Supplementary Materials for:

Performance of the SAEM and FOCEI algorithms in the open-source non-linear mixed effect modelling tool nlmixr

Rik Schoemaker1, Matthew Fidler2, Christian Laveille3, Justin J. Wilkins1, Richard Hooijmaijers4, Teun M. Post4, Mirjam N. Trame5, Yuan Xiong6, and Wenping Wang7

*1Occams Coöperatie UA, Amstelveen, The Netherlands*

*2Novartis Pharmaceuticals Corporation, Fort Worth, Texas, USA*

*3Calvagone, Liergues, France*

*4LAP&P Consultants BV, Leiden, The Netherlands*

*5Novartis Institutes for BioMedical Research, Cambridge, Massachusetts, USA*

*6Certara, Princeton, New Jersey, USA*

*7Novartis Pharmaceuticals Corporation, East Hanover, New Jersey, USA*

Address for correspondence:

Rik Schoemaker

Occams Coöperatie UA

Malandolaan 10

1187HE Amstelveen

[rik.schoemaker@occams.com](mailto:rik.schoemaker@occams.com)

tel: +31 20 441 6410

Table of Contents

[Appendix A Estimation and calculation of standard errors 2](#_Toc19262196)

[A.1 Estimation of SE of population typical parameters for the sparse-sampling estimates 2](#_Toc19262197)

[A.2 Calculation of the covariance matrix in Monolix and nlmixr for SAEM 2](#_Toc19262198)

[Appendix B Documentation for multiple models with rich data 3](#_Toc19262199)

[Appendix C Additional data-rich multiple-model graphs 5](#_Toc19262200)

[Appendix D Installation of nlmixr on Windows 11](#_Toc19262201)

[Appendix E Code and methodology for running sparse data analysis 13](#_Toc19262202)

[Appendix F Code and methodology for running richly sampled data and multiple models 19](#_Toc19262203)

##### Estimation and calculation of standard errors

###### Estimation of SE of population typical parameters for the sparse-sampling estimates

By calculating the standard deviation of the 500 population typical estimates that were obtained by analysing the 500 sampled datasets, a bootstrap-type estimate of the standard error of these population typical estimates is obtained. The values listed under NONMEM/FOCEI in Table S-1 were used as reference values in the graphs in the publication

Table S1 Bootstrapped SE estimates for the three population typical parameters using the results from the four algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| Analysis | SE\_CL | SE\_V | SE\_KA |
| Monolix/SAEM | 0.0344 | 0.0388 | 0.1101 |
| nlmixr/SAEM | 0.0334 | 0.0381 | 0.0986 |
| NONMEM/FOCEI | 0.0320 | 0.0375 | 0.0955 |
| nlmixr/FOCEI | 0.0322 | 0.0379 | 0.0952 |

###### Calculation of the covariance matrix in Monolix and nlmixr for SAEM

The covariance matrix of the fixed effects in Monolix, saemix & nlmixr comes from the classic result that the covariance matrix of parameters in linear regression is . For weighted LS, this result becomes with W as the weight. For nonlinear regression, the covariance matrix of parameters is with . For nonlinear mixed-effect models, the problem reduces to a weighted nonlinear regression with coefficient matrix and weight with Z a submatrix of X corresponding to the parameters with random effects.

The SAEM implementation in NONMEM provides an alternative calculation of standard errors and an objective function value. The NONMEM suggestion is to follow SAEM with IMP to obtain better SEs and an appropriate OFV. As described above, Monolix uses a different method to calculate the standard errors, and does not use a IMP step after SAEM, and in nlmixr, the calculation of SEs is performed fully in line with Monolix. The IMP step to follow SAEM is NONMEM-specific, and is not required to obtain adequate Ses in either Monolic or nlmixr.

##### Documentation for multiple models with rich data

A set of 36 pharmacokinetic control files was generated; different types of pharmacokinetic models were simulated from 1 or 2-compartment systems with either linear or Michaelis-Menten elimination, and input using a bolus, a 1 hour infusion, or an oral administration with first-order absorption. The simulations include different dosing regimens: single dose over 72 hours, multiple doses with a full profile on Day 7 and Day 10-13, and the combination of the 72 hour profile and the subsequent multiple doses. For each data set, single dose and multiple doses were simulated. The single dose was administered at time 0 and observations were made up to 72h. Then multiple daily doses started at time 72h and a dose was given every day during 10 days. Trough measurements were made every day. A full profile was drawn at Day 7 and at Day 10-13.

The sampling scheme was:

* Single dose at time=0h, 0.25h, 0.5h, 0.75h, 1h, 2h, 4h, 8h, 12h, 16h, 20h, 24h, 36h, 48h, 60h, and 72h
* Multiple doses at time=72h, 96h, 120h, 144h, 168h, 192h, and 216h:
  + Trough measurements: 96h, 120, 144h, 192h, and 216h
  + Full profile at Day 7: 144.25h, 144.5h, 144.75h, 145h, 145.5h, 146h, 146.5h, 147h, 148h, 150h, 152h, 156h, 160h, 164h, and 168h
  + Full profile at Day 10-13: 216.25h, 216.5h, 216.75h, 217h, 217.5h, 218h, 218.5h, 219h, 220h, 222h, 224h, 228h, 232h, 236h, 240h, 252h, 264h, 276h, and 288h

Depending of the type of models 4 different doses were simulated (e.g. 1x, 3x, 6x and 12x or 1x, 2x, 4x and 8x), and 120 subjects were used for each data set with 30 subjects per group.

All the random effects were simulated with a log-normal distribution. The inter-individual variability was set to 30% whatever the parameter and a residual error of 20% was used. Simulated concentrations below the value of 0.1 ng/mL were considered below the limit of quantification and were not taking into account in the analyses. All models analysed with NONMEM applied FOCE with interaction using ADVAN13 with settings NSIG=2 SIGL=6 NOABORT POSTHOC METHOD=COND INTER NOOBT, and TOL=6. For runs that did not converge with TOL=6, TOL was increased to 7 or 8. The parameters used for the simulation were the following, selected for the appropriate model:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | CL  (L/h) | Vc  (L) | Q  (L/h) | Vp  (L) | Vmax  (mg/h) | Km  (mg/L) | ka  (1/h) |
| THETA | 4.00 | 70.0 | 4.00 | 50.0 | 1000 | 250 | 1.00 |

Four arms of 30 subjects were simulated with administered doses of 10, 30, 60 and 120 mg for most models; the models with oral and bolus administration with Michalis-Menten elimination had doses of 10, 20, 40, and 80 mg administered.

* IV bolus administration with one compartment, linear elimination
  + runN001 Single dose, Vc and CL
  + runN002 Multiple dose, Vc and CL
  + runN004 Single and multiple doses, Vc and CL
* IV bolus administration with one compartment, non-linear elimination
  + runN009 Single dose, Vc, Vmax and Km
  + runN010 Multiple dose, Vc, Vmax and Km
  + runN011 Single and multiple doses, Vc, Vmax and Km
* IV infusion administration (1 hour) with one compartment, linear elimination
  + runN012 Single dose, Vc and CL
  + runN013 Multiple dose, Vc and CL
  + runN015 Single and multiple doses, Vc and CL
* IV infusion administration (1 hour) with one compartment, non-linear elimination
  + runN020 Single dose, Vc, Vmax and Km
  + runN021 Multiple dose, Vc, Vmax and Km
  + runN022 Single and multiple doses, Vc, Vmax and Km
* Oral administration with one compartment, linear elimination and first-order absorption
  + runN023 Single dose, ka, Vc and CL
  + runN024 Multiple dose, ka, Vc and CL
  + runN026 Single and multiple doses, ka, Vc and CL
* Oral administration with one compartment, non-linear elimination and first-order absorption
  + runN029 Single dose, ka, Vc, Vmax and Km
  + runN030 Multiple dose, ka, Vc, Vmax and Km
  + runN031 Single and multiple doses, ka, Vc, Vmax and Km
* IV bolus administration with two compartments, linear elimination
  + runN032 Single dose, Vc, Vp, CL and Q
  + runN033 Multiple dose, Vc, Vp, CL and Q
  + runN035 Single and multiple doses, Vc, Vp, CL and Q
* IV bolus administration with two compartments, non-linear elimination
  + runN040 Single dose, Vc, Vp, Q, Vmax and Km
  + runN041 Multiple dose, Vc, Vp, Q, Vmax and Km
  + runN042 Single and multiple doses, Vc, Vp, Q, Vmax and Km
* IV infusion administration (1 hour) with two compartments, linear elimination
  + runN046 Single dose, Vc, Vp, CL and Q
  + runN047 Multiple dose, Vc, Vp, CL and Q
  + runN049 Single and multiple doses, Vc, Vp, CL and Q
* IV infusion administration (1 hour) with two compartments, non-linear elimination
  + runN054 Single dose, Vc, Vp, Q, Vmax and Km
  + runN055 Multiple dose, Vc, Vp, Q, Vmax and Km
  + runN056 Single and multiple doses, Vc, Vp, Q, Vmax and Km
* Oral administration with two compartments, linear elimination and first-order absorption
  + runN060 Single dose, ka, Vc, Vp, CL and Q
  + runN061 Multiple dose, ka, Vc, Vp, CL and Q
  + runN063 Single and multiple doses, ka, Vc, Vp, CL and Q
* Oral administration with two compartments, non-linear elimination and first-order absorption
  + runN068 Single dose, ka, Vc, Vp, Q, Vmax and Km
  + runN069 Multiple dose, ka, Vc, K23, K32, Vmax and Km
  + runN070 Single and multiple doses, ka, Vc, K23, K32, Vmax and Km

##### Additional data-rich multiple-model graphs

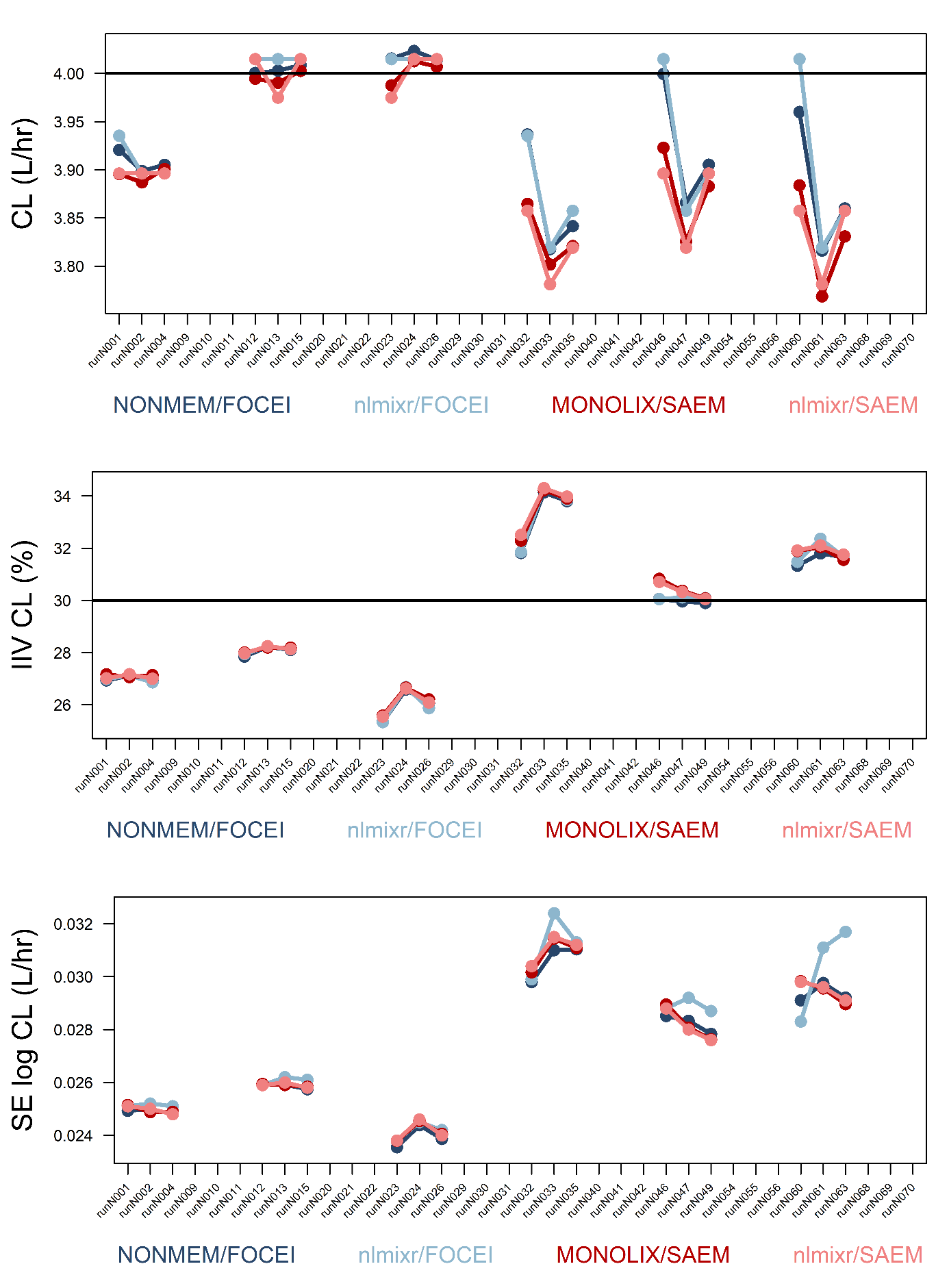


Figure S1 Results for CL

Theta (top), IIV (middle), SE (bottom)

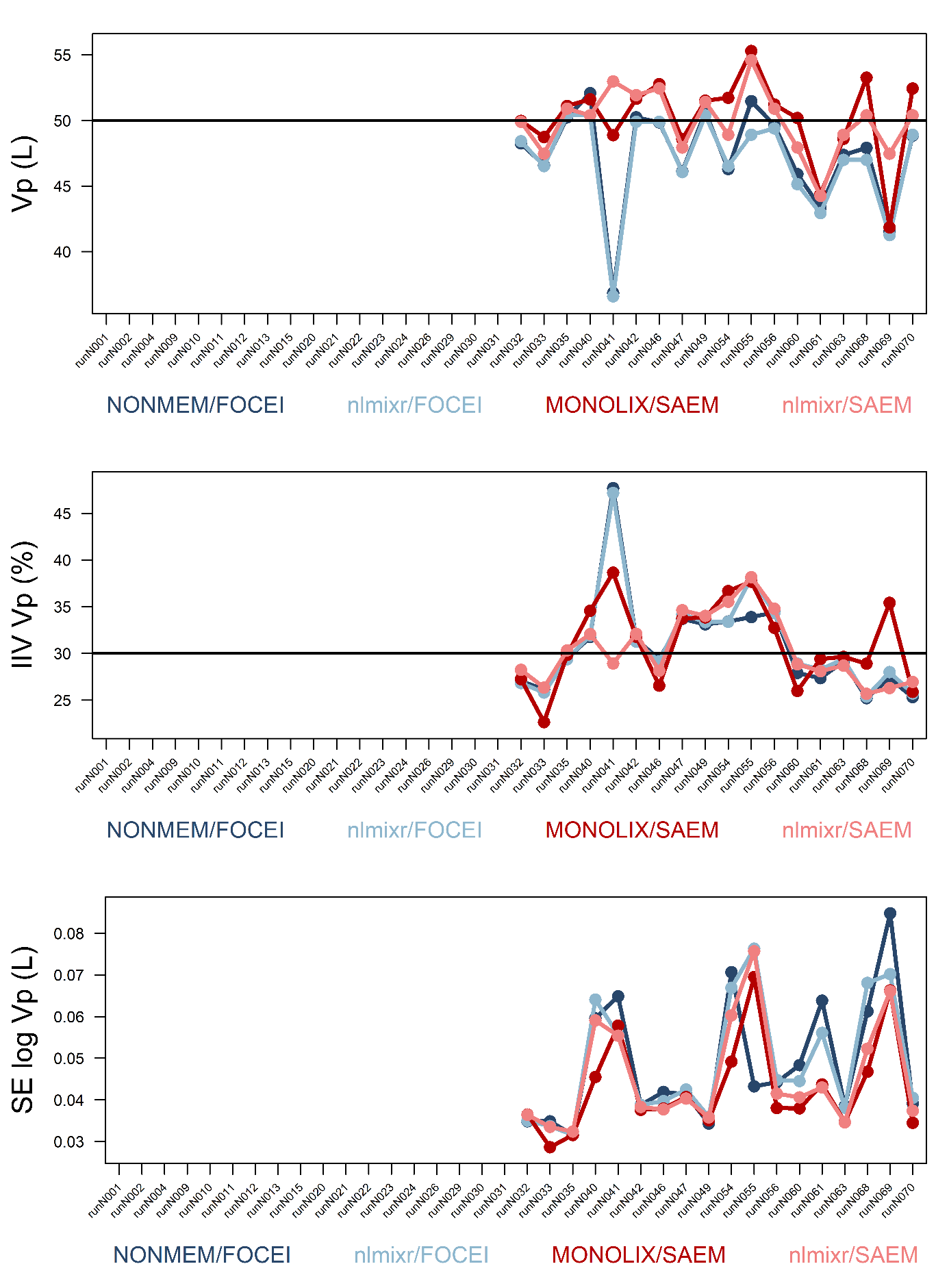


Figure S2 Results for Vp

Theta (top), IIV (middle), SE (bottom)

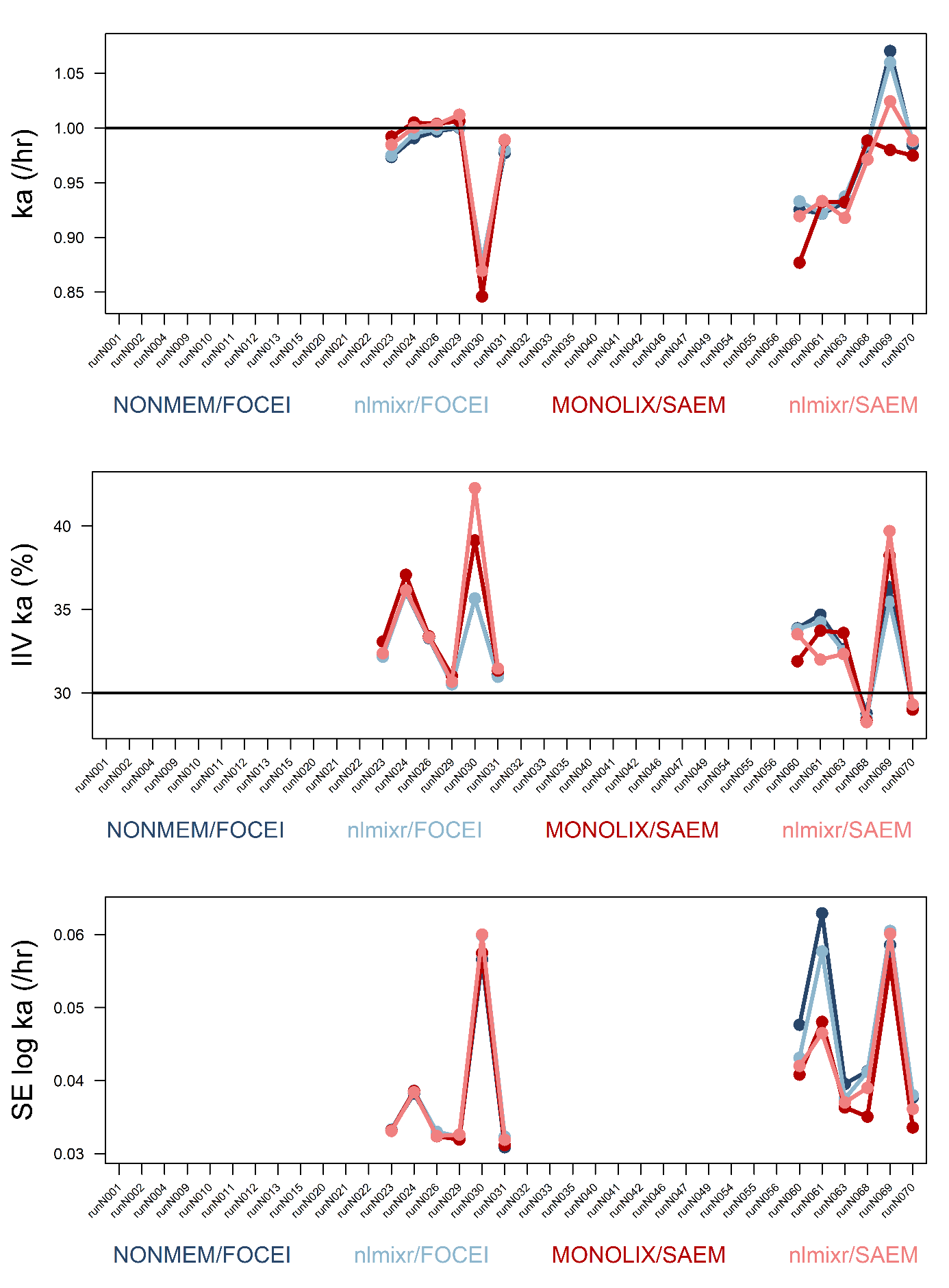


Figure S3 Results for ka

Theta (top), IIV (middle), SE (bottom)

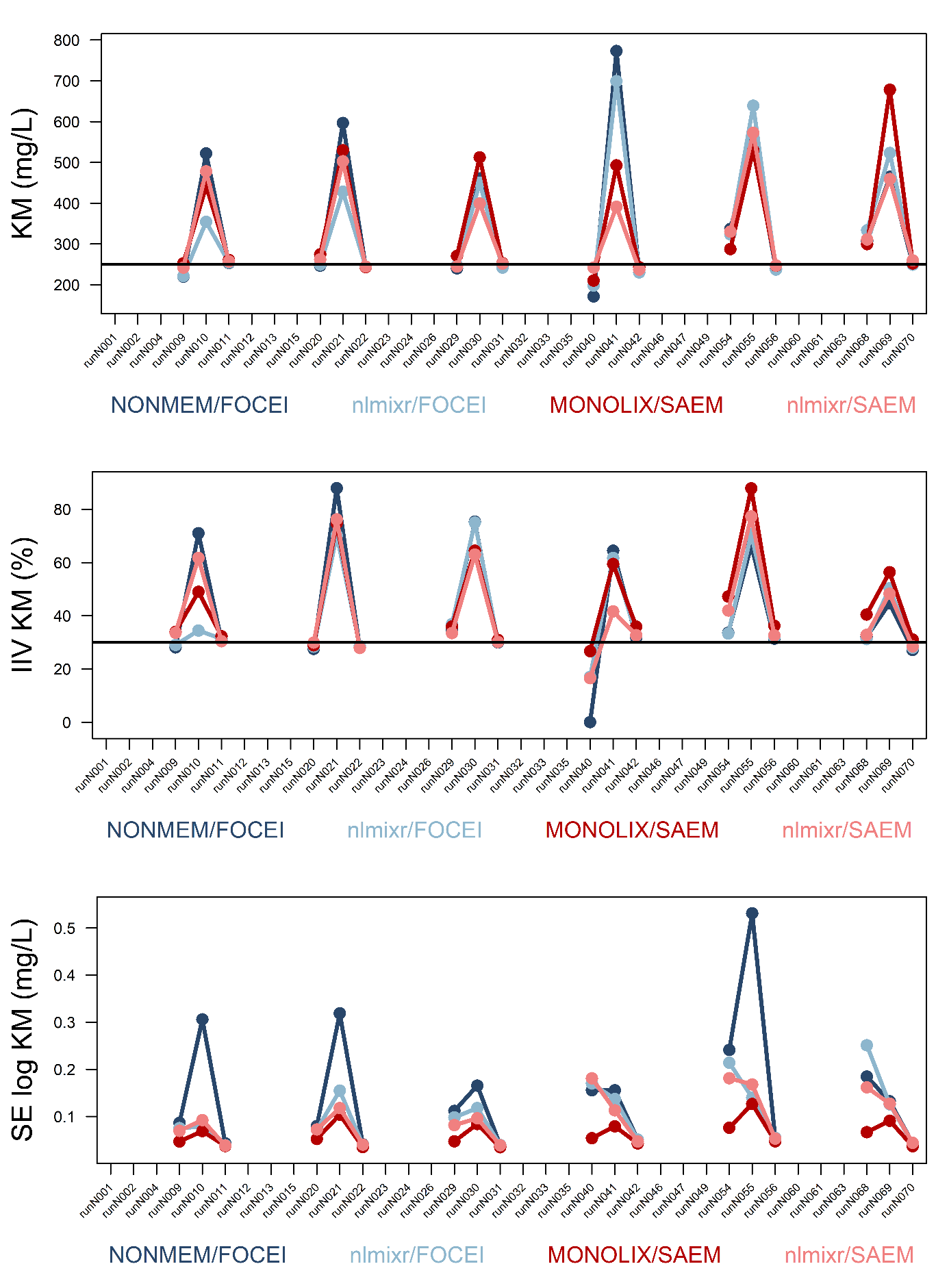


Figure S4 Results for KM

Theta (top), IIV (middle), SE (bottom)

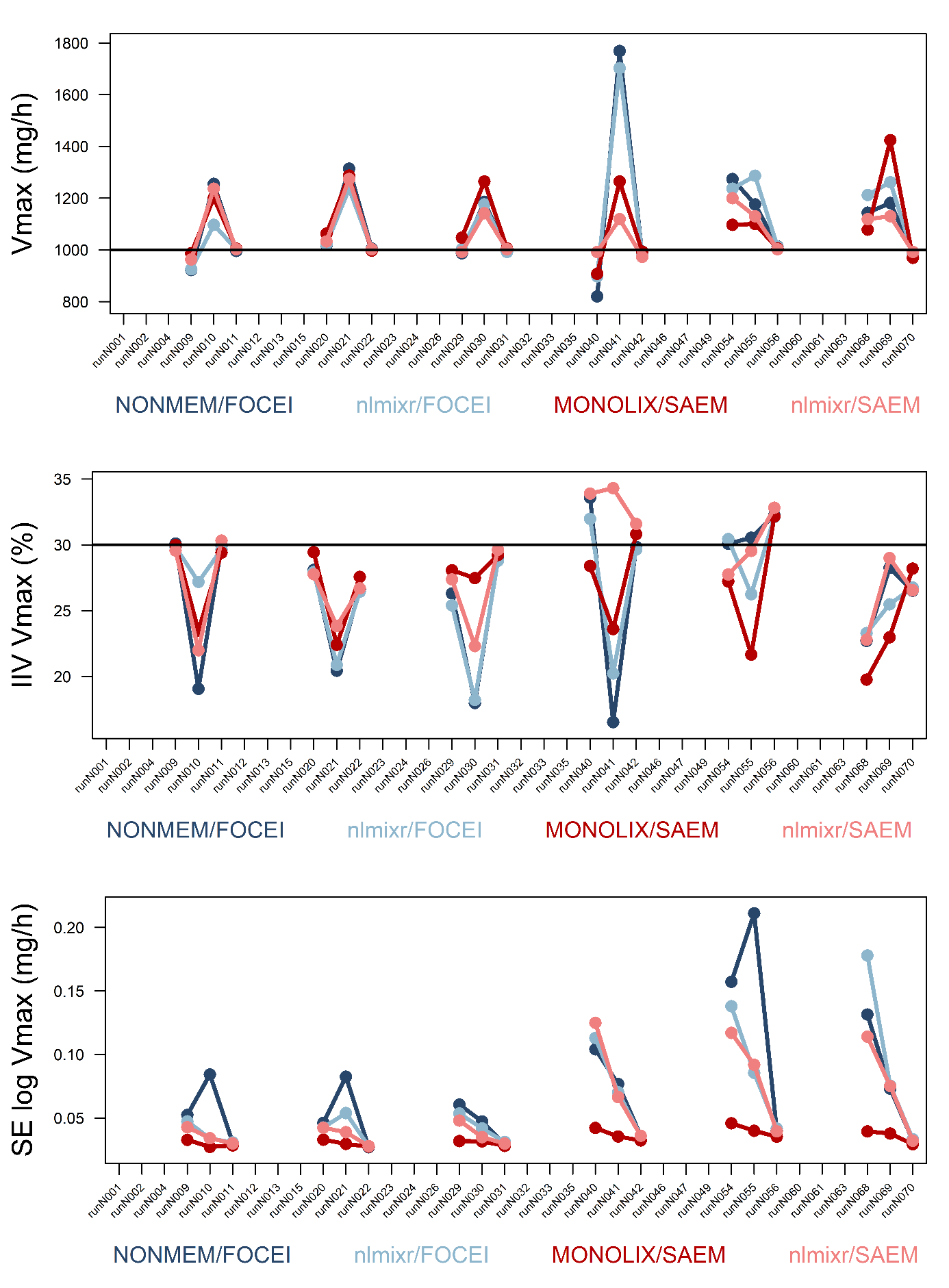


Figure S5 Results for Vmax

heta (top), IIV (middle), SE (bottom)

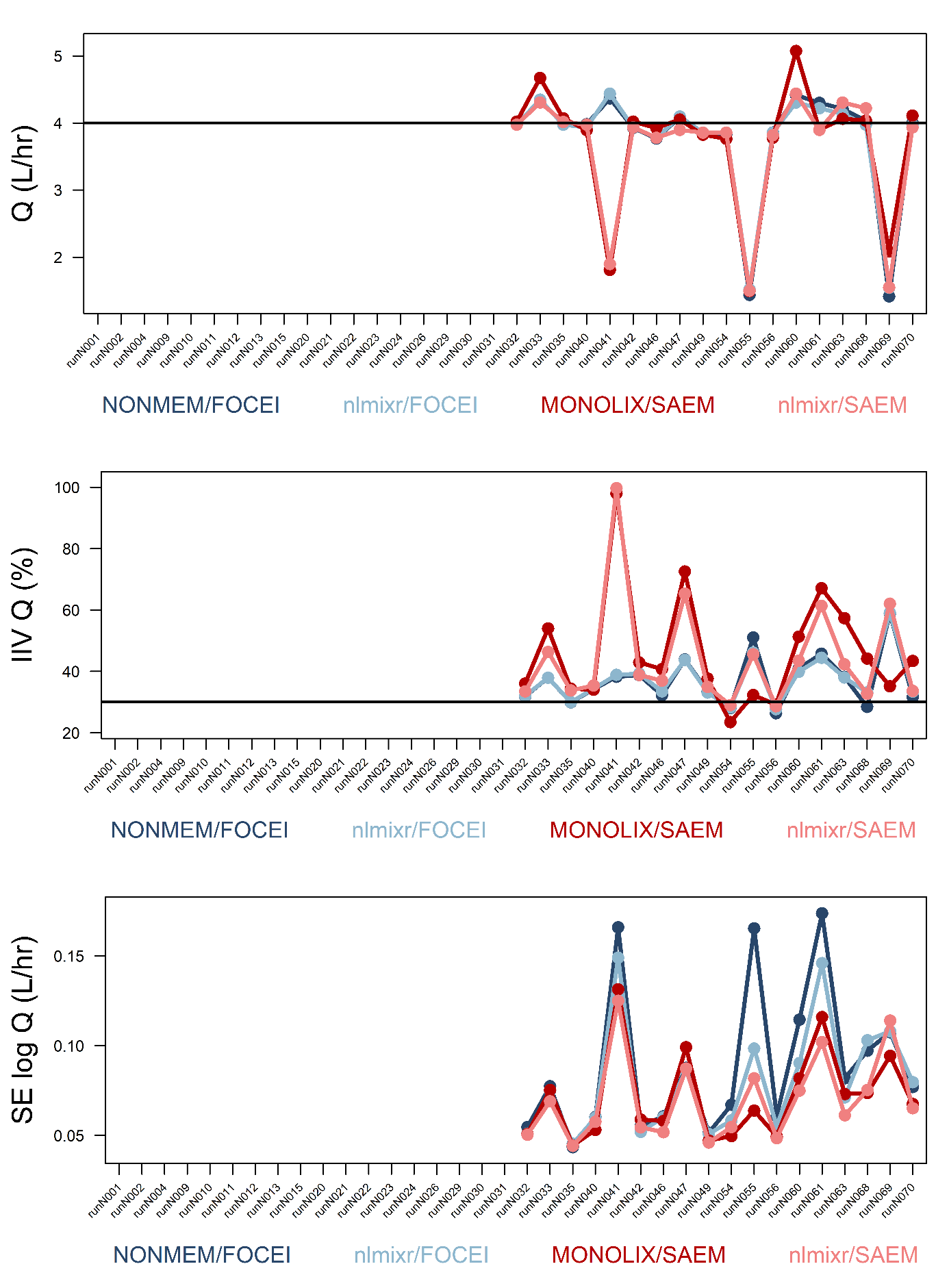
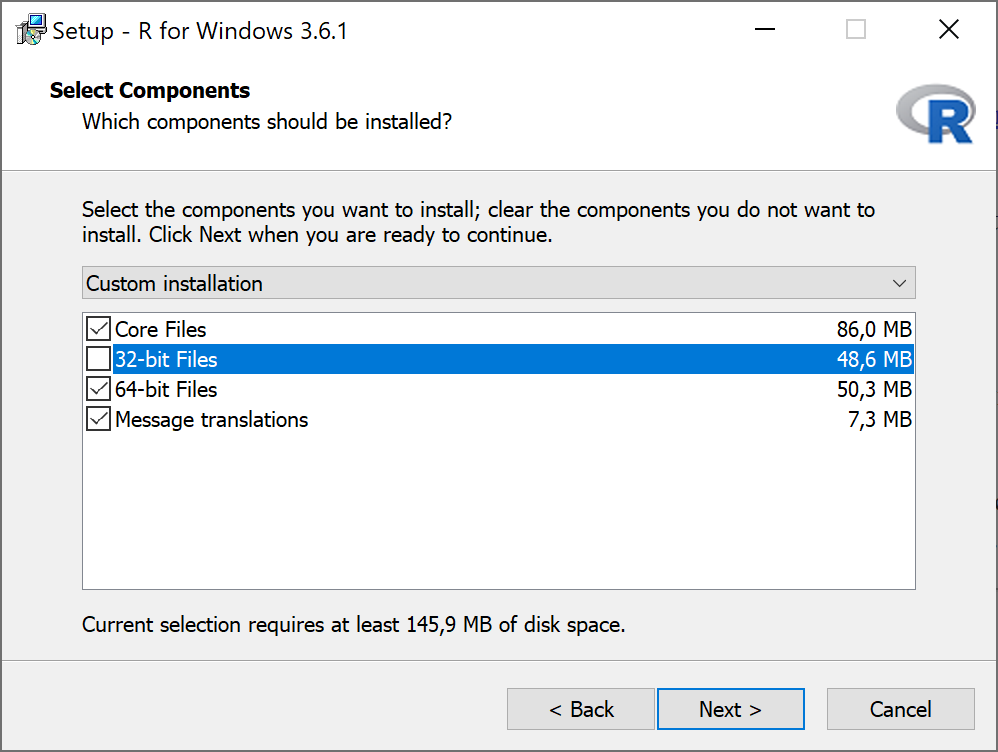


Figure S6 Results for Q

Theta (top), IIV (middle), SE (bottom)

##### Installation of nlmixr on Windows

Install R first by running R-3.6.1-win.exe from CRAN (<https://cran.r-project.org/bin/windows/base/R-3.6.1-win.exe>). Be sure to exclude the 32-bit files:



Prior to installing nlmixr, both Rtools and Python need to be installed. Uninstall any existing version of Rtools. Rtools35.exe can be downloaded from <http://cran.r-project.org/bin/windows/Rtools/>. Install by double-clicking the file and following the prompts. Remove the 32 bit toolchain option. During the installation process, in the “Select additional tasks” window, check the box for ‘Add rtools to the system path’, and make sure that the path contains the following items:

C:\Rtools\bin;  
C:\Rtools\mingw\_64\bin;  
C:\Program Files\R\R-3.6.1\bin\x64\;

add items that are not already there, and update if necessary.

Python is required for for additional computations in RxODE and nlmixr. Download the latest 64-bit version (3.7.3 at the time of writing) from <https://www.python.org/downloads/windows/>, scroll down to select the ‘Windows x86-64 executable installer’ and run the installer. Be sure to check-mark ‘Add Python 3.7 to PATH, and then use the ’Install Now’ option to have Python reside in users/*UserName*/AppData to avoid write permissions needed during additional setup.

These analyses were performed using the RxODE version (0.9.1-3 of 6 August 2019) and nlmixr version (1.1.1-1 of 23 August 2019) from CRAN that can be installed using:

install.packages("nlmixr")

followed by running:

library(RxODE)  
rxWinPythonSetup()

Alternatively, the latest version of nlmixr from GitHub can be installed with the following code:

library(devtools)  
install\_github("nlmixrdevelopment/RxODE")  
install\_github("nlmixrdevelopment/nlmixr")

##### Code and methodology for running sparse data analysis

The nlmixr/SAEM, and nlmixr/FOCEI parameter estimation algorithms were compared with Monolix/SAEM and NONMEM/FOCEI in a sparse-sampling data setting. To this end, 10,000 patients were simulated after a single dose with doses split between 10, 30, 60 and 120 mg. Four time points were randomly sampled in the 24 hours after the dose. A first order absorption, one compartment distribution, and linear elimination model was used with population values of Clearance=4.0 L/hr, Vc=70 L, and ka=1 /hr, 30% IIV for all three parameters (diagonal omega matrix), and 20% residual variability. Of these 10,000 patients, 120 patients were randomly sampled (stratified by dose, 30 subjects per dose), 500 times using PsN and analysed using NONMEM/FOCEI, Monolix/SAEM, nlmixr/SAEM, and nlmixr/FOCEI.

The code for the full analysis is provided here along with some output graphs to demonstrate the results. While these are interesting in their own right, the code also demonstrates how to perform parallel analysis of the estimations; with 500 datasets per analysis, it is extremely useful to be able to run these analyses side-by-side provided one has access to a computer with multiple cores. As these approaches are OS-dependent, the code below for running in parallel is only applicable to Windows.

Load the packages and define the simulation model using ODEs:

library(nlmixr)  
library(data.table)  
  
#Define the RxODE model  
 ode1 <- "  
 d/dt(abs) = -KA\*abs;  
 d/dt(centr) = KA\*abs-(CL/V)\*centr;  
 C2=centr/V;  
 "  
   
#Create the RxODE simulation object  
 mod1 <- RxODE(model = ode1)

Generate the 10,000 sampled parameters:

#Population parameter values on log-scale  
 paramsl <- c(CL = log(4),  
 V = log(70),  
 KA = log(1))  
#make 10,000 subjects to sample from:  
 nsubg <- 2500 # subjects per dose  
 doses <- c(10, 30, 60, 120)  
 nsub <- nsubg \* length(doses)  
#IIV of 30% for each parameter  
 omega <- diag(c(0.09, 0.09, 0.09))# IIV covariance matrix  
 sigma <- 0.2  
#Sample from the multivariate normal  
 set.seed(98176247)  
 library(MASS)  
 mv <-  
 mvrnorm(nsub, rep(0, dim(omega)[1]), omega) # Sample from covariance matrix  
#Combine population parameters with IIV  
 params.all <-  
 data.table(  
 "ID" = seq(1:nsub),  
 "CL" = exp(paramsl['CL'] + mv[, 1]),  
 "V" = exp(paramsl['V'] + mv[, 2]),  
 "KA" = exp(paramsl['KA'] + mv[, 3])  
 )  
#set the doses (looping through the 4 doses)  
 params.all[, AMT := rep(1000 \* doses, nsubg)]

Then do the simulation of all these profiles. Using lapply is super efficient; the initial code with a for loop as suggested in the RxODE tutorial paper is about 20 times slower:

Startlapply <- Sys.time()  
   
#Run the simulations using lapply for speed  
 s = lapply(1:nsub, function(i) {  
#selects the parameters associated with the subject to be simulated  
 params <- params.all[i]  
#creates an eventTable with 7 doses every 24 hours  
 ev <- eventTable()  
 ev$add.dosing(  
 dose = params$AMT,  
 nbr.doses = 1,  
 dosing.to = 1,  
 rate = NULL,  
 start.time = 0  
 )  
#generates 4 random samples in a 24 hour period  
 ev$add.sampling(c(0, sort(round(sample(runif(600, 0, 1440), 4) / 60, 2))))  
#runs the RxODE simulation  
 x <- as.data.table(mod1$run(params, ev))  
#merges the parameters and ID number to the simulation output  
 x[, names(params) := params]  
 })  
   
#runs the entire sequence of 10000 subjects and binds the results to the object res  
 res = as.data.table(do.call("rbind", s))  
   
Stoplapply <- Sys.time()  
   
Stoplapply - Startlapply  
 #10,000 subjects simulated in:  
 #Time difference of 43.23035 secs

Clean up the results and prepare for analysis using NONMEM:

setnames(res, "time", "TIME")  
#single administered dose:  
 Dose <- params.all  
 Dose[,TIME:=0]  
 Dose[, C2 := 0]  
 Dose[, EVID := 101]  
 Dose[, DOSE := AMT / 1000]  
 res[, EVID := 0]  
 res[, centr := NULL]  
 res[, abs := NULL]  
 res[, DOSE := AMT / 1000]  
 res[, AMT := 0]  
 res <- rbind(res, Dose)  
 setkey(res, ID, TIME)  
#Add residual error  
 res[, DV := C2 \* exp(rnorm(length(C2), 0, sigma))]  
 res[, C2 := NULL]  
 res[, DV := round(DV)]  
#NONMEM EVID is just 1 instead of the RxODE EVID of 101  
 res[, EVIDNM := as.numeric(EVID == 101)]  
 res <- res[, .(ID, DOSE, V, CL, KA, TIME, EVID, AMT, DV, EVIDNM)]  
 write.table(res1,file="FullSIM180816\_1.csv",sep=",",  
 col.names=TRUE,quote=FALSE,row.names=FALSE)

Then PsN is used to sample 120 subjects stratified by dose from the 10,000 subjects and analyse these with NONMEM. This is repeated 500 times using PsN bootstrap functionality, creating both 500 sets of output and 500 data files to be analysed using nlmixr with the following PsN syntax:

bootstrap -nm\_version=nm743 runN024\_1.mod -samples=500 -sample\_size=120   
-stratify\_on=DOSE -no-run\_base\_model -seed=12345 -threads=15   
-keep\_covariance -directory=runN024\_1

and the following NONMEM syntax file for FOCEi:

$PROB ORAL1\_1CPT\_KAVCL MULTIPLE DOSE FOCEI runN024\_1  
$INPUT ID DOSE VI CLI KAI TIME EVIDNLMX AMT DV EVID  
$DATA FullSIM180816\_1.csv IGNORE=@  
$SUBR ADVAN6 TOL=6  
$MODEL COMP=(ABS,DEFDOSE) COMP=(CENTRAL,DEFOBS)  
$PK   
 CL = EXP(THETA(1)+ETA(1))  
 V = EXP(THETA(2)+ETA(2))  
 KA = EXP(THETA(3)+ETA(3))  
 S2 = V  
$DES   
 DADT(1) = -KA\*A(1)  
 DADT(2) = KA\*A(1)-CL\*(A(2)/V)  
$ERROR   
 IPRED = F   
 RESCV = THETA(4)   
 W = IPRED\*RESCV  
 IRES = DV-IPRED  
 IWRES = IRES/W  
 Y = IPRED+W\*EPS(1)  
$THETA 1 ;CL  
$THETA 4 ;V  
$THETA 0.1 ;Ka  
$THETA (0,0.2,1) ;RSV  
$OMEGA 0.2 0.2 0.2  
$SIGMA 1 FIX  
$EST NSIG=2 SIGL=6 PRINT=5 MAX=9999 NOABORT POSTHOC METHOD=COND  
 INTER NOOBT  
$COV

If these 500 files are to be analysed using nlmixr, single core analysis would take a long time, but the analyses can be run in parallel under Windows using doParallel functionality. First the model needs to be defined and pre-compiled:

one.compartment.oral.model <- function() {  
 ini({  
 # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lCl <- 1 #log Cl (L/hr)  
 lVc <- 4 #log V (L)  
 lKA <- 0.1 #log Ka (/hr)  
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- c(0, 0.2, 1)  
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Cl ~ 0.2  
 eta.Vc ~ 0.2  
 eta.KA ~ 0.2  
 })  
 model({  
 # Where the model is specified  
 # The model uses the ini-defined variable names  
 Cl <- exp(lCl + eta.Cl)  
 Vc <- exp(lVc + eta.Vc)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 d / dt(depot) = -KA \* depot  
 d / dt(centr) = KA \* depot - (Cl / Vc) \* centr  
 ## Concentration is calculated  
 cp = centr / Vc  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
modX<-nlmixr(one.compartment.oral.model)

Then a function do\_nlmixr is used that reads in the data file, runs the parameter estimation with the pre-compiled model, and then saves the output file, to be analysed at a later date. The code to analyse these data sets using FOCEI is:

do\_nlmixrODE\_FOCEi <- function(i) {  
 datr <-  
 read.csv(  
 paste("bs\_pr1\_", i, ".dta", sep = ""),  
 header = TRUE,  
 stringsAsFactors = F  
 )  
   
 fit <-  
 nlmixr(  
 modX,  
 datr,  
 est = "focei",  
 control = foceiControl(cores=1,sigdig=3,   
 outerOpt= "nlminb",diagXform="sqrt",covGillF=TRUE)  
 )  
 save(fit, file = paste("fit\_FOCEi\_ODE\_UUI\_", i, "\_1.Rdata", sep = ""))  
}

The code for SAEM with an ODE implementation is:

#SAEM with ODE:  
   
 do\_nlmixrODE\_SAEM <- function(i) {  
 datr <-  
 read.csv(  
 paste("bs\_pr1\_", i, ".dta", sep = ""),  
 header = TRUE,  
 stringsAsFactors = F  
 )  
   
 fit <-  
 nlmixr(  
 modX,  
 datr,  
 est = "saem",  
 control = saemControl(print = 50,n.burn=200,n.em=300)  
 )  
   
 save(fit, file = paste("fit\_SAEM\_ODE\_UUI\_", i, "\_1.Rdata", sep = ""))  
 }

To run these analyses in parallel, one needs to set up a local virtual cluster using the doParallel package, in this case with 7 cores, but adjust to your own hardware:

#install.packages("doParallel")   
 library(doParallel)  
 cl <- makeCluster(7, outfile="")  
 registerDoParallel(cl)

And then run the 500 analyses using foreach syntax:

nlmixr\_out <-  
 foreach(i = 1:500, .packages = c('nlmixr')) %dopar%   
 do\_nlmixrODE\_FOCEi(i)  
  
 nlmixrS\_out <-  
 foreach(i = 1:500, .packages = c('nlmixr')) %dopar%   
 do\_nlmixrODE\_SAEM(i)

You can then read in the output from the 500 nlmixr analyses. With the Unified User Interface, uniform storage is obtained for all estimation routines, and so a single read routine suffices.

Read\_nlmixr <- function(Identifier,Dose) {  
 for (i in 1:500) {  
 filename <- file.path('.',paste(Identifier, "\_UUI\_", i, Dose,".Rdata", sep = ""))  
 if (file.exists(filename)) {  
 load(filename)  
 TMSE <- as.numeric(fit$par.fixed$SE)  
 TM <- fit$theta  
 CovMatrix<-c("CovMatrix"=fit$covMethod)  
 if (length(TMSE)==0) {TMSE<-rep(NA,4)  
 CovMatrix<-c("CovMatrix"="NA")}  
 names(TMSE) <- paste(names(TM), "\_SE", sep = "")  
 OFV<-c("OFV"=fit$objDf$OBJF)  
 Time <- c("Time" = sum(fit$time))  
 IIV <- sqrt(diag(fit$omega))  
 run <- c("Run" = i)  
 MISSING\_CWRES <- c("MISSING\_CWRES" = sum(is.na(fit$CWRES)))  
 TM <- c(run, TM, TMSE, Time, IIV, MISSING\_CWRES,OFV)  
 TM <- as.data.frame(t(TM))  
 TM$CovMatrix<-CovMatrix  
 fit <- NULL  
 print(i)  
 if (i == 1) {  
 nlmixrpars <<- TM  
 } else{  
 nlmixrpars <<- rbind(nlmixrpars, TM)  
 }  
 }  
 }  
 save(nlmixrpars, file = paste(Identifier,Dose, ".Rdata", sep = ""))  
}  
  
Read\_nlmixr(Identifier = "fit\_FOCEi\_ODE",Dose="\_1")  
  
Read\_nlmixr(Identifier = "fit\_SAEM\_ODE",Dose="\_1")

The nlmixr results can then be compared with the Monolix and NONMEM output and plotted to see the results, as presented in the manuscript.

##### Code and methodology for running richly sampled data and multiple models

The nlmixr/SAEM, and nlmixr/FOCEI parameter estimation algorithms were compared with Monolix/SAEM and NONMEM/FOCEI in a richly-sampled data setting with multiple models.

Richly sampled profiles were simulated for 4 different dose levels of 30 subjects each, for a range of test models with: . one- or two-compartmental disposition . oral (first-order absorption), intravenous (IV) bolus, or IV infusion administration . linear or Michaelis-Menten (MM) clearance. In addition, three dosing and sampling scenarios were investigated: . a single administration with 19 samples over 72 hours . seven repeated daily administrations, with 15 samples over 24 hours after the 4th dose, 19 samples over 72 hours after the 7th dose, and 5 trough samples . the single administration profile followed by the repeated administrations profile, with a total of 58 samples over 12 days These combinations provided a total of 36 test cases. Inter-individual variability was applied to all pharmacokinetic parameters, and all IIVs were set to 30% (implemented as a diagonal matrix with no covariances). Proportional residual variability was set to 20%. All one-compartment models had a population Vc of 70 L, and all two-compartment models had an additional peripheral volume (Vp) of 40 L. For all oral absorption models, ka was set to 1.0 h-1. All models with linear elimination had a CL of 4.0 L/h, and for all models with non-linear MM elimination, CL was replaced with a Km of 250 mg/L and a Vmax of 1000 mg/h. All two-compartment models had inter-compartmental clearance (Q) set to 4.0 L/h.

The code for the full analysis is provided here along with some output graphs to demonstrate the results.

Load the packages and run all the FOCEI and SAEM analysis with models defined using ODEs:

library(nlmixr)  
library(data.table)  
  
one.compartment.IV.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lCl <- 1.6 #log Cl (L/hr)   
 lVc <- 4.5 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.1 #IIV V  
 eta.Cl ~ 0.1 #IIV Cl  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 Cl <- exp(lCl + eta.Cl)  
 # RxODE-style differential equations are supported  
 d / dt(centr) = -(Cl / Vc) \* centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
datr <- fread("BOLUS\_1CPT.csv")  
  
runno<-"N001"  
data<-datr[SD==1]  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N002"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N004"  
data<-datr  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
datr <- fread("Infusion\_1CPT.csv")  
  
rxClean()  
runno<-"N012"  
data<-datr[SD==1]  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N013"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N015"  
data<-datr  
fit <- nlmixr(one.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
  
one.compartment.IV.MM.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 lVM <- 7 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.5 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.15   
 eta.VM ~ 0.15  
 eta.KM ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 # RxODE-style differential equations are supported  
 d/dt(centr) = -(VM\*centr/Vc)/(KM+centr/Vc);  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
datr <- fread("BOLUS\_1CPTMM.csv")  
  
rxClean()  
runno<-"N009"  
data<-datr[SD==1]  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
#Modified starting values in the NONMEM analysis:  
one.compartment.IV.MM.model2 <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 lVM <- 6.9 #log Vmax (mg/hr)   
 lKM <- 5.8 #log KM (mg/L)   
 lVc <- 4.2 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.2   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.1   
 eta.VM ~ 0.14  
 eta.KM ~ 0.1  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 # RxODE-style differential equations are supported  
 d/dt(centr) = -(VM\*centr/Vc)/(KM+centr/Vc);  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
rxClean()  
runno<-"N010"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.IV.MM.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N011"  
data<-datr  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
datr <- fread("Infusion\_1CPTMM.csv")  
  
rxClean()  
runno<-"N020"  
data<-datr[SD==1]  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N021"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N022"  
data<-datr  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
one.compartment.oral.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lCl <- 1.8 #log Cl (L/hr)   
 lVc <- 4.7 #log V (L)   
 lKA <- 0.2 #log V (L)   
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3  
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Cl ~ 0.15  
 eta.Vc ~ 0.15   
 eta.KA ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Cl <- exp(lCl + eta.Cl)  
 Vc <- exp(lVc + eta.Vc)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 d/dt(depot) = -KA\*depot;  
 d/dt(centr) = KA\*depot-(Cl/Vc)\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
  
datr <- fread("ORAL\_1CPT.csv")  
  
rxClean()  
runno<-"N023"  
data<-datr[SD==1]  
fit <- nlmixr(one.compartment.oral.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
one.compartment.oral.model2 <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lCl <- 1.6 #log Cl (L/hr)   
 lVc <- 4.5 #log V (L)   
 lKA <- 0.2 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Cl ~ 0.15  
 eta.Vc ~ 0.15   
 eta.KA ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Cl <- exp(lCl + eta.Cl)  
 Vc <- exp(lVc + eta.Vc)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 d/dt(depot) = -KA\*depot;  
 d/dt(centr) = KA\*depot-(Cl/Vc)\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
rxClean()  
runno<-"N024"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.oral.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N026"  
data<-datr  
fit <- nlmixr(one.compartment.oral.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
one.compartment.oral.MM.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lVM <- 7 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.5 #log V (L)   
 lKA <- 0.2 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.15   
 eta.VM ~ 0.15  
 eta.KM ~ 0.15  
 eta.KA ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 d/dt(depot) = -KA\*depot;  
 d/dt(centr) = KA\*depot-(VM\*centr/Vc)/(KM+centr/Vc);  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
datr <- fread("ORAL\_1CPTMM.csv")  
  
  
rxClean()  
runno<-"N029"  
data<-datr[SD==1]  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N030"  
data<-datr[SD==0]  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N031"  
data<-datr  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(one.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
  
two.compartment.IV.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lCl <- 1.6 #log Cl (L/hr)   
 lVc <- 4.5 #log Vc (L)   
 lQ <- 1.6 #log Q (L/hr)   
 lVp <- 4 #log Vp (L)   
  
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.15   
 eta.Cl ~ 0.15  
 eta.Vp ~ 0.15   
 eta.Q ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 Cl <- exp(lCl + eta.Cl)  
 Vp <- exp(lVp + eta.Vp)  
 Q <- exp(lQ + eta.Q)  
 # RxODE-style differential equations are supported  
 K10<- Cl/Vc  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(centr) = K21\*periph-K12\*centr-K10\*centr;  
 d/dt(periph) =-K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
  
datr <- fread("BOLUS\_2CPT.csv")  
  
  
rxClean()  
runno<-"N032"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N033"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N035"  
data<-datr  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
  
datr <- fread("Infusion\_2CPT.csv")  
  
  
rxClean()  
runno<-"N046"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N047"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N049"  
data<-datr  
fit <- nlmixr(two.compartment.IV.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
  
two.compartment.IV.MM.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lVM <- 7.1 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.3 #log Vc (L)   
 lQ <- 1.5 #log Q (L/hr)   
 lVp <- 4 #log Vp (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.VM ~ 0.15  
 eta.KM ~ 0.1  
 eta.Vc ~ 0.15   
 eta.Q ~ 0.15  
 eta.Vp ~ 0.15   
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 Vc <- exp(lVc + eta.Vc)  
 Q <- exp(lQ + eta.Q)  
 Vp <- exp(lVp + eta.Vp)  
 # RxODE-style differential equations are supported  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(centr) = K21\*periph-K12\*centr-(VM\*centr/Vc)/(KM+centr/Vc);  
 d/dt(periph) =-K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
  
  
datr <- fread("BOLUS\_2CPTMM.csv")  
  
rxClean()  
runno<-"N040"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.IV.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
two.compartment.IV.MM.model2 <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lVM <- 7 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.5 #log Vc (L)   
 lQ <- 1.5 #log Q (L/hr)   
 lVp <- 4 #log Vp (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.VM ~ 0.15  
 eta.KM ~ 0.15  
 eta.Vc ~ 0.15   
 eta.Q ~ 0.15  
 eta.Vp ~ 0.15   
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 Vc <- exp(lVc + eta.Vc)  
 Q <- exp(lQ + eta.Q)  
 Vp <- exp(lVp + eta.Vp)  
 # RxODE-style differential equations are supported  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(centr) = K21\*periph-K12\*centr-(VM\*centr/Vc)/(KM+centr/Vc);  
 d/dt(periph) =-K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
rxClean()  
runno<-"N041"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N042"  
data<-datr  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
datr <- fread("Infusion\_2CPTMM.csv")  
  
  
rxClean()  
runno<-"N054"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
two.compartment.IV.MM.model3 <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 # Simple numeric expressions are supported  
 lVM <- 7 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.5 #log Vc (L)   
 lQ <- 1.5 #log Q (L/hr)   
 lVp <- 4 #log Vp (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.VM ~ 0.1  
 eta.KM ~ 0.1  
 eta.Vc ~ 0.1   
 eta.Q ~ 0.1  
 eta.Vp ~ 0.1   
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 Vc <- exp(lVc + eta.Vc)  
 Q <- exp(lQ + eta.Q)  
 Vp <- exp(lVp + eta.Vp)  
 # RxODE-style differential equations are supported  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(centr) = K21\*periph-K12\*centr-(VM\*centr/Vc)/(KM+centr/Vc);  
 d/dt(periph) =-K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
rxClean()  
runno<-"N055"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.IV.MM.model3,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model3,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N056"  
data<-datr  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.IV.MM.model2,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
  
two.compartment.oral.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 lCl <- 1.6 #log Cl (L/hr)   
 lVc <- 4.5 #log Vc (L)   
 lQ <- 1.6 #log Q (L/hr)   
 lVp <- 4 #log Vp (L)   
 lKA <- 0.2 #log V (L)   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.15   
 eta.Cl ~ 0.15  
 eta.Vp ~ 0.15   
 eta.Q ~ 0.15  
 eta.KA ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 Cl <- exp(lCl + eta.Cl)  
 Vp <- exp(lVp + eta.Vp)  
 Q <- exp(lQ + eta.Q)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 K10<- Cl/Vc  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(depot) = -KA\*depot;  
 d/dt(centr) = KA\*depot+K21\*periph-K12\*centr-K10\*centr;  
 d/dt(periph) = -K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
  
datr <- fread("ORAL\_2CPT.csv")  
  
  
rxClean()  
runno<-"N060"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.oral.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N061"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.oral.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N063"  
data<-datr  
fit <- nlmixr(two.compartment.oral.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
  
two.compartment.oral.MM.model <- function(){  
 ini({ # Where initial conditions/variables are specified  
 # '<-' or '=' defines population parameters  
 lVM <- 7.1 #log Vmax (mg/hr)   
 lKM <- 5.7 #log KM (mg/L)   
 lVc <- 4.5 #log Vc (L)   
 lQ <- 1.6 #log Q (L/hr)   
 lVp <- 4.1 #log Vp (L)   
 lKA <- 0.22 #log V (L)   
   
 # Bounds may be specified by c(lower, est, upper), like NONMEM:  
 # Residuals errors are assumed to be population parameters  
 prop.err <- 0.3   
 # Between subject variability estimates are specified by '~'  
 # Semicolons are optional  
 eta.Vc ~ 0.15   
 eta.Vp ~ 0.15   
 eta.VM ~ 0.15  
 eta.KM ~ 0.15  
 eta.Q ~ 0.15  
 eta.KA ~ 0.15  
 })  
 model({ # Where the model is specified  
 # The model uses the ini-defined variable names  
 Vc <- exp(lVc + eta.Vc)  
 Vp <- exp(lVp + eta.Vp)  
 Q <- exp(lQ + eta.Q)  
 VM <- exp(lVM + eta.VM)  
 KM <- exp(lKM + eta.KM)  
 KA <- exp(lKA + eta.KA)  
 # RxODE-style differential equations are supported  
 K12<- Q/Vc  
 K21<- Q/Vp  
 d/dt(depot)=-KA\*depot;  
 d/dt(centr)= KA\*depot+K21\*periph-K12\*centr -   
 (VM\*centr/Vc)/(KM+centr/Vc);  
 d/dt(periph)=-K21\*periph+K12\*centr;  
 ## Concentration is calculated  
 cp = centr / Vc;  
 # And follows proportional error estimated by prop.err  
 cp ~ prop(prop.err)  
 })  
}  
  
  
datr <- fread("ORAL\_2CPTMM.csv")  
  
rxClean()  
runno<-"N068"  
data<-datr[SD==1]  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N069"  
data<-datr[SD==0]  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
runno<-"N070"  
data<-datr  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="saem")  
fit  
save(fit,file=paste("fitSAEM\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL  
  
rxClean()  
fit <- nlmixr(two.compartment.oral.MM.model,data,est="focei")  
fit  
save(fit,file=paste("fitFOCEI\_ODE\_UUI",runno,".Rdata",sep=""))  
fit<-NULL

Read in nlmixr results:

Read\_nlmixr<-function(RunNumbers,Analysis,path,stub,out,  
 tracePLT=FALSE,CovMat=TRUE,printOUT=FALSE){  
  
for (i in RunNumbers) {  
 desc<-paste(Analysis," run",i,sep="")  
 print(i)  
 load(file.path(path,paste(stub,i,".Rdata",sep="")))  
 fit  
  
 if (tracePLT) {traceplot(fit)  
 iG(paste("Traceplot for parameters from",desc))}  
  
 theta<-as.data.frame(fit$par.fixed)  
 theta$Parameter<-rownames(theta)  
  
 if (CovMat==TRUE) {CovMatrix<-fit$covMethod  
 if(is.null(CovMatrix)) {  
 CovMatrix<-"No covariance step"  
 TMSE<-data.frame(value=theta$"Est.",Parameter=rownames(theta),  
 variable="SE",stringsAsFactors=FALSE)  
 TMSE<-TMSE[!TMSE$Parameter=='prop.err',]  
 TMSE$value<-NA  
 } else {  
 TMSE<-data.frame(value=theta$SE,Parameter=rownames(theta),  
 variable="SE",stringsAsFactors=FALSE)  
 TMSE<-TMSE[!TMSE$Parameter=='prop.err',]  
 }  
 } else {  
 TMSE<-data.frame(value=theta$"Est.",Parameter=rownames(theta),  
 variable="SE",stringsAsFactors=FALSE)  
 TMSE<-TMSE[!TMSE$Parameter=='prop.err',]  
 TMSE$value<-NA  
 }  
  
 TM<-data.frame(value=theta$"Est.",Parameter=rownames(theta),  
 variable="Estimate",stringsAsFactors=FALSE)  
 TM$Parameter[TM$Parameter=='prop.err']<-"RSV"  
  
 TM<-rbind(TM,TMSE)  
 Time<-data.frame(Parameter="Time",variable="Estimate",  
 value=sum(fit$time),stringsAsFactors=FALSE)  
 TM<-rbind(TM,Time)  
 IIV<-data.frame(Parameter=row.names(fit$omega),variable="EtaCV",  
 value=100\*sqrt(diag(fit$omega)),stringsAsFactors=FALSE)  
 TM<-rbind(TM,IIV)  
 OFV<-data.frame(Parameter="OFV",variable="Estimate",value=fit$objDf$OBJF,  
 stringsAsFactors=FALSE)  
 TM<-rbind(TM,OFV)  
  
 FFP<-function(number,sigdig=3) {   
 res<-formatC(signif(number,digits=sigdig),  
 digits=sigdig,format="fg", flag="#")   
 res<-sub('\\.$',"",res)   
 res}  
 TM1<-data.frame(  
 Parameter=ifelse(theta$Parameter!="prop.err",  
 as.character(theta$Parameter),"Proportional error"),  
 Estimate=FFP(as.numeric(theta$"Est."),3),  
 SE=ifelse(theta$Parameter!="prop.err",FFP(as.numeric(theta$SE),3)," "),  
 Backtransformed=ifelse(theta$Parameter!="prop.err",  
 paste(FFP(exp(as.numeric(theta$"Est.")),3),  
 "(",FFP(exp(as.numeric(theta$"Est.")-1.96\*as.numeric(theta$SE)),3),  
 "/",FFP(exp(as.numeric(theta$"Est.")+1.96\*as.numeric(theta$SE)),3),  
 ")",sep=""),  
 FFP(100\*(as.numeric(theta$"Est.")),3)),  
 stringsAsFactors=FALSE)  
  
 TM2<-data.frame(Parameter=c(IIV$Parameter,"Time (sec)"),  
 Estimate=FFP(c(IIV$value,Time$value),3),  
 SE="",Backtransformed="",stringsAsFactors=FALSE)  
 TM2<-rbind(TM1,TM2)  
 TM2$Parameter[grep("eta.",TM2$Parameter)]<-paste(   
 TM2$Parameter[grep("eta.",TM2$Parameter)],"(%)")  
 TM2$Parameter<-gsub("eta.", "IIV ", TM2$Parameter)  
 TM2$Parameter<-gsub("^l", "", TM2$Parameter)  
 names(TM2)[names(TM2)=="Backtransformed"]<-"Back-transformed (95%CI)"  
  
 if (printOUT==TRUE){  
 if (CovMat==TRUE) { iT(TM2,paste("Parameters from ",desc,  
 ifelse(CovMatrix=="No covariance step",  
 "; no covariance step possible",  
 paste("; SEs estimated with matrix",CovMatrix)),", and OFV = ",  
 round(fit$objDf$OBJF,2),sep=""),RowNames=FALSE)  
 } else { iT(TM2,paste("Parameters from",desc),RowNames=FALSE)}  
 }  
  
 TM$run<-paste("run",i,sep="")  
 TM$analysis<-Analysis  
  
 if (i==RunNumbers[[1]]) {nlmixrpars<<-TM  
 }else{nlmixrpars<<-rbind(nlmixrpars,TM)}  
 assign(out,nlmixrpars,envir = .GlobalEnv)  
 }  
  
}#End Read\_nlmixr  
  
  
RunNumbersODE<-Cs(N001,N002,N004,N009,N010,N011,N012,N013,N015,N020,N021,  
N022,N023,N024,N026,N029, N030,N031,N032,N033,N035,N040,N041,N042,N046,  
N047,N049,N054,N055,N056,N060,N061,N063,N068, N069,N070)

#nlmixr/FOCEI using ODEs: nlminb with sqrt scaling:  
Read\_nlmixr(RunNumbers=RunNumbersODE,Analysis="nlmixr/FOCEI",path=".",  
 stub="fitFOCEI\_ODE\_UUI",  
 out="nlmixrFOCEI\_ODE",tracePLT=FALSE,CovMat=TRUE,  
 printOUT=FALSE)  
nlmixrFOCEI\_ODE<-data.table(nlmixrFOCEI\_ODE)  
  
#SAEM using ODEs:  
Read\_nlmixr(RunNumbersODE,Analysis="nlmixr/SAEM",path=".",  
 stub="fitSAEM\_ODE\_UUI",  
 out="nlmixrSAEM\_ODE",tracePLT=FALSE,CovMat=TRUE,  
 printOUT=FALSE)  
nlmixrSAEM\_ODE<-data.table(nlmixrSAEM\_ODE)

The nlmixr results can then be compared with the Monolix and NONMEM output and plotted to see the results, as presented in the manuscript.