access existing DE solver libraries, and perform pre and post processing quickly and easily. RStudio, an integrated development environment (IDE) for R includes a user friendly package for interactivity. The objective here is to show illustrate the use of these packages in RStudio for interactive PK/PD simulations.

Methods

The following R packages were installed on a standard laptop (Intel Core i5 – 3427U @ 1.8 Ghz, 4GB RAM): deSolve [1], manipulate (installed with Rstudio), plyr, and ggplot2. The “deSolve” package provides an interface to the Fortran LSODA function for solving systems of ODEs. The option to pre-compile enables fast solution of ODEs and an interface is provided for bolus and zero order input into compartments. “plyr” and “ggplot2” provided user friendly data manipulation and plotting capabilities. The “manipulate” package provides a simple function to add sliders, checkboxes and menus to plots in Rstudio. We show example interactive output with full source code.

Results

The code below shows the entire script for creating the interactive simulation. The simulation has the flexibility to explore a variety of dosing administrations and covariates. The model is based on the drug Adalimumab. Shaded areas depict 95% prediction intervals. We have used and presented ordinary and delayed differential equation models in internal project team meeting. We have not tested the functionality for partial differential equations.

Speed

The displayed code is without compilation (n=50) and takes 13 seconds to run.

Uncompiled run time for n=100 = 26.48 seconds.

Compiled (C++) run time for n=100 = 4.03 seconds.

For easy instuncions on how to compile your model, go to:

<http://cran.r-project.org/web/packages/deSolve/vignettes/compiledCode.pdf>

Flexibility

The flexibility of the R language has provided virtually unlimited flexibility in variance specification. Specification of ODEs in C++ for compilation, drastically reduced run time by more than 10 fold. A variety of model specifications were implemented.

The manipulate library allows for sliders, checkboxes and drop down menus to be created for continuous, binary, categorical variables respectively.

Accuracy

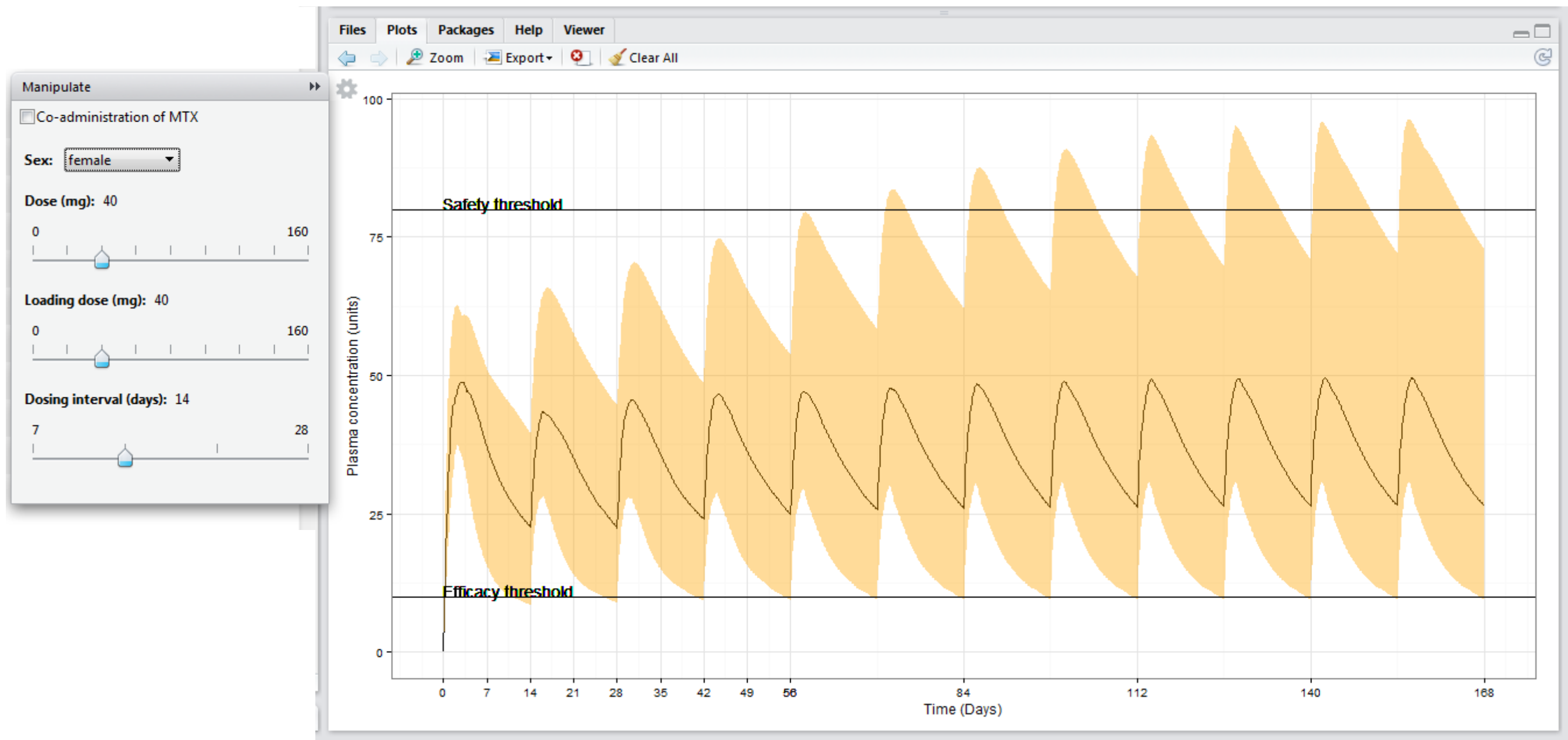
The default ode solver used is the LSODA FORTRAN library. This is the same ODE solver used by NONMEMS ADVAN13.

Conclusions

The proposed set of R packages (deSolve and manipulate) in conjunction with Rstudio enables the production of high quality, interactive graphical outputs from PK/PD simulations.

References

[1] Soetaert, K., et al. (2010). Solving Differential Equations in R: Package deSolve Journal of Statistical Software, 33(9), 1--25.



Code

```
1 library(manipulate) ## for interactivity : Requires Rstudio
2 library(deSolve)   ## DE solver
3 library(ggplot2)   ## for plotting
4 library(MASS)      ## for mvnrm function : simulation from multivariate normal
5 library(plyr)      ## for the ldply and ddply aggregation functions
6
7 #####
8
9 ## wrap simulation and plotting in function for later
10 ## define interactive variables as arguments
11 plot.fun <- function(dose,loading.dose=20,tau=14,mtx=0,sex="female"){
12
13   ## Define dosing events longitudinally (like in nonmem)
14   dose.events <- data.frame(var="depot",time=seq(0,7*4*6,by=tau),value=dose,method="add")
15
16   ## add loading dose
17   dose.events$value[1] <- dose.events$value[1] + loading.dose
18
19   ## initial values in compartments
20   yini <- c(depot = 0, blood = 0, periph = 0)
21
22   ## observation times
23   times <- seq(0,7*4*6,3/24)
24
25   ## parameter values
26   n.ind <- 50
27   TVCL <- 240
28   if(sex=="male") TVCL <- TVCL*1.2
29   if(mtx==1) TVCL <- TVCL*1.3
30   TVV2 <- 3100; TVV3 <- 2900; TVQ <- 336; TVKA <- 0.75
31   ## log parameter values assumed to be multivariate normal
32   params <- mvnrm(n.ind, mu = log(c(TVCL,TVV2,TVV3,TVQ,TVKA)),
33                   Sigma = diag(c(0.1, 0.1,0.1, 0.1, 0.1)))
34   params <- exp(params)
35
36   ## Define function for ODEs
37   pkeq <- function(t, y, p) {
38     CL <- p[1]; V2 <- p[2]; V3 <- p[3]; Q <- p[4]; KA <- p[5]
39
40     dy1 <- - KA*y[1]
41     dy2 <- KA*y[1] - (CL/V2)*y[2] - (Q/V2)*y[2] + (Q/V3)*y[3]
42     dy3 <- (Q/V2)*y[2] - (Q/V3)*y[3]
43     list(c(dy1, dy2, dy3))
44   }
45
46   ## loop through matrix row by row (i.e. by ID) output data.frame as you go
47   out <- ldply(1:nrow(params),function(i){
48     outi <- ode(func = pkeq, times = times,
49               y = yini, parms = params[i,],events=list(data=dose.events))
50
51     outi <- as.data.frame(outi)
52     outi$SIM <- i
53     outi
54   })
55
56   ## create any plot you want, below I do a vpc type plot.
57   ds <- ddply(out,"time",summarise,
58             low = quantile(blood,probs=0.025),
59             mid = quantile(blood,probs=0.5),
60             hi = quantile(blood,probs=0.975))
61
62   ## plot the results
63   xbreaks <- c(seq(0,56,by=7),seq(56,168,by=7*4))
64   ggplot(ds,aes(x=time,y=mid)) + geom_line() + theme_bw() +
65     geom_ribbon(aes(ymin=low,ymax=hi),fill="orange",alpha=0.4) +
66     geom_hline(y=c(80)) + geom_text(y=c(80),x=0,vjust=0,hjust=0,label="Safety threshold") +
67     geom_hline(y=c(10)) + geom_text(y=c(10),x=0,vjust=0,hjust=0,label="Efficacy threshold")+
68     scale_x_continuous("Time (Days)",breaks=xbreaks) +
69     scale_y_continuous("Plasma concentration (units)")
70
71 }
72
73 ## End of function definition
74 #####
75
76 ## Interactive plot
77 manipulate(plot.fun(dose=dose,loading.dose=loading.dose,tau=tau,sex=sex),
78   mtx=checkbox(FALSE,"Co-administration of MTX"),
79   sex=picker("male","female",label="Sex"),
80   dose=slider(0,160,initial=40,step=20,label="Dose (mg)"),
81   loading.dose=slider(0,160,initial=40,step=20,label="Loading dose (mg)"),
82   tau=slider(7,28,initial=14,step=7,label="Dosing interval (days)"))
```