

Simulation and parameter estimation with RxODE and nlmixr

Rik Schoemaker, PhD

The nlmixr development team:

Wenping Wang, Matt Fidler, Teun Post, Richard
Hooijmaijers, Mirjam Trame, Yuan Xiong,
Justin Wilkins and Rik Schoemaker



Course outline

- Introduction to RxODE
- Introduction to nlmixr
- Hands on with the warfarin PK models (part 1)
- Lunch
- Performance of nlmixr compared to NONMEM (FOCEI) and Monolix (SAEM)
- Advanced capabilities nlmixr
- Hands on with the warfarin PK and PKPD models (part 2)

RxODE is pharmacometric simulation software as an open-source R package

- Written by Wenping Wang and Matt Fidler, available on CRAN¹ and GitHub², and described in a tutorial in CPT:PSP³ and with online documentation⁴
- Simulation of ODEs was already possible in R (using deSolve), but was slow and virtually impossible to code with flexible dosing history
- RxODE has rapid execution due to compilation in C
- RxODE allows fully flexible dosing history
- Stable and mature software for Windows, OS X, Linux
- Requires external compilers (provided by Rtools on Windows)

[1] CRAN: <https://cran.r-project.org/web/packages/RxODE>

[2] GitHub: <https://github.com/nlmixrdevelopment/RxODE>

[3] Wang W et al. CPT:PSP (2016) 5, 3–10.

[4] RxODE packagedown: <https://nlmixrdevelopment.github.io/RxODE>

Basic example load the library and define the ODEs

```
library(RxODE)
```

```
## set up the system of differential equations (ODEs)
```

```
odeKA1 <- "
```

```
  d/dt(depot)   = -ka*depot;                # This is compartment number 1 (depot)
```

```
  d/dt(central) = ka*depot-(cl/v)*central; # This is compartment number 2 (central)
```

```
  C1=central/v;                # Calculates concentration from amount
```

```
"
```

Compile the model and set the parameter values

```
library(RxODE)

## set up the system of differential equations (ODEs)
odeKA1 <- "
  d/dt(depot)    = -ka*depot;          # This is compartment number 1 (depot)
  d/dt(central) = ka*depot-(cl/v)*central; # This is compartment number 2 (central)
  C1=central/v;          # Calculates concentration from amount
"

## compile the model
modKA1 <- RxODE(model = odeKA1)

## provide the parameter values to be simulated:
Params <-
  c(ka = log(2)/0.5, # 1/h (absorption half-life of 30 minutes)
    cl = 0.135,      # L/h
    v = 8)           # L
```

Create an eventTable that defines the doses and the sampling times

```
library(RxODE)
## set up the system of differential equations (ODEs)
odeKA1 <- "
  d/dt(depot)    = -ka*depot;          # This is compartment number 1 (depot)
  d/dt(central) = ka*depot-(cl/v)*central; # This is compartment number 2 (central)
  C1=central/v;          # Calculates concentration from amount
"

## compile the model
modKA1 <- RxODE(model = odeKA1)
## provide the parameter values to be simulated:
Params <-
  c(ka = log(2)/0.5, # 1/h (absorption half-life of 30 minutes)
    cl = 0.135,      # L/h
    v = 8)           # L

## create an empty event table that stores both dosing and sampling information :
ev <- eventTable()

## add a dose to the event table:
ev$add.dosing(dose = 500) #mg

## add time points to the event table where concentrations will be simulated
## these actions are cumulative
ev$add.sampling(seq(0, 120, 0.1))
```

Run the model and plot the results

```
library(RxODE)
## set up the system of differential equations (ODEs)
odeKA1 <- "
  d/dt(depot)    = -ka*depot;          # This is compartment number 1 (depot)
  d/dt(central) = ka*depot-(cl/v)*central; # This is compartment number 2 (central)
  C1=central/v;          # Calculates concentration from amount
"

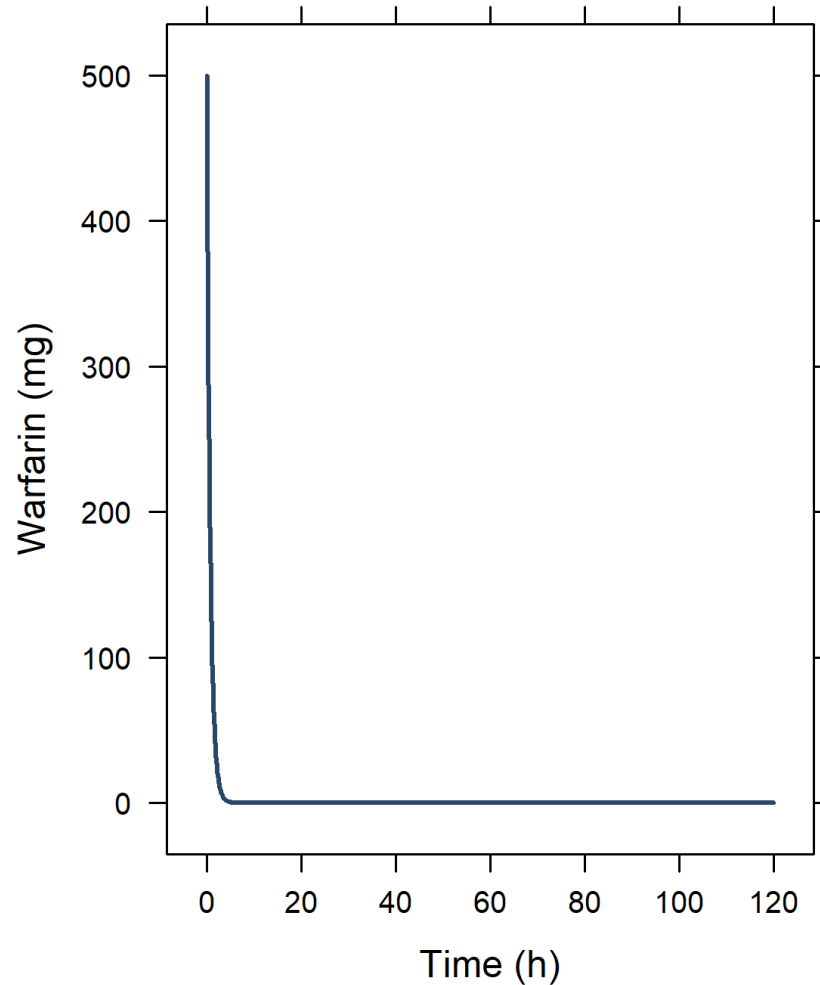
## compile the model
modKA1 <- RxODE(model = odeKA1)
## provide the parameter values to be simulated:
Params <-
  c(ka = log(2)/0.5, # 1/h (absorption half-life of 30 minutes)
    cl = 0.135,      # L/h
    v = 8)           # L

## create an empty event table that stores both dosing and sampling information :
ev <- eventTable()
## add a dose to the event table:
ev$add.dosing(dose = 500) #mg
## add time points to the event table where concentrations will be simulated
## these actions are cumulative
ev$add.sampling(seq(0, 120, 0.1))

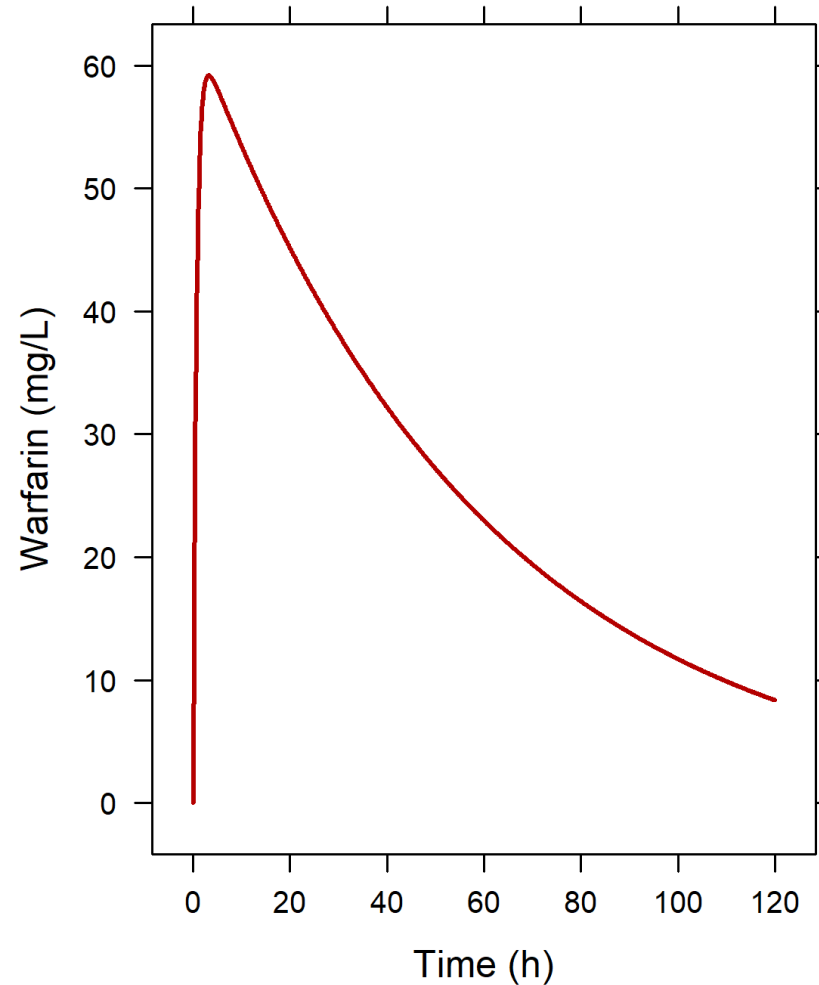
## Then solve the system
## The output from rxSolve is a solved RxODE object,
## By making it a data.frame only the simulated values are retained:
Res <- data.frame(rxSolve(modKA1, Params, ev))
```

Single bolus dose in the first (depot) compartment

Depot compartment amounts



Central compartment concentrations



Adding extra doses (expand the existing eventTable): three additional infusions in the central compartment

```
## Extend the eventTable by adding three infusions to the central compartment
## Remember: updates to the eventTable are cumulative

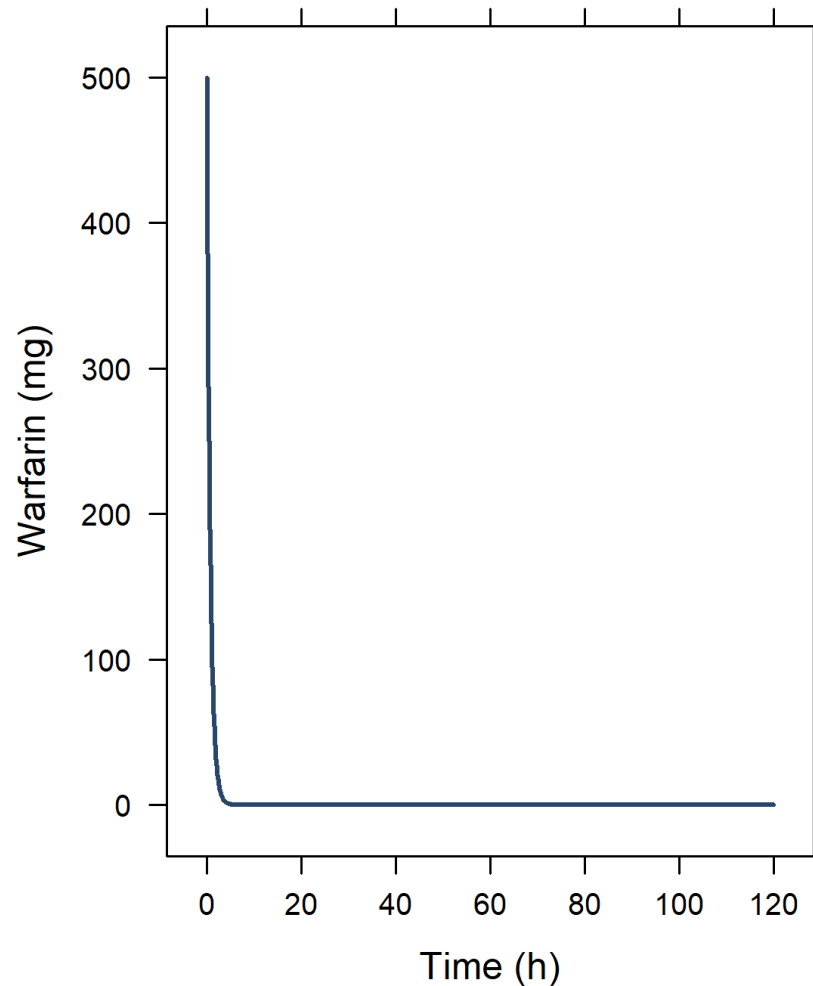
ev$add.dosing(
  dose = 250,          #mg
  nbr.doses = 3,       #add three doses
  dosing.to = 2,       #add them to the second ODE in the model (=central)
  dosing.interval = 12, #h; set the doses 12 hours apart
  rate = 125,          #mg/h; infuse at a rate of 125 mg/h, resulting in 2-hour infusions
  start.time = 36      #h; have the three doses start at 36h
)

Res <- data.frame(rxSolve(modKA1, Params, ev))
```

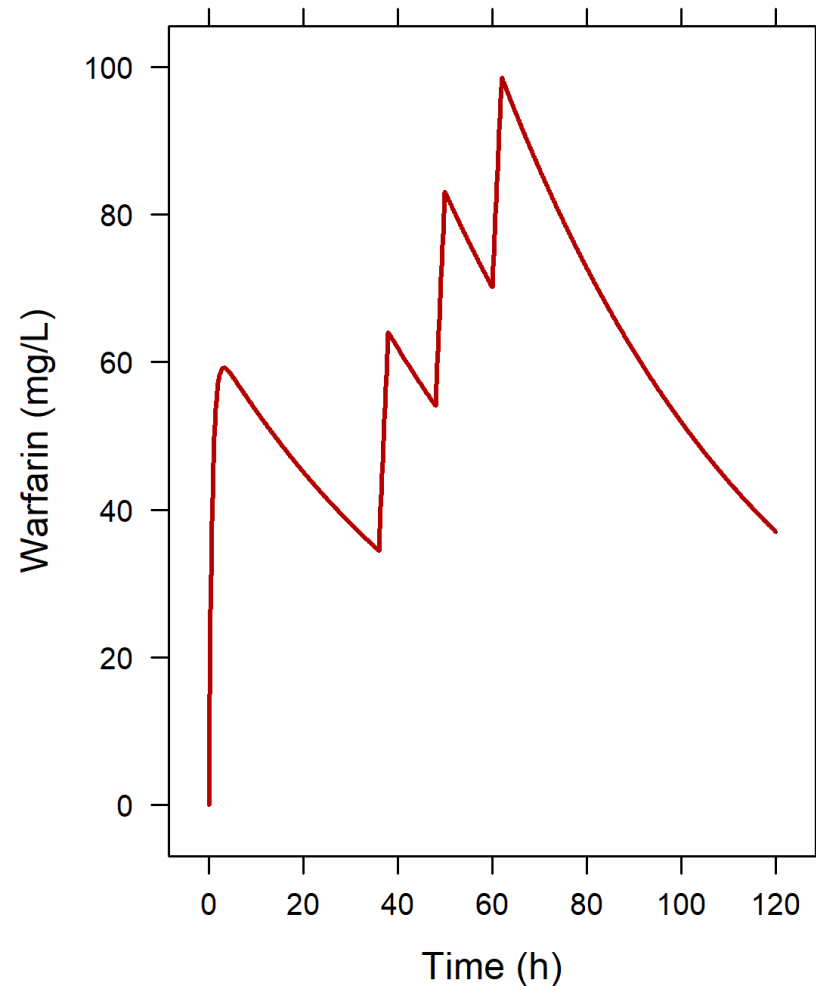
Multiple dose in different compartments

Only the first dose goes into the depot (first) compartment

Depot compartment amounts



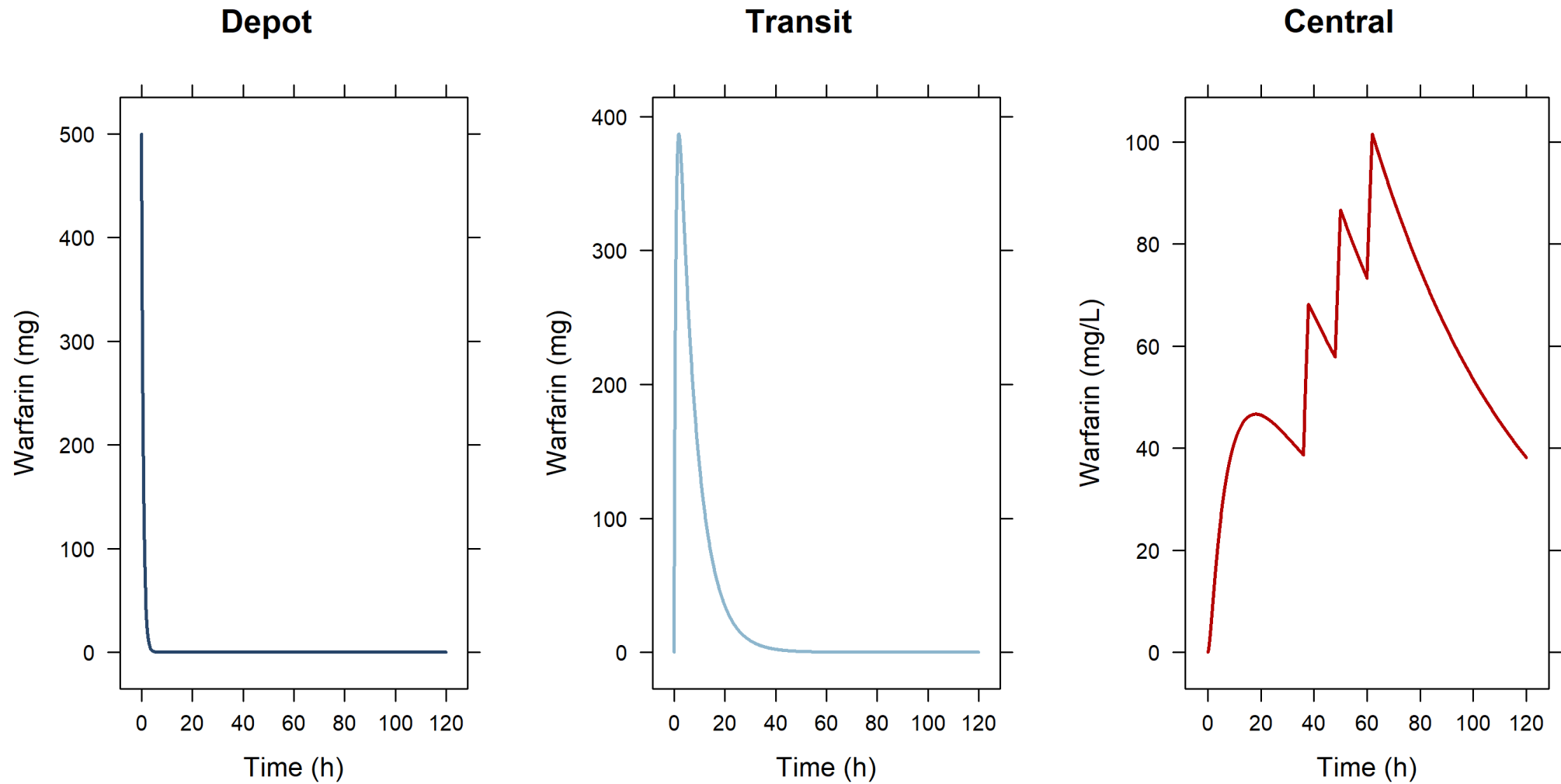
Central compartment concentrations



Add a transit compartment...

```
odeKA1trans <- "  
  d/dt(depot)   = -ka*depot;  
  d/dt(central) = ktr*trans-(cl/v)*central; # update to central: input from trans  
  d/dt(trans)   = ka*depot-ktr*trans;      # transit compartment between depot and central  
  C1=central/v;  
  "  
  
## compile the model  
modKA1trans <- RxODE(model = odeKA1trans)  
  
## provide the extra ktr parameter:  
Params2 <- c(  
  ka = log(2)/0.5, # 1/h (absorption half-life of 30 minutes)  
  cl = 0.135,      # L/h  
  v = 8,           # L  
  ktr = log(2)/5)  # 1/h (transit half-life of 5 hours)  
  
## the eventTable does not have to change  
Res <- data.frame(rxSolve(modKA1trans, Params2, ev))  
  
## if the trans compartment had been put as second compartment above,  
## the eventTable would need an update to infuse in compartment 3 instead
```

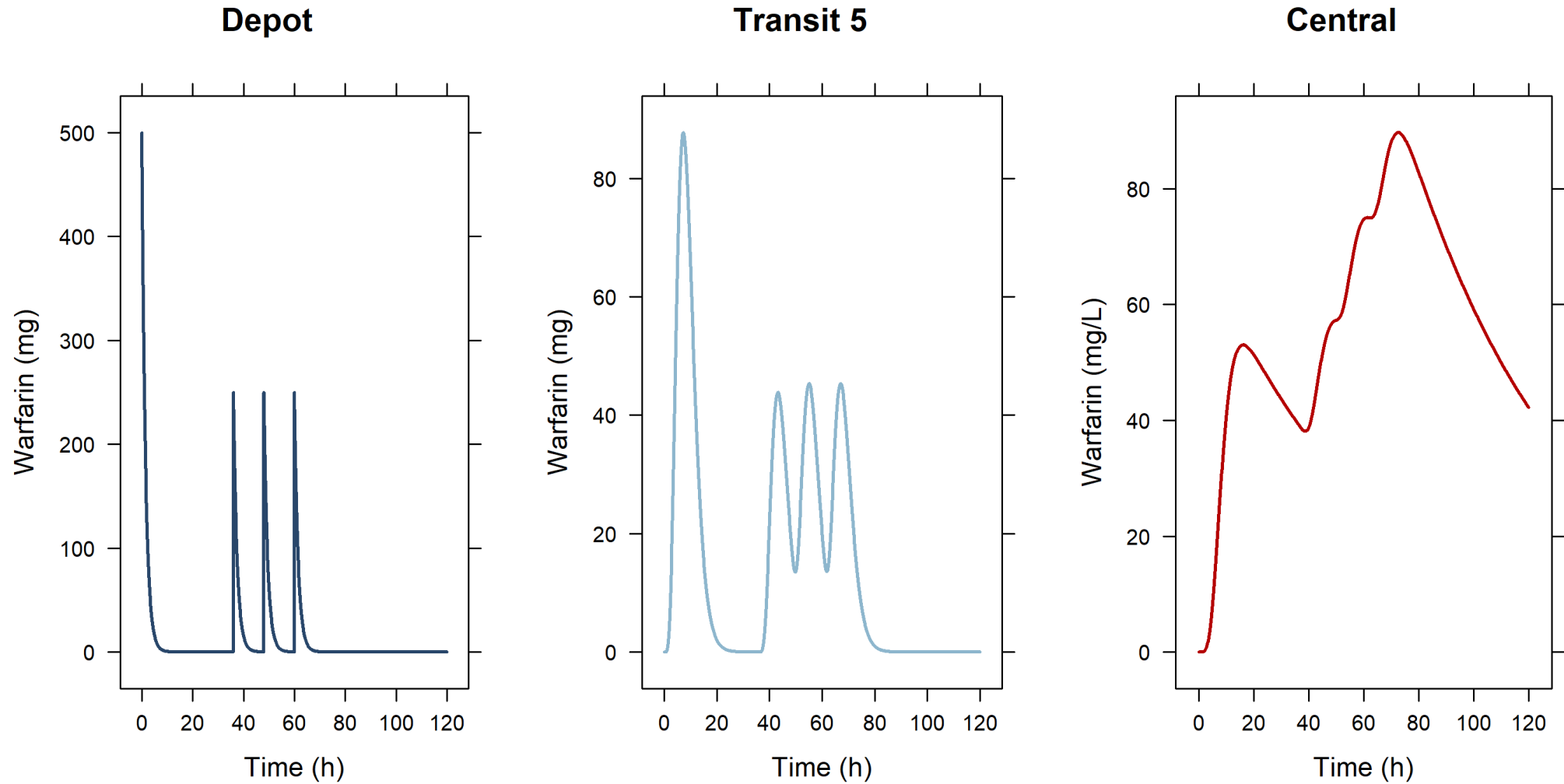
...adding a transit compartment between depot and central



Or with five transit compartments and only bolus doses in the depot...

```
odeKA5trans <- "  
  d/dt(depot)    = -ktr*depot;  
  d/dt(central) = ktr*trans5-(cl/v)*central; # update to central: input from trans5  
  d/dt(trans1)   = ktr*depot-ktr*trans1;      # use same constant for every compartment  
  d/dt(trans2)   = ktr*trans1-ktr*trans2;  
  d/dt(trans3)   = ktr*trans2-ktr*trans3;  
  d/dt(trans4)   = ktr*trans3-ktr*trans4;  
  d/dt(trans5)   = ktr*trans4-ktr*trans5;  
  C1=central/v;  
  "  
  
ev3 <- eventTable()  
ev3$add.dosing(dose = 500) # mg; 1st bolus  
ev3$add.dosing(  
  dose = 250,          # mg  
  nbr.doses = 3,        # 3 additional doses (bolusses because rate is absent so rate=0)  
  dosing.interval = 12, # h; at 12 hour intervals  
  dosing.to = 1,        # dosed into depot (compartment 1)  
  start.time = 36       # h; starting at 36 hours  
)  
ev3$add.sampling(seq(0, 120, 0.1))  
  
Params3 <-  
  c(ktr = log(2)/1, # use same constant for every compartment  
    cl = 0.135,  
    v = 8)  
  
modKA5trans <- RxODE(model = odeKA5trans)  
  
Res <- data.frame(rxSolve(modKA5trans, Params3, ev3))
```

...adding 5 transit compartments between depot and central
and giving 4 bolus doses in the 1st (depot) compartment



Hands-on session I: RxODE simulations

- Make sure you can either access the Otago server at <https://student.desktop.otago.ac.nz/vpn/index.html> or run your own nlmixr installation
- Examine the code in PAWS_1.R to run pre-programmed simulations and try out your own variations

You need to simulate before you can estimate

- With simulation covered, you can start to think about estimation
- Combine the simulation core with estimation routines and you get:

nlmixr!

nlmixr is an open-source R package

- Written by Wenping Wang and Matt Fidler, and available on CRAN¹ and GitHub²:
 - builds on RxODE
 - combined with nlme, SAEM, and FOCEI estimation routines, provides an R package for parameter estimation in nonlinear mixed effect models
 - under very active development!
- nlmixr is completely free and open, and does not depend on any other commercial tool such as NONMEM or Monolix
- nlmixr provides an efficient and versatile way to specify pharmacometric models (both closed-form and ODEs) and dosing scenarios, with rapid execution due to compilation in C

[1] <https://cran.r-project.org/web/packages/nlmixr>

[2] <https://github.com/nlmixrdevelopment/nlmixr>

nlmixr is an open-source R package

- Models are defined using a unified user interface (UI): common input and output structure for the various estimation algorithms
- `xpose.nlmixr`¹ written by Justin Wilkins provides linkage to the new Xpose package², written by Ben Guiaastrennec, feeding the uniform output into a highly flexible diagnostics package
- The shinyMixR³ project management tool written by Richard Hooijmaijers and Teun Post provides an interface to `nlmixr` from both the R command line and a user-friendly browser-based Shiny dashboard application
- `nlmixr` requires access to compilers (e.g. using Rtools) and Python: both a full-package windows installer is available⁴, and instructions on managing your own installation⁵
- Documentation is available in the form of a bookdown ([nlmixr.github.io](https://nlmixrdevelopment.github.io)) written and curated by Teun Post
- Runs on Linux, Windows, and OS X

[1] <https://github.com/nlmixrdevelopment/xpose.nlmixr>

[2] <https://CRAN.R-project.org/package=xpose>

[3] <https://github.com/RichardHooijmaijers/shinyMixR>

[4] <https://github.com/nlmixrdevelopment/nlmixr/releases/>

[5] <https://nlmixrdevelopment.github.io/nlmixr>

Defining **nlmixr** models

- Models are defined using a function containing an initialisation block (**ini**) and a model definition block (**model**)

```
One.comp.KA.solved <- function() {  
  ini({  
    # Where initial conditions/variables are specified  
  
  })  
  model({  
    # Where the model is specified  
  
  })  
}
```

Defining **nlmixr** models

- The **ini** block defines the parameters
 - Thetas and residual error defined using assign operators (**<-** or **=**)
 - Etas defined using a model formula (**~**)
- Parameter names, starting values, labels (using **#**), bounds for some estimation routines (like FOCEI)

```
One.comp.KA.solved <- function() {  
  ini({  
    # Where initial conditions/variables are specified  
    lka <- log(1.15) #Log ka (1/h)  
    lcl <- log(0.135) #Log CL (L/h)  
    lv <- log(8) #Log V (L)  
    prop.err <- 0.15 #proportional error (SD/mean)  
    add.err <- 0.6 #additive error (mg/L)  
    eta.ka ~ 0.5 #IIV ka  
    eta.cl ~ 0.1 #IIV cl  
    eta.v ~ 0.1 #IIV v  
  })  
  model({  
  
  })  
}
```

Defining **nlmixr** models

- The **model** block defines
 - the relationship between thetas and etas
 - the model structure using either closed-form solutions or ODEs
 - the residual error structure, and where it is applied

```
One.comp.KA.solved <- function() {  
  ini({  
  
  })  
  model({  
    # Where the model is specified  
    cl <- exp(lcl + eta.cl)  
    v  <- exp(lv + eta.v)  
    ka <- exp(lka + eta.ka)  
    ## solved system example  
    ## where residual error is assumed to follow proportional and additive error  
    linCmt() ~ prop(prop.err) + add(add.err)  
  })  
}
```

Running **nlmixr** models: the full model

```
One.comp.KA.solved <- function() {  
  ini({  
    # Where initial conditions/variables are specified  
    lka <- log(1.15) #log ka (1/h)  
    lcl <- log(0.135) #log CL (L/h)  
    lv <- log(8) #log V (L)  
    prop.err <- 0.15 #proportional error (SD/mean)  
    add.err <- 0.6 #additive error (mg/L)  
    eta.ka ~ 0.5 #IIV ka  
    eta.cl ~ 0.1 #IIV cl  
    eta.v ~ 0.1 #IIV v  
  })  
  model({  
    # Where the model is specified  
    cl <- exp(lcl + eta.cl)  
    v <- exp(lv + eta.v)  
    ka <- exp(lka + eta.ka)  
    ## solved system example  
    ## where residual error is assumed to follow proportional and additive error  
    linCmt() ~ prop(prop.err) + add(add.err)  
  })  
}
```

Running nlmixr models: check the model code

```
## Check the model and some of the assumptions made by nlmixr
## note assumption that AMT goes into CMT=1 is not shown
nlmixr(One.comp.KA.solved)
```

```
> nlmixr(One.comp.KA.solved)
__ RxODE-based 1-compartment model with first-order absorption __
-- Initialization: -----
Fixed Effects ($theta):
      lka      lcl      lv
0.1397619 -2.0024805  2.0794415
```

```
Omega ($omega):
      eta.ka eta.cl eta.v
eta.ka    0.5   0.0   0.0
eta.cl    0.0   0.1   0.0
eta.v     0.0   0.0   0.1
```

```
-- mu-referencing ($muRefTable): -----
```

theta	eta
lcl	eta.cl
lv	eta.v
lka	eta.ka

```
-- Model: -----
# where the model is specified
cl <- exp(lcl + eta.cl)
v  <- exp(lv + eta.v)
ka <- exp(lka + eta.ka)
## solved system example
## where residual error is assumed to follow proportional and additive error
linCmt() ~ prop(prop.err) + add(add.err)
```

Running **nlmixr** models with the **nlmixr** command

```
## estimate parameters using nlmixr:
fitOne.comp.KA.solved_S <-
  nlmixr(
    One.comp.KA.solved,      #the model definition
    PKdata,                 #the data set
    est = "saem",           #the estimation algorithm (SAEM)
                           #the SAEM minimisation options:
    saemControl(nBurn = 200, #200 SAEM burn-in iterations (the default)
               nEm    = 300, #300 EM iterations (the default)
               print = 50),  #only print every 50th estimation step
    tableControl(cwres = TRUE) #calculates NONMEM-style conditional weighted residuals for diagnostics
  )

## results are stored in the nlmixr object and can be viewed:
fitOne.comp.KA.solved_S
```


nlmixr output for SAEM

```
> fitOne.comp.KA.solved_S
-- nlmixr SAEM(solved); OBJF by FOCEi approximation fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 468.2237 945.5308 973.7345      -464.7654      25.11291

-- Time (sec; fitOne.comp.KA.solved_S$time): -----
      saem setup table cwres covariance other
elapsed -6.67 20.23  0.02 20.28      0.02  2.21

-- Population Parameters (fitOne.comp.KA.solved_S$parFixed or fitOne.comp.KA.sol
Registered S3 method overwritten by 'R.oo':
  method      from
throw.default R.methodss3

      Parameter      Est.      SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
1ka      log ka (1/h) -0.584  0.225 38.6      0.558 (0.359, 0.867) 76.7      39.5%
1cl      log cl (L/h) -2.01  0.0526 2.61      0.134 (0.121, 0.148) 28.6      4.05%
1v      log V (L)      2.05  0.0449 2.2      7.74 (7.09, 8.45) 22.1      13.3%
prop.err      0.077
add.err      0.583

Covariance Type (fitOne.comp.KA.solved_S$covMethod): linFim
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitOne.comp.KA.solved_S$omega) or correlation (fitOne.comp.KA.solved_S$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitOne.comp.KA.solved_S$shrink

-- Fit Data (object fitOne.comp.KA.solved_S is a modified tibble): -----
# A tibble: 251 x 22
  ID    TIME    DV    EVID    PRED    RES    WRES    IPRED    IRES    IWRES    CPRED    CRES    CWRES    eta.ka    eta.cl    eta.v    rx1c    cl    v    ka    depot    central
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1      0.5      0      0    3.13 -3.13 -2.05    1.32 -1.32 -2.22    2.50 -2.50 -2.33    -1.00  0.699 -0.0530  1.32 0.269  7.34 0.205  90.3    9.66
2 1      1      1.9      0    5.47 -3.57 -2.31    2.48 -0.579 -0.944    4.62 -2.72 -1.58    -1.00  0.699 -0.0530  2.48 0.269  7.34 0.205  81.5   18.2
3 1      2      3.3      0    8.51 -5.21 -3.27    4.41 -1.11 -1.64    7.86 -4.56 -1.70    -1.00  0.699 -0.0530  4.41 0.269  7.34 0.205  66.4   32.4
# ... with 248 more rows
```

The labels in the ini block end up in the output, and the log-transformed parameters are returned with a back-transformation and 95%CIs

```
> fitOne.comp.KA.solved_S
-- nlmixr SAEM(Solved); OBJF by FOCEi approximation fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 468.2237 945.5308 973.7345      -464.7654      25.11291

-- Time (sec; fitOne.comp.KA.solved_S$time): -----
      saem setup table cwres covariance other
elapsed -6.67 20.23 0.02 20.28      0.02 2.21

-- Population Parameters (fitOne.comp.KA.solved_S$parFixed or fitOne.comp.KA.solved_S$parFree): -----
Registered S3 method overwritten by 'R.oo':
  method      from
throw.default R.methodsS3

Parameter Est. SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
1ka log ka (1/h) -0.584 0.225 38.6 0.558 (0.359, 0.867) 76.7 39.5%
1cl log cl (L/h) -2.01 0.0526 2.61 0.134 (0.121, 0.148) 28.6 4.05%
1v log v (L) 2.05 0.0449 2.2 7.74 (7.09, 8.45) 22.1 13.3%
prop.err 0.077
add.err 0.583

Covariance Type (fitOne.comp.KA.solved_S$covMethod): linFim
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitOne.comp.KA.solved_S$omega) or correlation (fitOne.comp.KA.solved_S$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitOne.comp.KA.solved_S$shrink

-- Fit Data (object fitOne.comp.KA.solved_S is a modified tibble): -----
# A tibble: 251 x 22
  ID TIME DV EVID PRED RES WRES IPRED IRES IWRES CPRED CRES CWRES eta.ka eta.cl eta.v rx1c cl v ka depot central
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1 0.5 0 0 3.13 -3.13 -2.05 1.32 -1.32 -2.22 2.50 -2.50 -2.33 -1.00 0.699 -0.0530 1.32 0.269 7.34 0.205 90.3 9.66
2 1 1 1.9 0 5.47 -3.57 -2.31 2.48 -0.579 -0.944 4.62 -2.72 -1.58 -1.00 0.699 -0.0530 2.48 0.269 7.34 0.205 81.5 18.2
3 1 2 3.3 0 8.51 -5.21 -3.27 4.41 -1.11 -1.64 7.86 -4.56 -1.70 -1.00 0.699 -0.0530 4.41 0.269 7.34 0.205 66.4 32.4
# ... with 248 more rows
```

Running **nlmixr** models: save the object, and examine parameter trace plots when using SAEM to check convergence

results are stored in the nlmixr object and can be viewed:

```
fitOne.comp.KA.solved_S
```

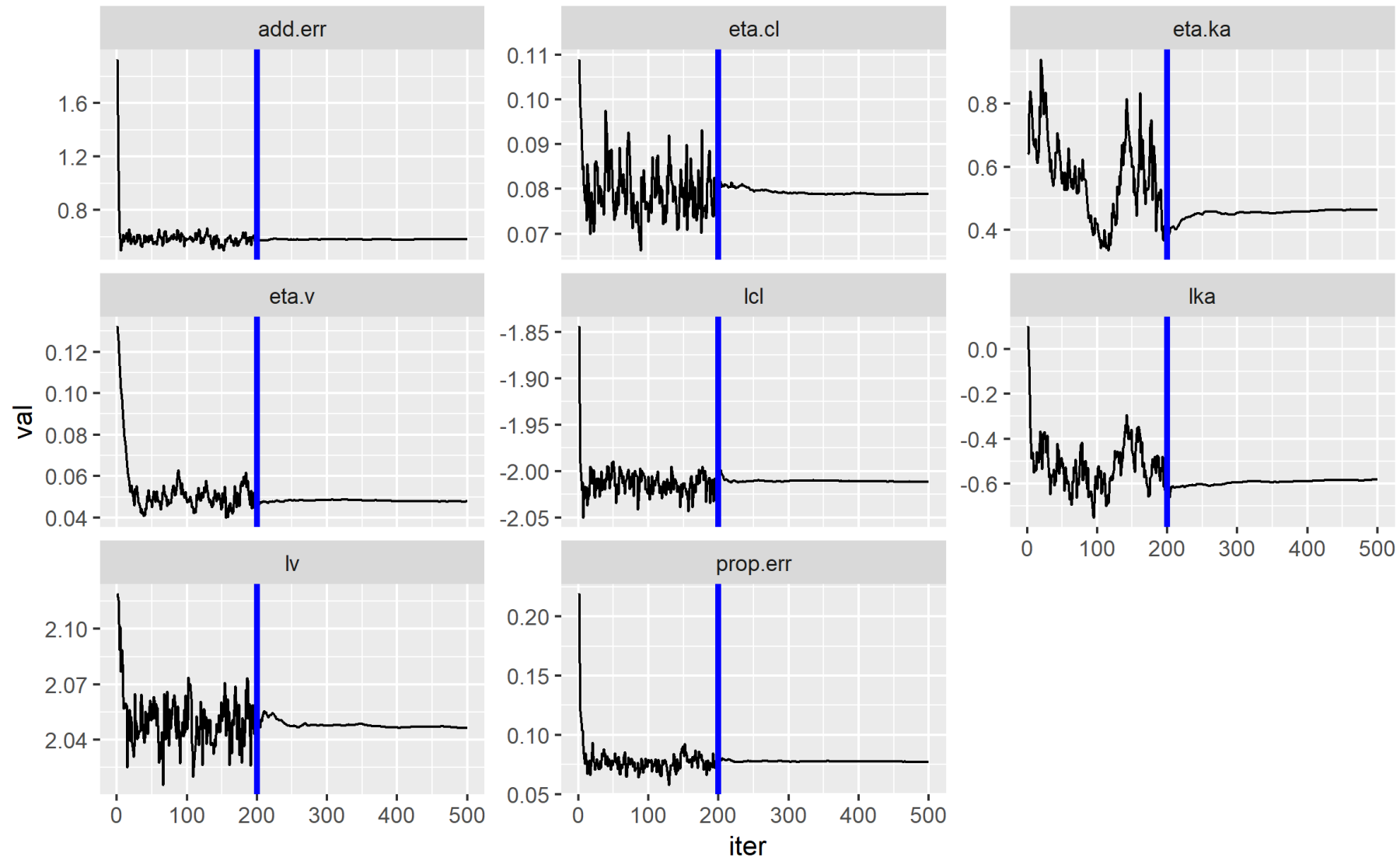
and saved for future use or reference:

```
save(fitOne.comp.KA.solved_S, file = "fitOne.comp.KA.solved_S.Rdata")
```

and for SAEM, convergence can be checked using a parameter trace plot:

```
traceplot(fitOne.comp.KA.solved_S)
```

Traceplot for SAEM parameter estimates using `traceplot` command



nlmixr is linked to Ben Guiastronnec's xpose* package that uses ggplot2

```
## the nlmixr object can be transformed into an xpose object to allow diagnostics with the new xpose package  
## the link between nlmixr and xpose is provided by the xpose.nlmixr package  
## only xpose_data_nlmixr is from xpose.nlmixr  
## all further commands (see cheatsheet) are from the xpose package
```

```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
```

```
## this can also be used to generate trace plots (parameters vs iterations:)  
prm_vs_iteration(xpdb.1s)  
## to remove the path to the script from the plot use:  
prm_vs_iteration(xpdb.1s,caption=NULL)
```

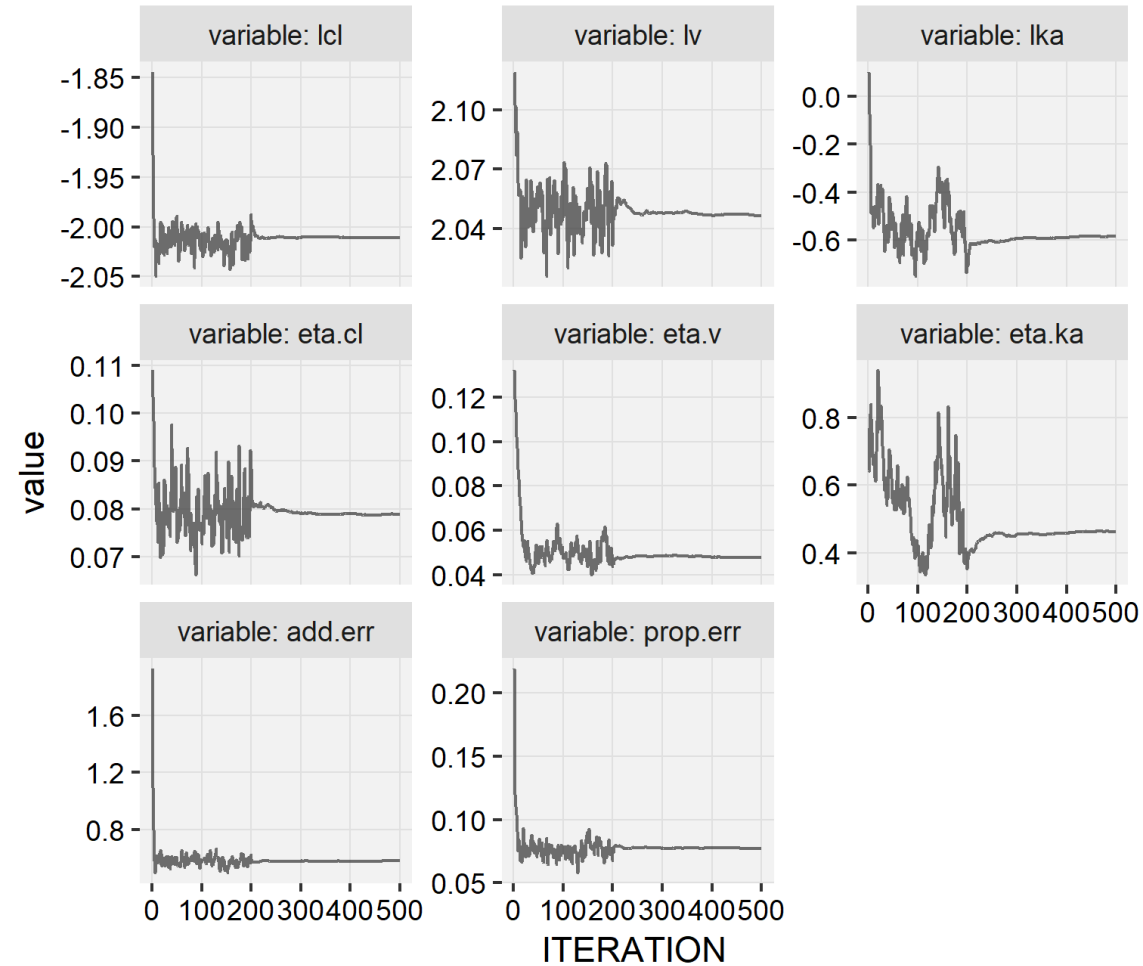
*<https://uupharmacometrics.github.io/xpose/>

Traceplot for SAEM parameter estimates using xpose

Parameter value vs. ITERATION | One.comp.KA.sol

Method: SAEM, minimization time: 32.7

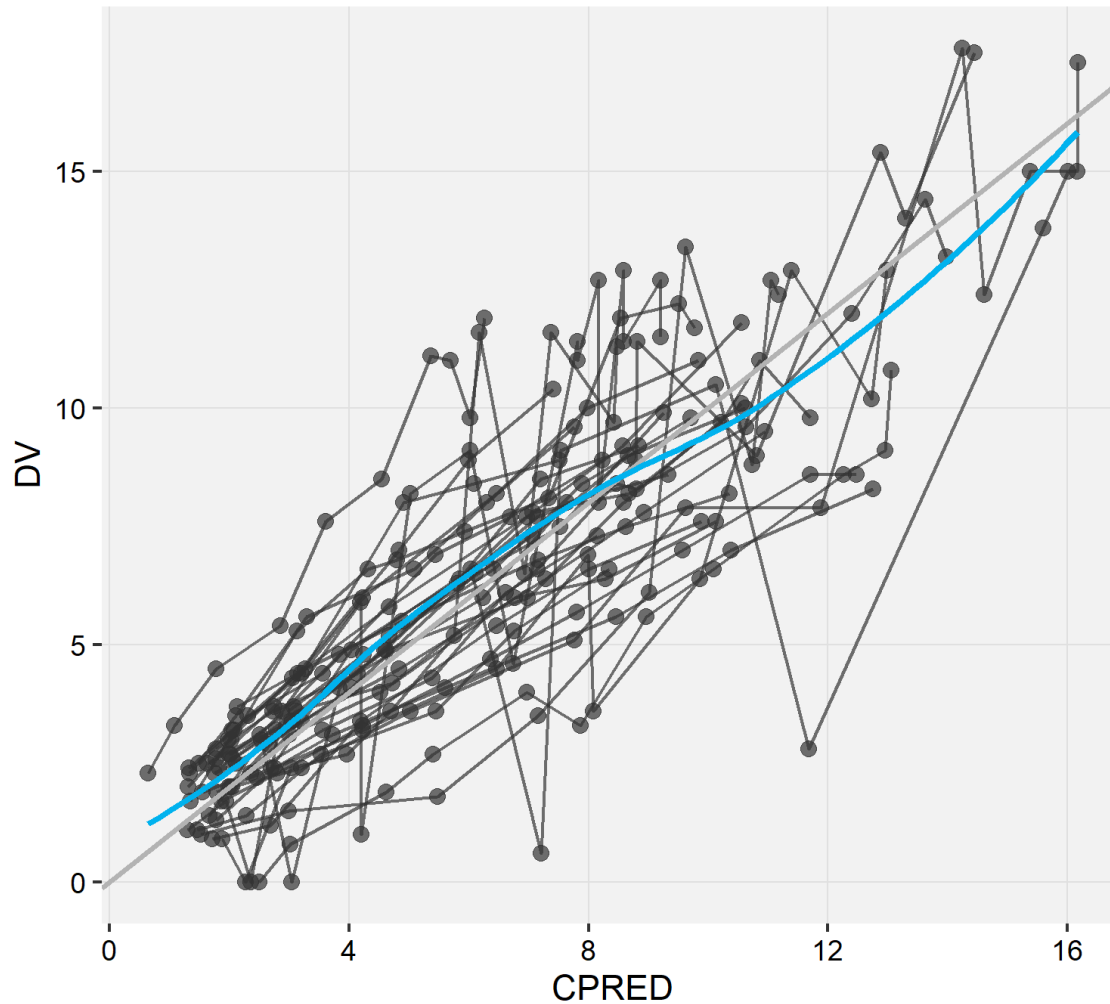
Termination message: na



DV vs conditional population predictions (CPRED) using xpose

DV vs. CPRED | One.comp.KA.solved

Ofv: 468.2



```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
## dv vs pred plot:
dv_vs_pred(xpdb.1s,
           caption = NULL)
```

DV vs PRED using xpose

by default model typical predictions (PRED) are assigned to CPRED (conditional population predictions):
`list_vars(xpdb.1s)`

```
List of available variables for problem no. 1
- Subject identifier (id)           : ID
- Dependent variable (dv)          : DV
- Independent variable (idv)       : TIME
- Dose amount (amt)                : AMT
- Event identifier (evid)          : EVID
- Model typical predictions (pred) : CPRED
- Model individual predictions (ipred) : IPRED
- Model parameter (param)         : cl, v, ka
- Eta (eta)                        : eta.ka, eta.cl, eta.v
- Residuals (res)                  : RES, WRES, IRES, IWRES, CRES, CWRES
- Categorical covariates (catcov)  : SEX
- Continuous covariates (contcov)  : WT, AGE
- Not attributed (na)              : PRED, rx1c, depot, central
```

if you want this to be PRED instead, these can be updated, either using 'standard' syntax:

```
xpdb.1s<-set_var_types(xpdb.1s,pred = 'PRED')
```

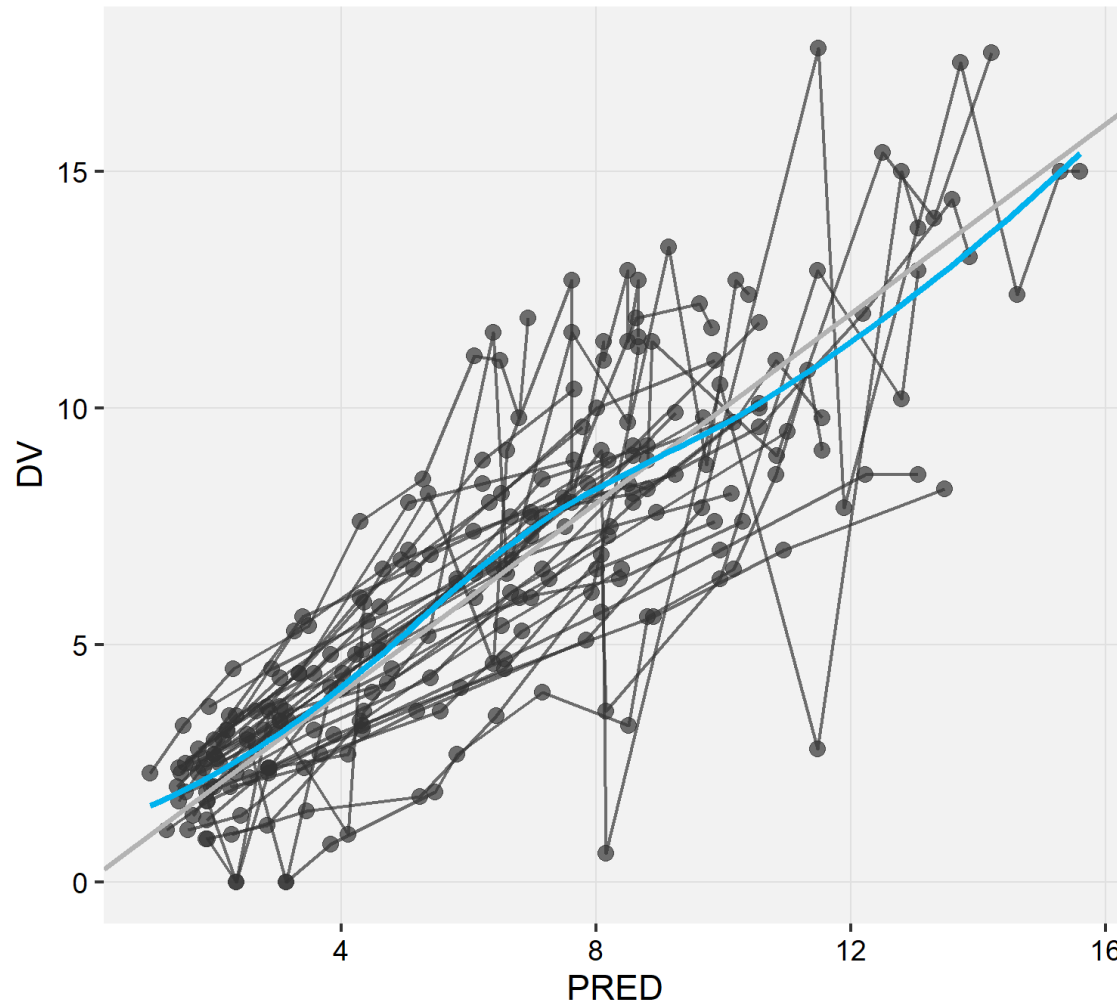
or using magrittr piping type code:

```
xpdb.1s<-xpdb.1s %>% set_var_types(pred = 'PRED')
```


DV vs PRED using xpose

DV vs. PRED | One.comp.KA.solved

Ofv: 468.2

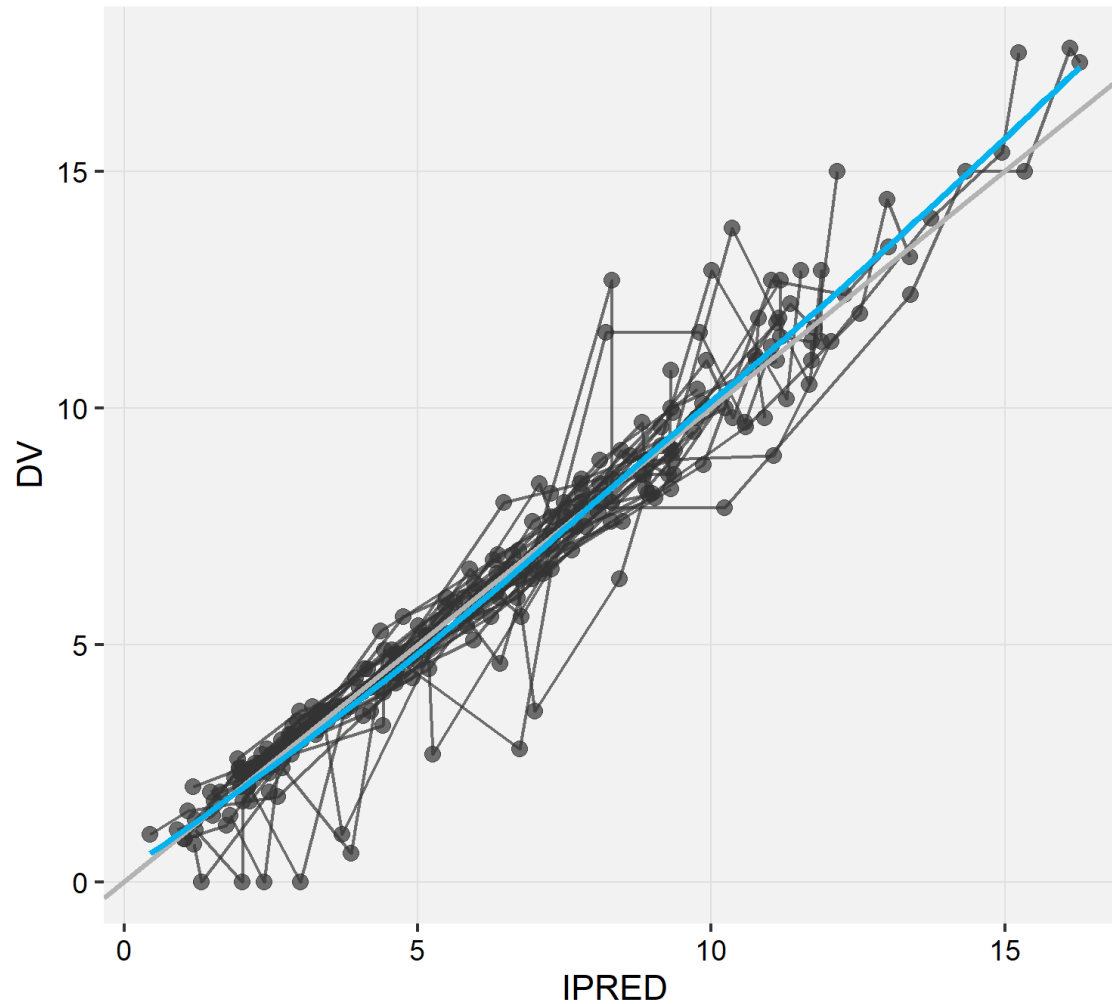


```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
## plot PRED instead of CPRED:
xpdb.1s<-xpdb.1s %>% set_var_types(pred = 'PRED')
## dv vs pred plot:
dv_vs_pred(xpdb.1s,
           caption = NULL)
```

DV vs IPRED using xpose

DV vs. IPRED | One.comp.KA.solved

Ofv: 468.2, Eps shrink: -17.5 [1]

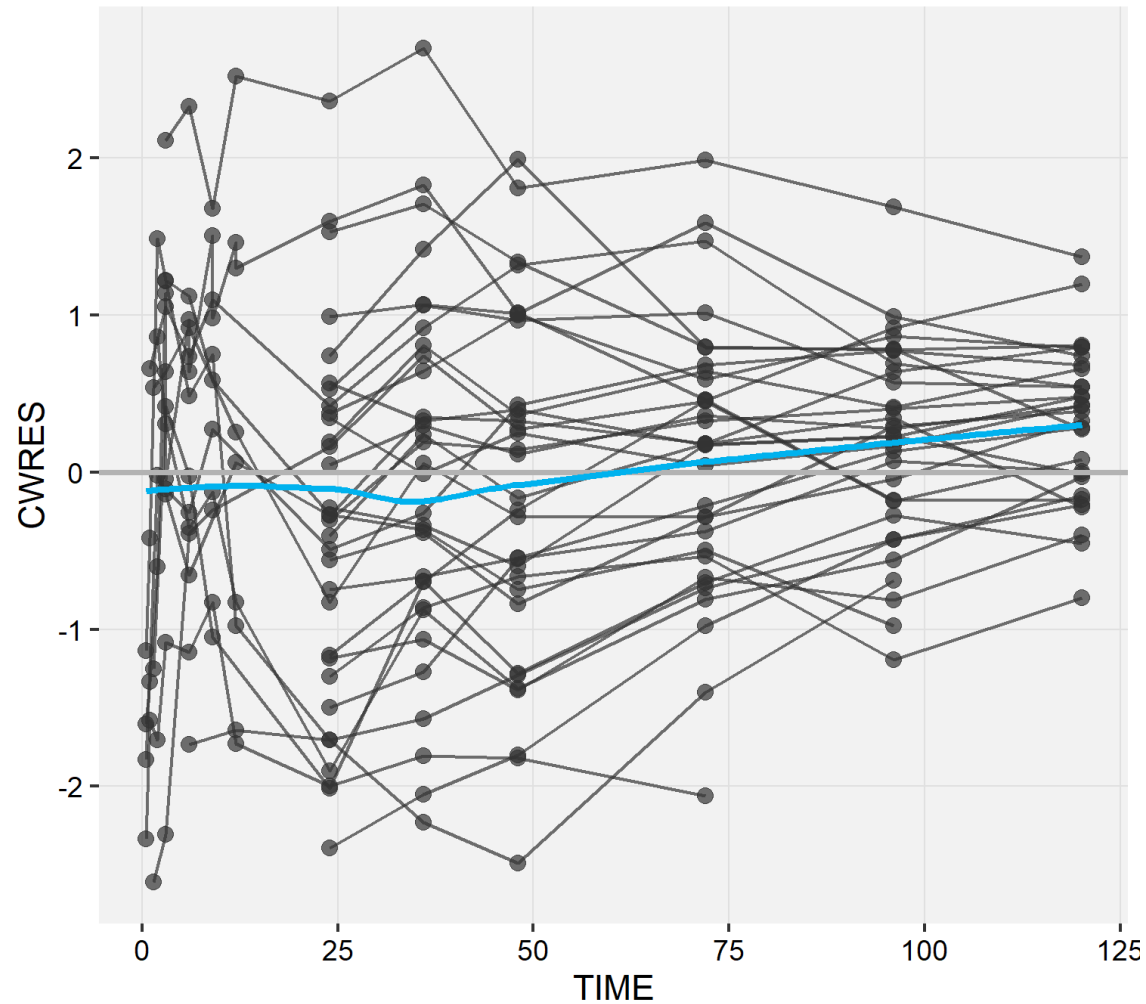


```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
## dv vs ipred plot:
dv_vs_ipred(xpdb.1s,
            caption = NULL)
```

Conditional weighted residuals vs. time using xpose

CWRES vs. TIME | One.comp.KA.solved

Ofv: 468.2

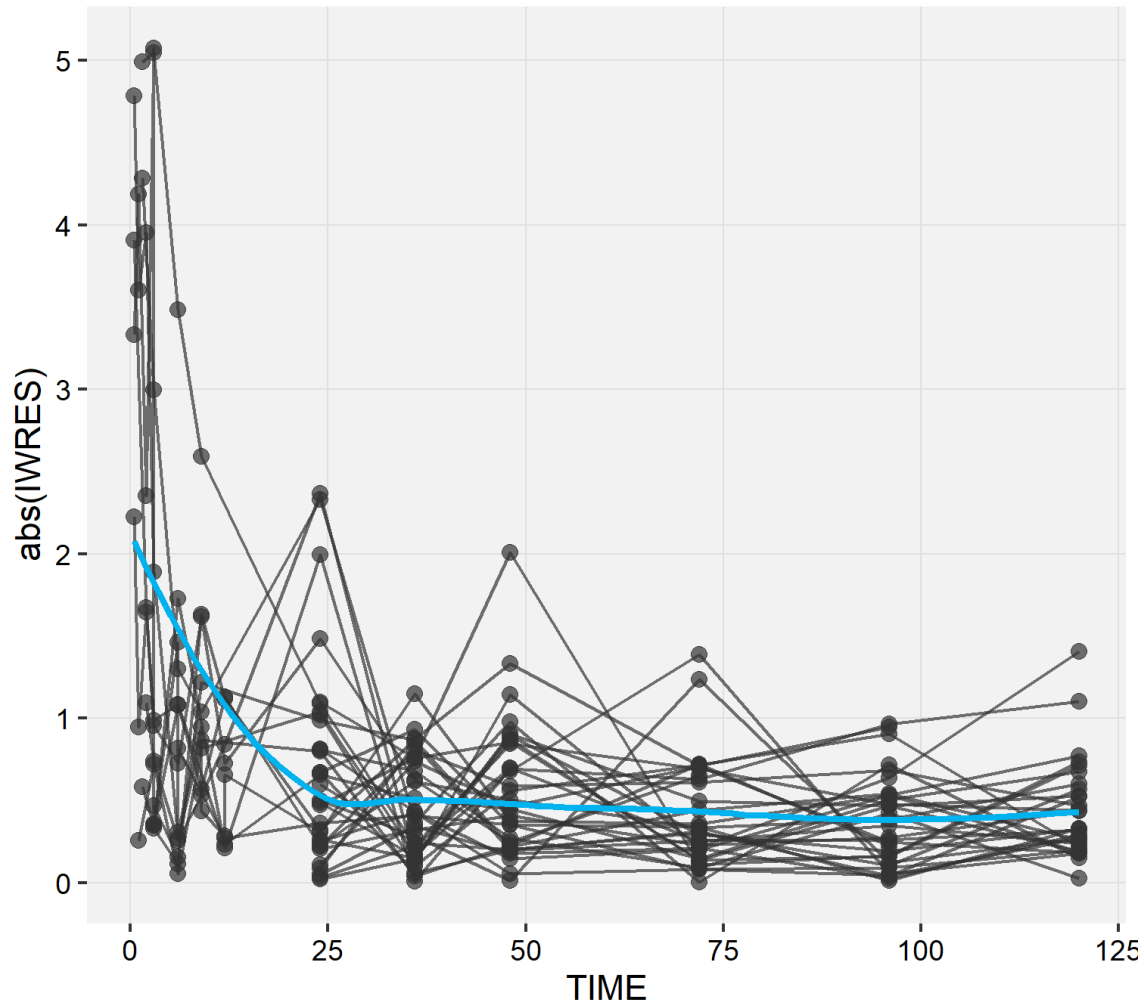


```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
## CWRES vs time:
res_vs_idv(xpdb.1s,                               #the xpose object
            res = "CWRES",                         #examine CWRES
            idv = "TIME",                          #as a function of time
            caption = NULL)
```

Absolute values of individual weighted residuals vs. time

abs(IWRES) vs. TIME | One.comp.KA.solved

Ofv: 468.2



```
xpdb.1s <- xpose_data_nlmixr(fitOne.comp.KA.solved_S)
## |IWRES| vs time:
absval_es_vs_idv(xpdb.1s,      #the xpose object
                  res = "IWRES", #examine |IWRES|
                  idv = "TIME",  #as a function of time
                  caption = NULL)

## Issue with absorption?
```

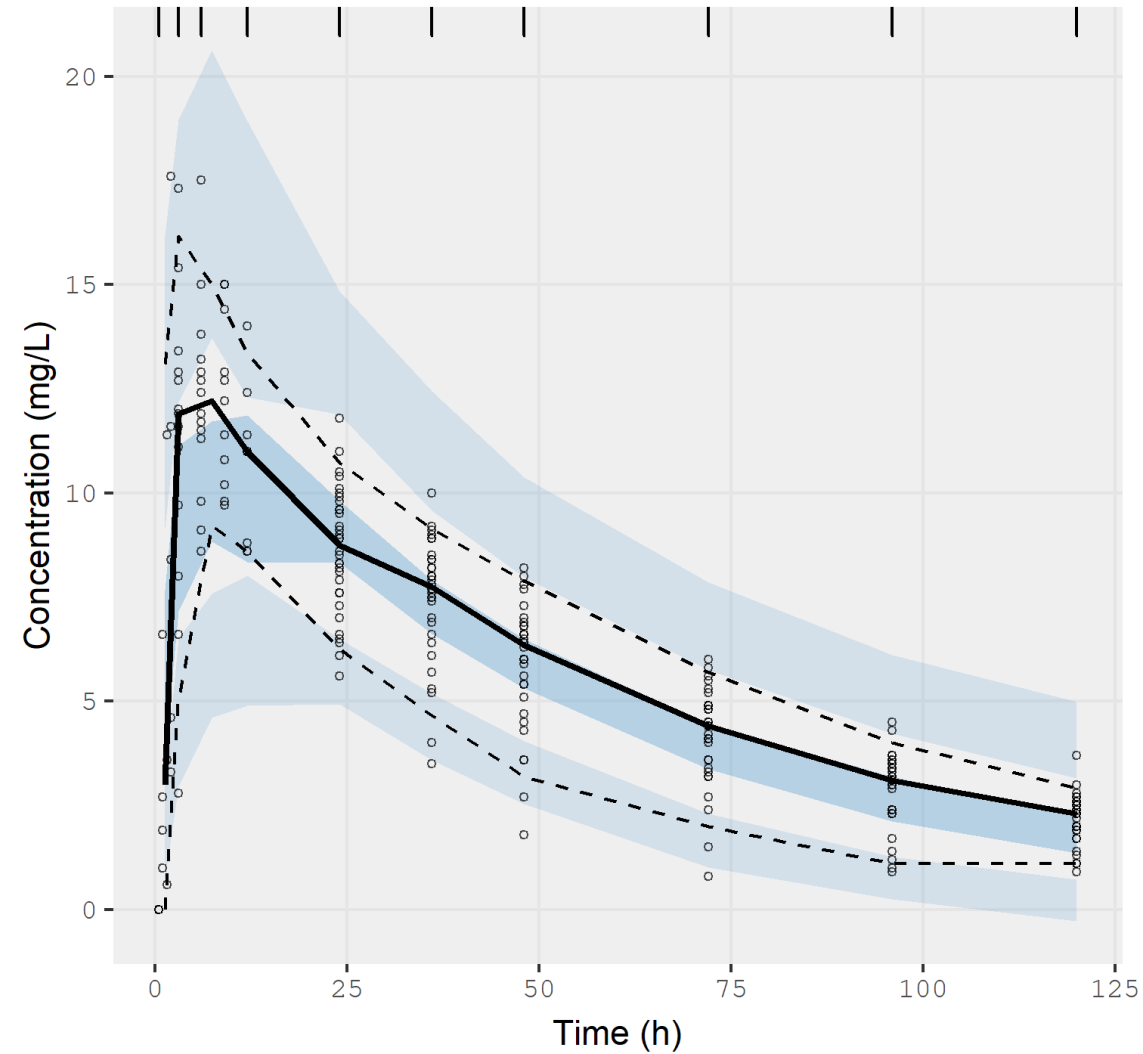
nlmixr is linked to Ron Keizer's vpc* package

```
## nlmixr comes with its own built-in vpc functionality that uses Ron Keizer's vpc package
## see the cheatsheet for further options
vpc_ui(
  fitOne.comp.KA.solved_S,      #the nlmixr object
  n = 500,                      #number of trials simulated using estimated
                                # parameters and study sampling structure
  show = list(obs_dv = TRUE),   #additional items to show, like the observations
  xlab = "Time (h)",            #x-axis label
  ylab = "Concentration (mg/L)", #y-axis label
  title = "VPC for first order absorption PopPK model"
)
```

*<http://vpc.ronkeizer.com/>

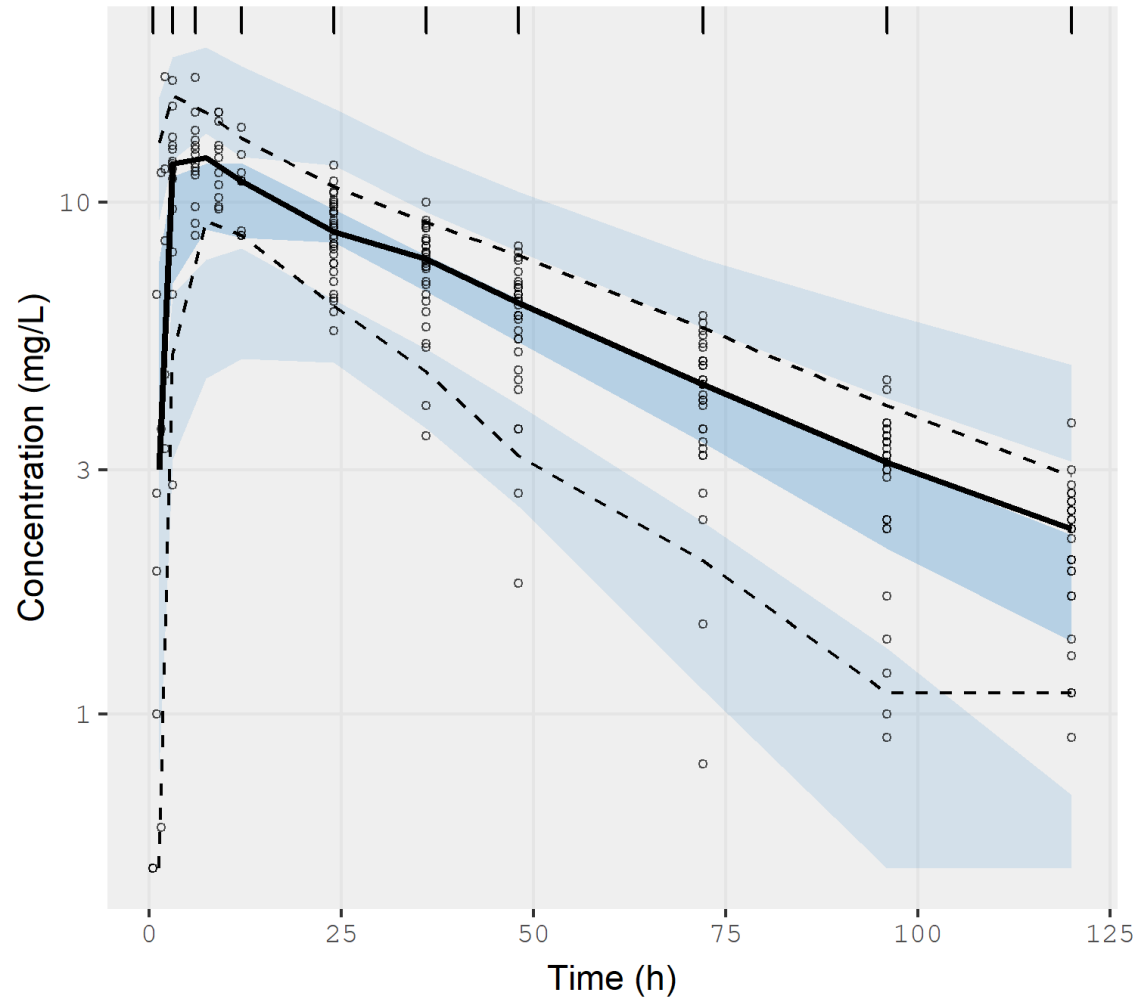
VPC for the base model on linear scale...

VPC for first order absorption PopPK model



...and on log scale. It's super fast 😊

VPC for first order absorption PopPK model
with log y-axis



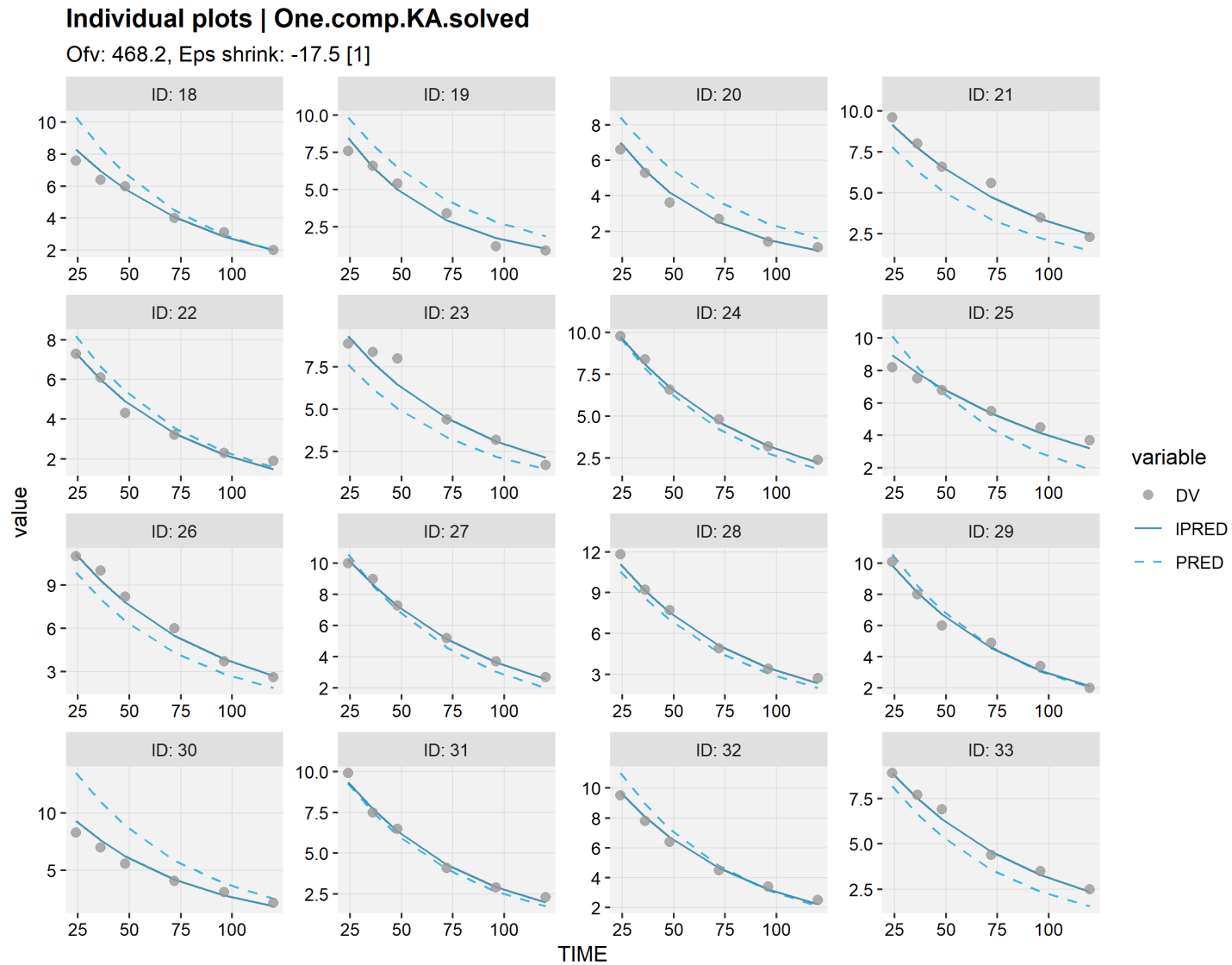
```
## or with a log y-axis starting at 0.5
vpc_ui(
  fitOne.comp.KA.solved_S,
  n = 500,
  show = list(obs_dv = TRUE),
  xlab = "Time (h)",
  ylab = "Concentration (mg/L)",
  title = "VPC for first order absorption PopPK model with log
y-axis",
  log_y = TRUE,
  log_y_min = 0.5
)
```

*#to request a log y-axis
#starting at 0.5*

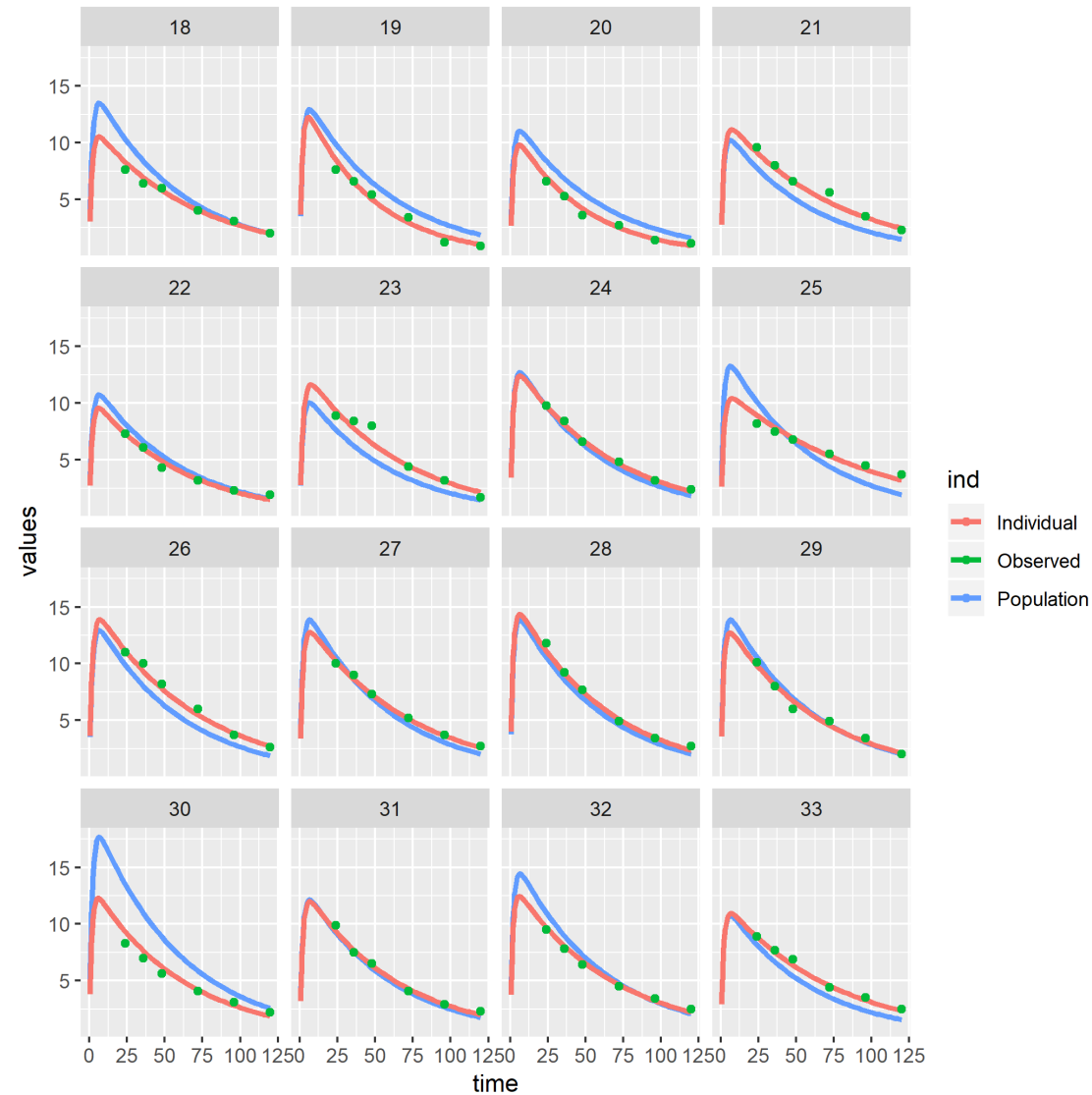
nlmixr can generate individual graphs using xpose or augPred

```
## Individual fits can be generated using using xpose:  
ind_plots(xpdb.1s,caption = NULL,ncol = 4,nrow = 4)  
## ...use the arrows in the plot window to examine the earlier curves  
  
## Individual fits can also be generated using augPred (augmented predictions)  
## that provides smooth profiles by interpolating the predictions between observations:  
plot(augPred(fitOne.comp.KA.solved_S))  
## ...use the arrows in the plot window to examine the earlier curves
```


Individual fits can be generated using xpose



Individual fits can also be generated using **augPred** (augmented predictions) that provides smooth profiles by interpolating the predictions between observations



use augPred output to plot using your favourite package...

#or the augPred output can be plotted to your liking, for instance using ggplot2 or the lattice function xyplot:

```
indivpk<-augPred(fitOne.comp.KA.solved_S)
```

```
nlmixCOLS <- c("#28466A", "#8DB6CD", "#B40000") ## specify array of colours for curves
```

```
xyplot(
```

```
  values~time|id,      ## plot the variable values by time and make a separate panel for each id  
  data=indivpk,        ## data source with smooth interpolated predictions and observations  
  groups=ind,          ## make separate curves by ind that separates Observed data,  
                      ## Individual predictions and Population predictions
```

```
  layout=c(8,4),      ## arrange as 8 columns and 4 rows  
  type=c("l","l","p"), ## represent these three by a line, a line and only markers (l=line, p=points)
```

```
  col=nlmixCOLS[c(2,1,3)], ## colours for each curve  
  cex=c(0.1,0.1,1),      ## character size for the markers
```

```
  lwd=c(2,2,0.1),       ## line width of the lines  
  pch=19,               ## use closed circles as marker
```

```
  xlab="Time (hr)\n",    ## x-axis label
```

```
  ylab="Warfarin (mg/L)", ## y-axis label
```

```
  as.table=TRUE,        ## have the first plot at the top left (otherwise plot 1 starts at the lower left corner)
```

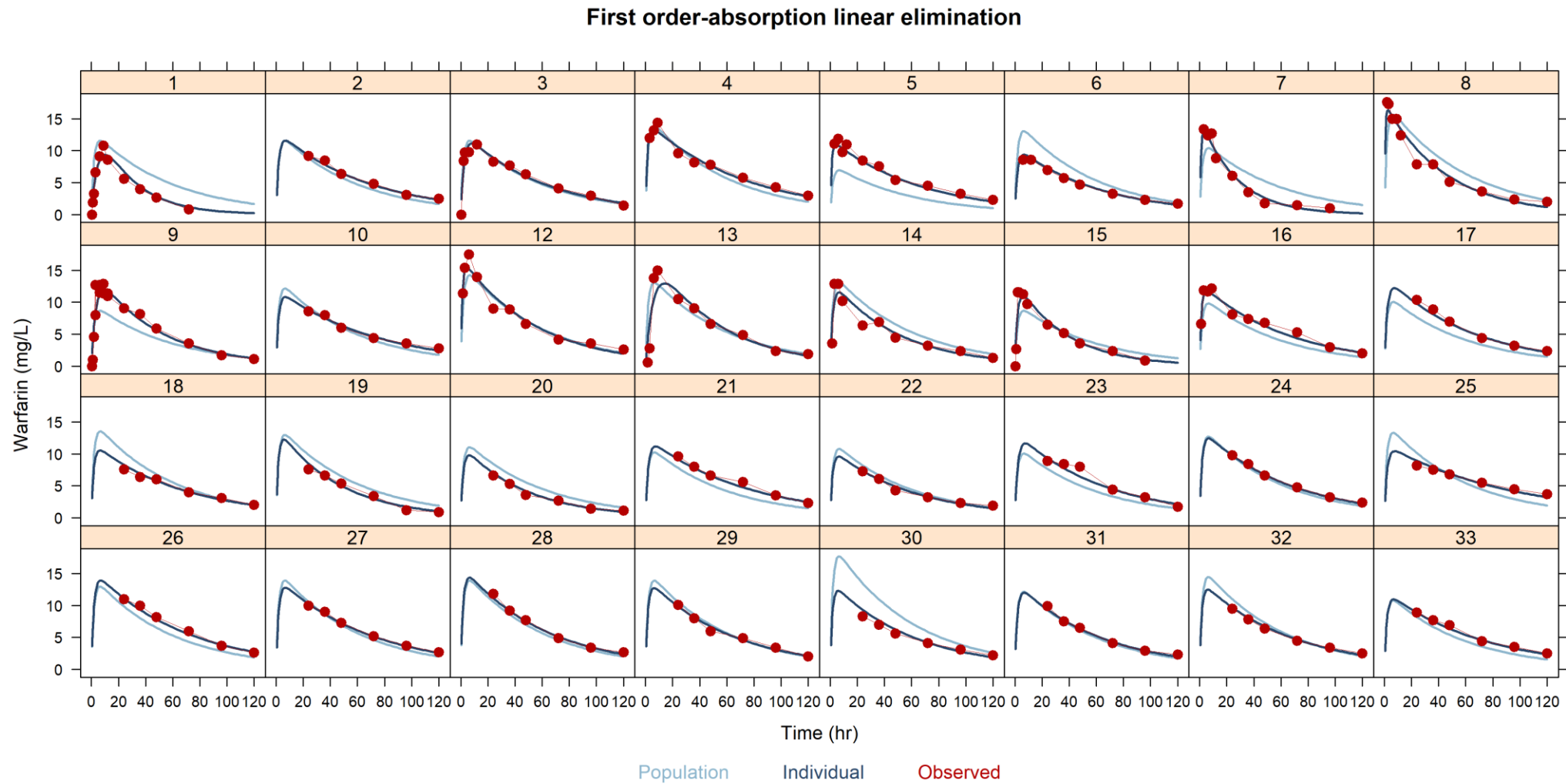
```
  scales=list(alternating=1), ## have axis labels at left and bottom (and not alternating)
```

```
  main="First order-absorption linear elimination", ## title for plot
```

```
  auto.key=list(adj=1,col=nlmixCOLS[c(2,1,3)],columns=3,space="bottom",rectangles=FALSE,points=FALSE) ## key for curves
```

```
)
```

..like lattice



Hands-on session II: running **nlmixr** and diagnostics

- Examine the code in PAWS_2.R to run a pre-programmed SAEM analysis with a solved system and its diagnostics
- Stop at `nlmixr analysis Part 2`

Solved systems and ODEs...

- Using solved-system code:

```
linCmt() ~ prop(prop.err) + add(add.err)
```

- For a solved system, model structure is automatically derived (!) from the parameter names in the **ini** block

- Using ODEs:

```
# RxODE-style differential equation definition  
d/dt(gut)      = -ka * gut  
d/dt(central) = ka * gut - (cl / v) * central  
## Concentration is calculated  
cp = central / v  
# And is assumed to follow proportional and additive error  
cp ~ prop(prop.err) + add(add.err)
```

- ODEs are much more flexible but also more time-consuming
- Solved systems are currently only available for SAEM and nlme, but FOCEI will follow soon; ODEs are available for all estimation routines

Running **nlmixr** for a system of ODEs using FOCEI

```
PK001 <- function() {  
  ini({  
    # Where initial conditions/variables are specified  
    lka <- log(1.15) #log ka (/h)  
    lcl <- log(0.135) #log CL (L/hr)  
    lv <- log(8) #log V (L)  
    prop.err <- 0.15 #proportional error (SD/mean)  
    add.err <- 0.6 #additive error (mg/L)  
    eta.ka ~ 0.5 #IIV ka  
    eta.cl ~ 0.1 #IIV cl  
    eta.v ~ 0.1 #IIV v  
  })  
  model({  
    # Where the model is specified  
    cl <- exp(lcl + eta.cl)  
    v <- exp(lv + eta.v)  
    ka <- exp(lka + eta.ka)  
    # RxODE-style differential equation definition  
    d/dt(gut) = -ka * gut  
    d/dt(center) = ka * gut - (cl / v) * center  
    ## Concentration is calculated  
    cp = center / v  
    ## And is assumed to follow proportional and additive error  
    cp ~ prop(prop.err) + add(add.err)  
  })  
}
```

```
fitPK001_F <- nlmixr(PK001, NMdata, est = "focei")
```

nlmixr output for FOCEI with ODEs

```
> fitOne.comp.KA.ODE_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 426.6131 903.9202 932.1238      -443.9601      16.08653

-- Time (sec; fitOne.comp.KA.ODE_F$time): -----
      setup optimize covariance table other
elapsed 16.44      2.083      2.083 0.03 4.364

-- Population Parameters (fitOne.comp.KA.ODE_F$parFixed or fitOne.comp.KA.ODE_F$
Parameter Est. SE %RSE Back-transformed(95%CI) BSV(CV%) shrink(SD)%
lka log ka (1/h) -0.229 0.116 50.5 0.795 (0.634, 0.998) 108. 48.3%
lcl log cl (L/h) -2.02 0.0806 3.99 0.133 (0.113, 0.156) 28.7 3.82%
lv log v (L) 2.05 0.0299 1.46 7.78 (7.33, 8.24) 21.9 14.7%
prop.err 0.137 0.137
add.err 0.622 0.622

Covariance Type (fitOne.comp.KA.ODE_F$covMethod): r,s
Fixed parameter correlations in fitOne.comp.KA.ODE_F$cor
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitOne.comp.KA.ODE_F$omega) or correlation (fitOne.comp.KA.ODE_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitOne.comp.KA.ODE_F$shrink
Minimization message (fitOne.comp.KA.ODE_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outerOpt="bobyqa"))

-- Fit Data (object fitOne.comp.KA.ODE_F is a modified tibble): -----
# A tibble: 251 x 22
  ID TIME DV EVID PRED RES WRES IPRED IRES IWRES CPRED CRES CWRES eta.ka eta.cl eta.v cl v ka cp depot central
  <fct> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1 0.5 0 0 4.20 -4.20 -2.59 1.29 -1.29 -1.99 2.90 -2.90 -2.27 -1.35 0.724 -0.0334 0.274 7.52 0.205 1.29 90.2 9.67
2 1 1 1.9 0 6.99 -5.09 -2.71 2.42 -0.522 -0.742 5.33 -3.43 -1.63 -1.35 0.724 -0.0334 0.274 7.52 0.205 2.42 81.4 18.2
3 1 2 3.3 0 10.0 -6.72 -2.71 4.31 -1.01 -1.18 8.99 -5.69 -1.73 -1.35 0.724 -0.0334 0.274 7.52 0.205 4.31 66.3 32.4
# ... with 248 more rows
```


Hands-on session III: running **nlmixr** with ODEs and perform model development

- Start at `nlmixr analysis Part 2`
- Examine the code in `PAWS_2.R` to run a pre-programmed FOCEI analysis with ODEs
- Examine the goodness of fit plots and implement alternative models for absorption (like one or more transit compartments, lag-time...)

Running **nlmixr**: 1 transit compartment

```
## 5 transit compartments
One.comp.transit <- function() {
  ini({
    # Where initial conditions/variables are specified
    lktr <- log(1.15) #log transit rate constant (/h)
    lcl <- log(0.135) #log CL (L/h)
    lv <- log(8) #log V (L)
    prop.err <- 0.15 #proportional error (SD/mean)
    add.err <- 0.6 #additive error (mg/L)
    eta.ktr ~ 0.5 #IIV ktr
    eta.cl ~ 0.1 #IIV cl
    eta.v ~ 0.1 #IIV v
  })
  model({
    # Where the model is specified
    ktr <- exp(lktr + eta.ktr)
    cl <- exp(lcl + eta.cl)
    v <- exp(lv + eta.v)
    ## ODE example
    d/dt(depot) = -ktr*depot
    d/dt(central) = ktr*trans - (cl/v)*central
    d/dt(trans) = ktr*(depot - trans)
    ## where residual error is assumed to follow proportional and additive error
    central ~ prop(prop.err) + add(add.err)
  })
}
```

nlmixr output: 1 transit compartment with ODEs using FOCEI

```
> fitOne.comp.transit_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 321.1425 798.4497 826.6533      -391.2248          66.68887

-- Time (sec; fitOne.comp.transit_F$time): -----
      setup optimize covariance table other
elapsed 18.346    2.923    2.923  0.02 3.588

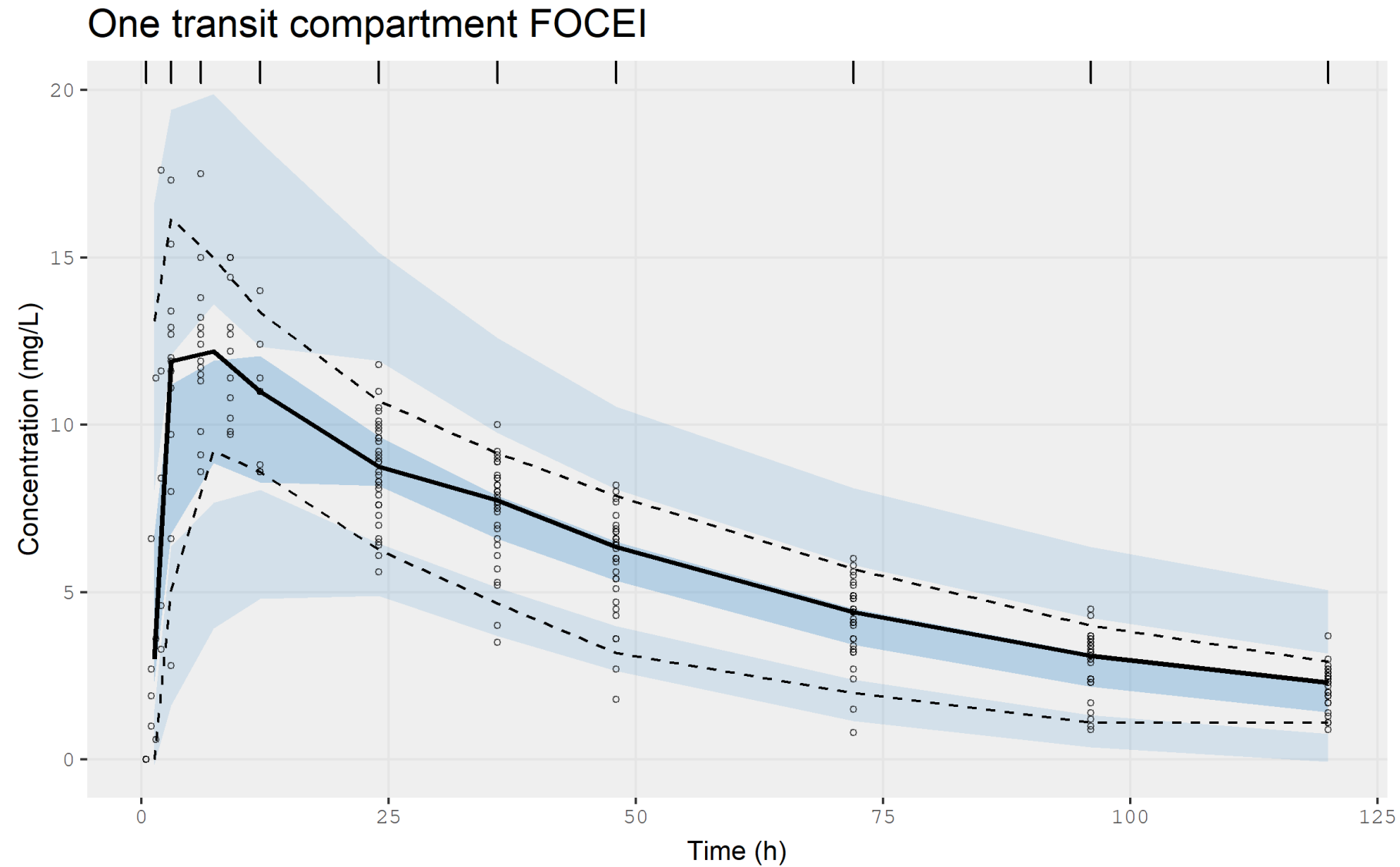
-- Population Parameters (fitOne.comp.transit_F$parFixed or fitOne.comp.transit_
Registered S3 method overwritten by 'R.oo':
  method      from
throw.default R.methodsS3

Parameter      Est.      SE %RSE Back-transformed(95%CI) BSV(CV%) shrink(SD)%
lktr           log k transit (/h) 0.0796 0.0441 55.4      1.08 (0.993, 1.18) 77.7      52.4%
lcl            log cl (L/hr) -2.02 0.233 11.5      0.132 (0.0837, 0.209) 29.4      1.52%
lv             log v (L) 2.06 0.0844 4.09      7.88 (6.68, 9.29) 21.6      3.75%
prop.err       proportional error (SD/mean) 0.0985      0.0985
add.err        additive error (mg/L) 0.449      0.449

Covariance Type (fitOne.comp.transit_F$covMethod): r,s
Some strong fixed parameter correlations exist (fitOne.comp.transit_F$cor) :
  cor:lcl,lktr cor:lv,lktr cor:lv,lcl
      -0.721      0.640      -0.875

No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitOne.comp.transit_F$omega) or correlation (fitOne.comp.transit_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitOne.comp.transit_F$shrink
Minimization message (fitOne.comp.transit_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outerOpt="bobyqa"))
```

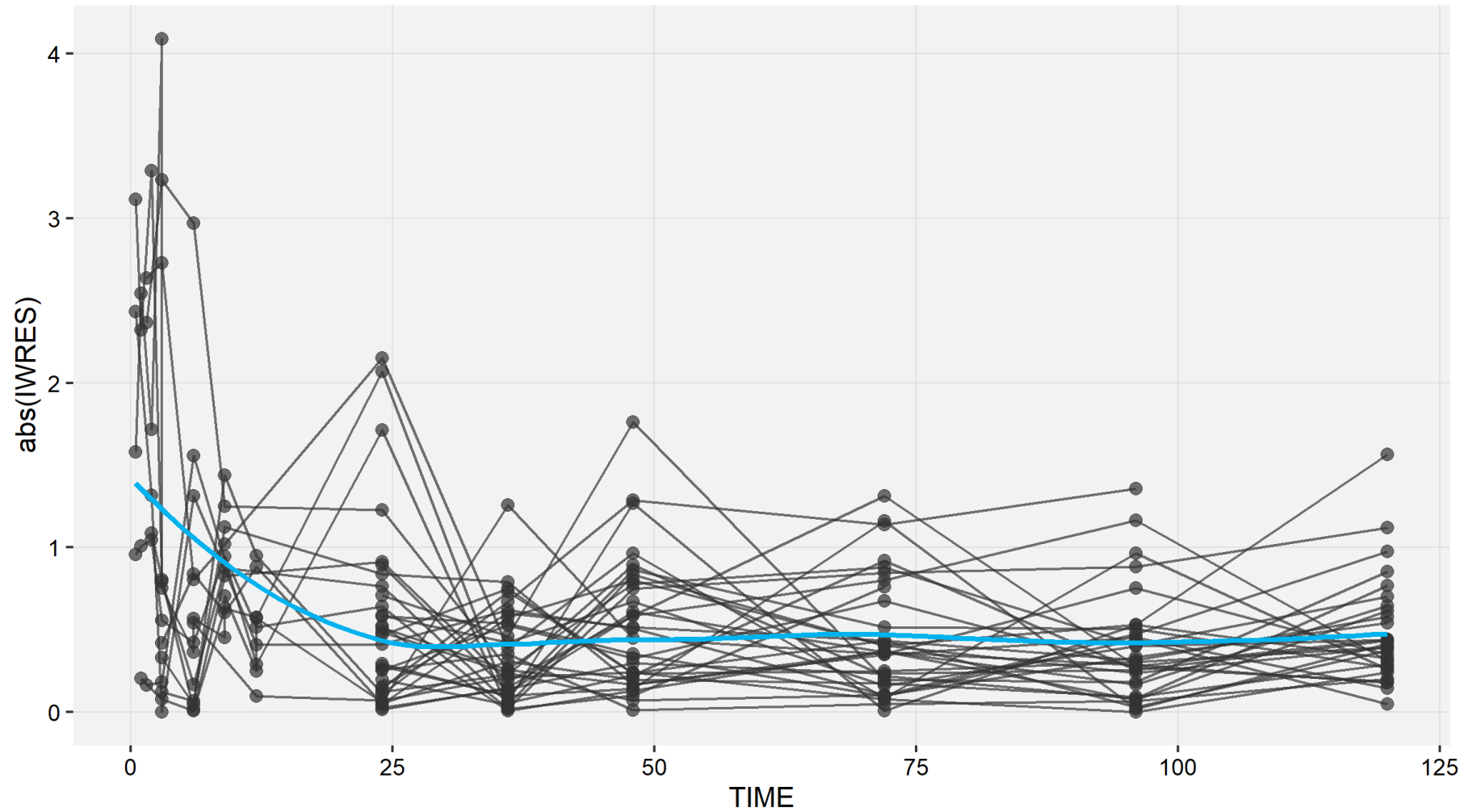
VPC for one compartment model with a transit compartment using FOCEI



| IWRES | by time for first order absorption model with 1 transit compartment using FOCEI

abs(IWRES) vs. TIME | One.comp.transit

Ofv: 321.1



Running `nlmixr`: 5 transit compartments

```
## 5 transit compartments
KA1tr5ode <- function() {
  ini({
    # Where initial conditions/variables are specified
    lktr <- log(1.15) #log transit rate constant (/h)
    lcl <- log(0.135) #log CL (L/h)
    lv <- log(8)      #log V (L)
    prop.err <- 0.15   #proportional error (SD/mean)
    add.err <- 0.6     #additive error (mg/L)
    eta.ktr ~ 0.5      #IIV ktr
    eta.cl ~ 0.1        #IIV cl
    eta.v ~ 0.1         #IIV v
  })
  model({
    # Where the model is specified
    ktr <- exp(lktr + eta.ktr)
    cl <- exp(lcl + eta.cl)
    v <- exp(lv + eta.v)
    ## ODE example
    d/dt(depot) = -ktr*depot
    d/dt(central) = ktr*transit5 - cl* central/v
    d/dt(transit1) = ktr*(depot - transit1)
    d/dt(transit2) = ktr*(transit1 - transit2)
    d/dt(transit3) = ktr*(transit2 - transit3)
    d/dt(transit4) = ktr*(transit3 - transit4)
    d/dt(transit5) = ktr*(transit4 - transit5)
    ## where residual error is assumed to follow proportional and additive error
    central ~ prop(prop.err) + add(add.err)
  })
}
```

nlmixr output: 5 transit compartments with ODEs using SAEM

```
> fitKA1tr5ode_s
-- nlmixr SAEM(ODE); OBJF by FOCEi approximation fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 270.5943 747.9015 776.1051      -365.9507      16.00938

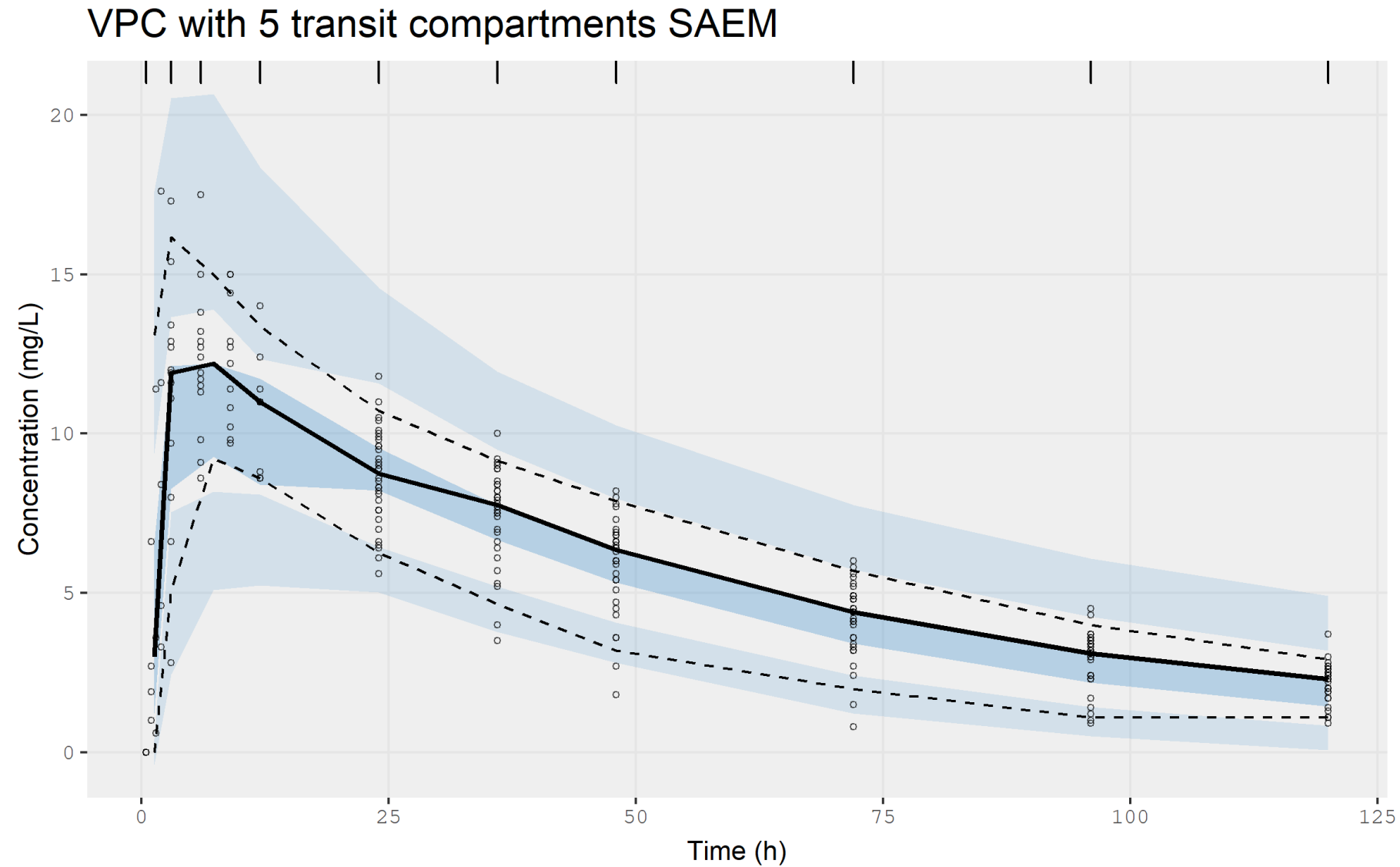
-- Time (sec; fitKA1tr5ode_s$time): -----
      saem  setup table cwres covariance other
elapsed 17.65 25.356 0.04 25.4      0.04 0.494

-- Population Parameters (fitKA1tr5ode_s$parFixed or fitKA1tr5ode_s$parFixedDF):
      Parameter      Est.      SE %RSE Back-transformed(95%CI) BSV(CV%) shrink(SD)%
lktr      log transit rate constant (/h) 1.32 0.164 12.4      3.76 (2.72, 5.18) 51.1 41.1%
lcl      log cl (L/h) -2.03 0.0522 2.58      0.132 (0.119, 0.146) 29.5 0.594%
lv      log v (L) 2.07 0.0412 1.98      7.96 (7.34, 8.63) 22.2 4.34%
prop.err      0.0495      0.0495
add.err      additive error (mg) 0.335      0.335

Covariance Type (fitKA1tr5ode_s$covMethod): linFim
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKA1tr5ode_s$omega) or correlation (fitKA1tr5ode_s$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKA1tr5ode_s$shrink

-- Fit Data (object fitKA1tr5ode_s is a modified tibble): -----
# A tibble: 251 x 27
  ID TIME DV EVID PRED RES WRES IPRED IRES IWRES CPRED CRES CWRES eta.ktr eta.cl eta.v ktr cl v cc depot
  <fct> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 1 0.5 0 0 0.158 -0.158 -0.101 0.0127 -0.0127 -0.0379 0.0464 -0.0464 -0.138 -0.509 0.736 0.0872 2.26 0.275 8.68 0.0127 32.3
2 1 1 1.9 0 2.23 -0.329 -0.211 0.318 1.58 4.72 1.02 0.880 1.22 -0.509 0.736 0.0872 2.26 0.275 8.68 0.318 10.5
3 1 2 3.3 0 9.44 -6.14 -3.93 3.40 -0.1000 -0.267 8.18 -4.88 -1.14 -0.509 0.736 0.0872 2.26 0.275 8.68 3.40 1.09
# ... with 248 more rows, and 6 more variables: central <dbl>, transit1 <dbl>, transit2 <dbl>, transit3 <dbl>, transit4 <dbl>, transit5 <dbl>
```

VPC for first order absorption with 5 transit compartments using SAEM



nlmixr output: 5 transit compartments with ODEs using FOCEI

Change in OFV compared to model with 1 transit compartment: -90.82

```
> fitKA1tr5ode_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 230.3234 707.6306 735.8342      -345.8153      4288.053

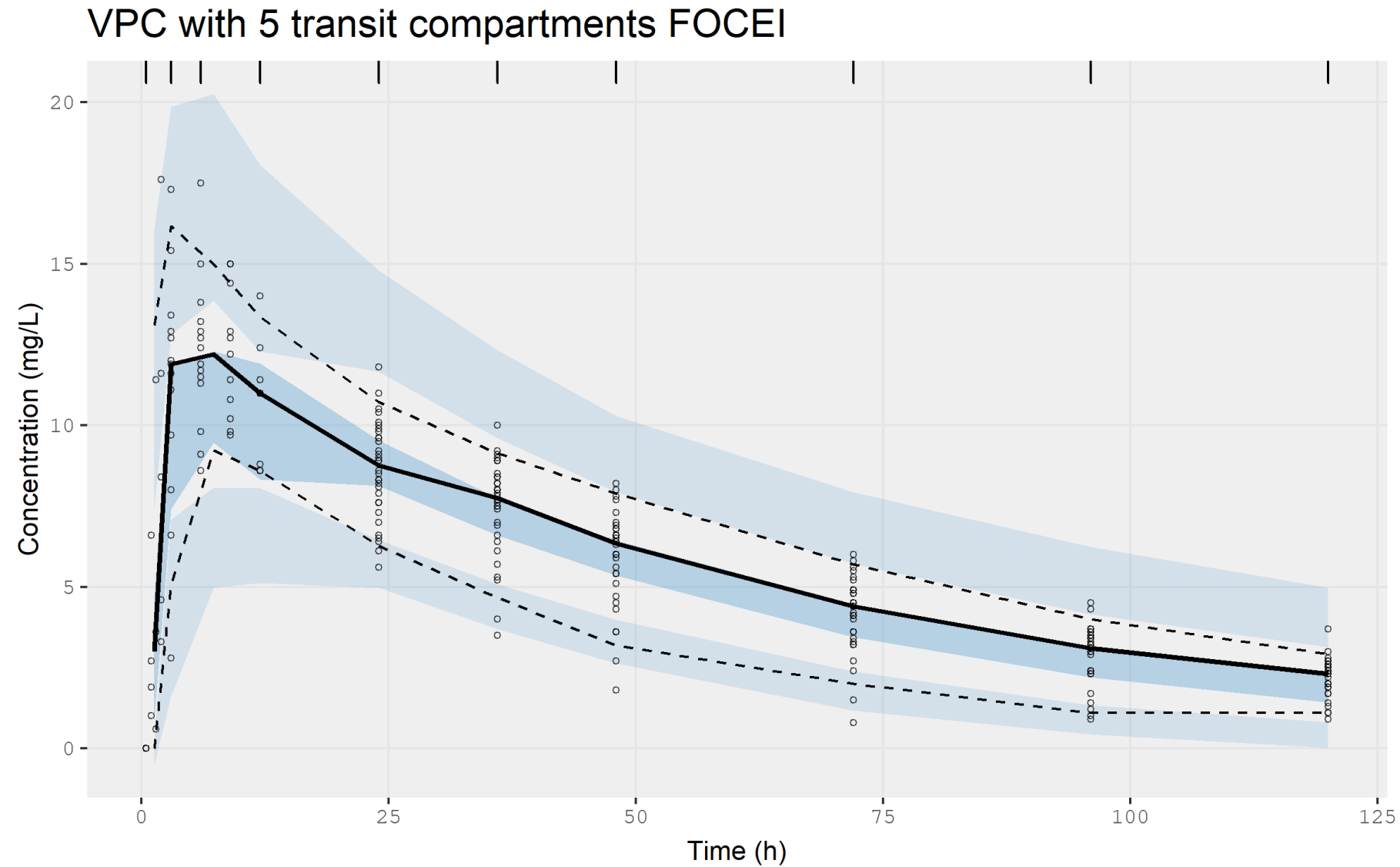
-- Time (sec; fitKA1tr5ode_F$time): -----
      setup optimize covariance table  other
elapsed 0.826    5.328    5.328  0.03 17.178

-- Population Parameters (fitKA1tr5ode_F$parFixed or fitKA1tr5ode_F$parFixedDf):
      Parameter      Est.      SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr    log transit rate constant (/h)  1.17  0.711  61      3.21 (0.796, 12.9)    47.6    44.1%
lcl      log cl (L/h) -2.02 0.0858 4.25    0.133 (0.112, 0.157)    29.7    0.304%
lv       log V (L)  2.08 0.0367 1.77     7.97 (7.41, 8.56)    22.3    4.29%
prop.err                                0.0765                                0.0765
add.err      additive error (mg)  0.374                                0.374

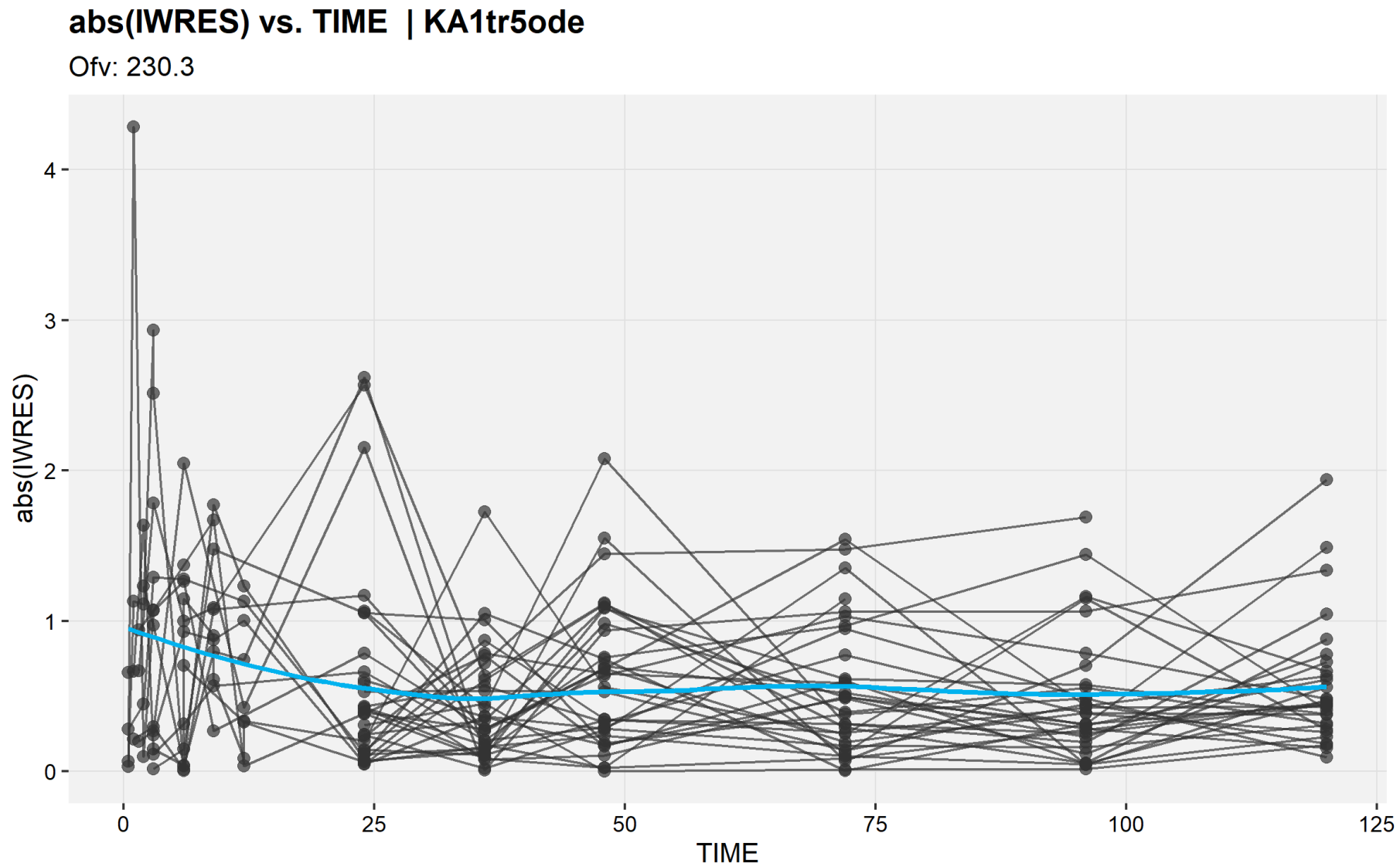
Covariance Type (fitKA1tr5ode_F$covMethod): r,s
Some strong fixed parameter correlations exist (fitKA1tr5ode_F$cor) :
      cor:lcl,lktr cor:lv,lktr cor:lv,lcl
      0.986      0.522      0.598

No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKA1tr5ode_F$omega) or correlation (fitKA1tr5ode_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKA1tr5ode_F$shrink
Minimization message (fitKA1tr5ode_F$message):
      false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
      nlmixr(...,control=list(outerOpt="bobyqa"))
```

VPC for first order absorption with 5 transit compartments using FOCEI

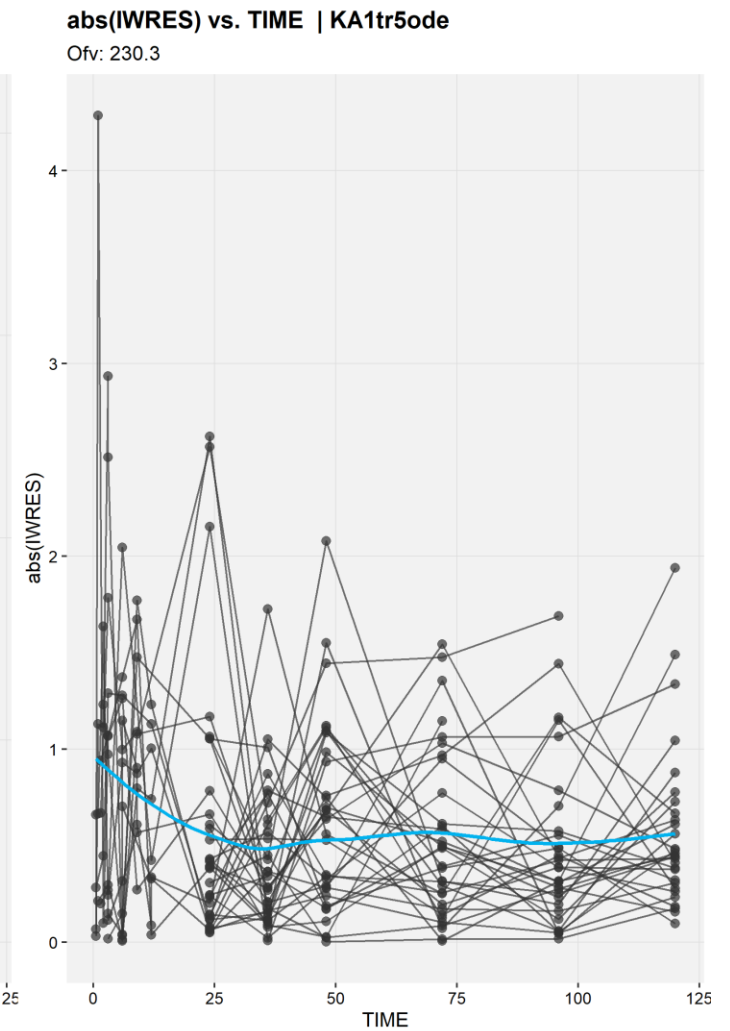
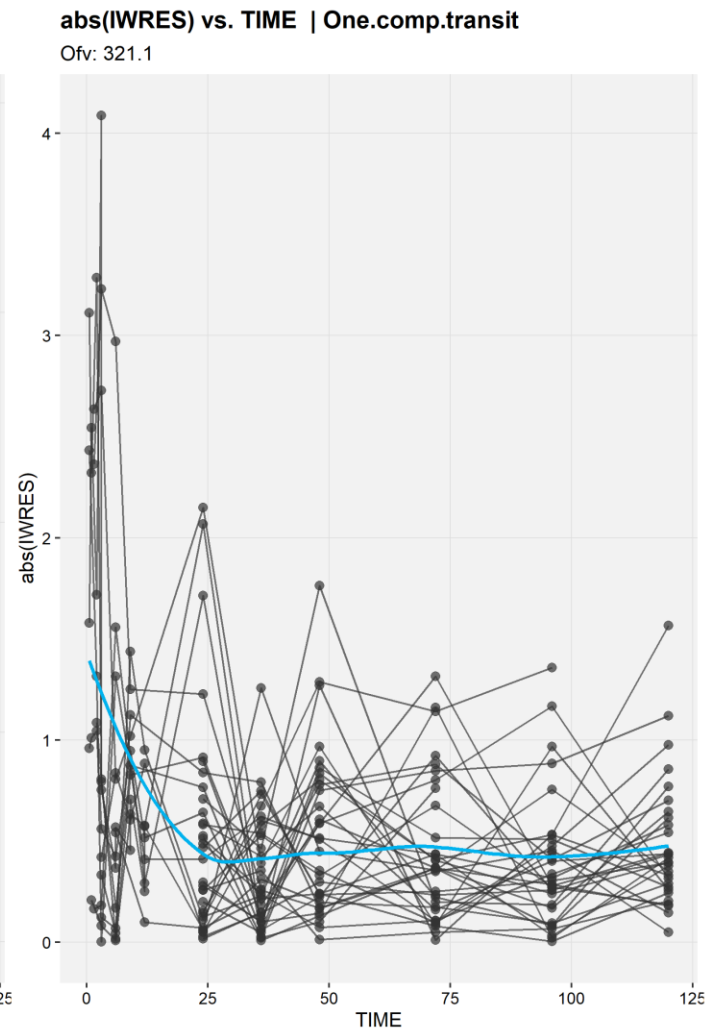
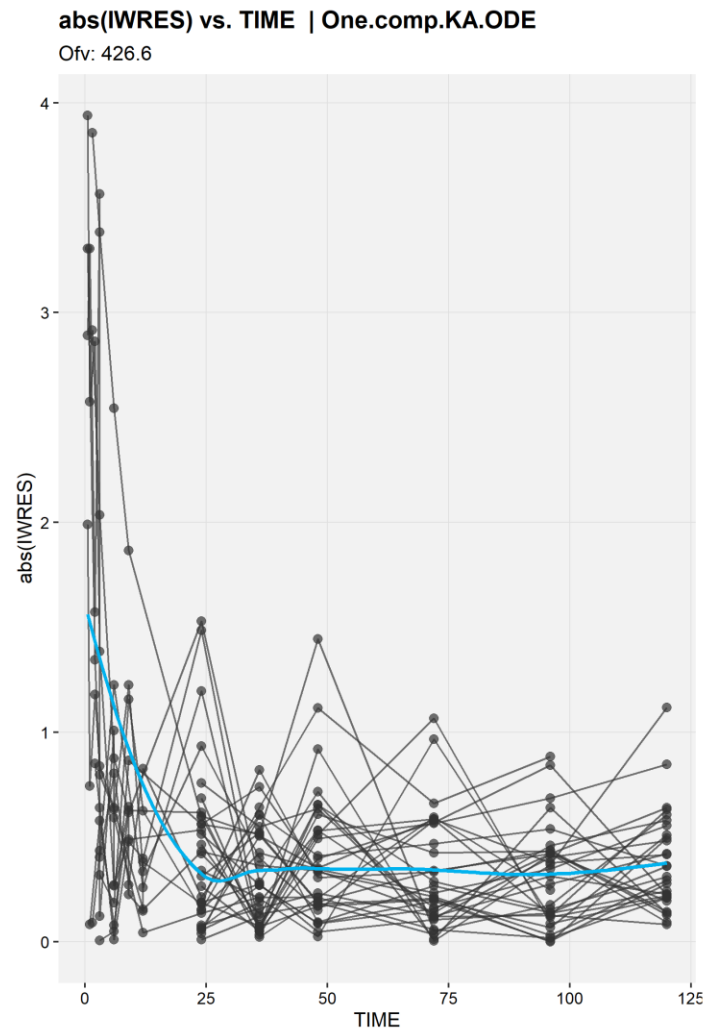


| IWRES | by time for first order absorption model with 5 transit compartments using FOCEI



Comparison of GOF plots for different absorption models

No transit compartment (left), one transit compartment (middle), 5 transit compartments (right)



Objective function values and estimation algorithms

- When you request `cwres=TRUE` for an SAEM analysis, `nlmixr` calculates an FOCEI-type objective function value, and FOCEI-type conditional weighted residuals
- However, this does not mean that they can be formally (or even informally) compared: as with NONMEM, comparisons should only ever be performed with nested models using the same estimation algorithm
- Differences in OFV are not an indication of superiority of estimates obtained from one algorithm over another

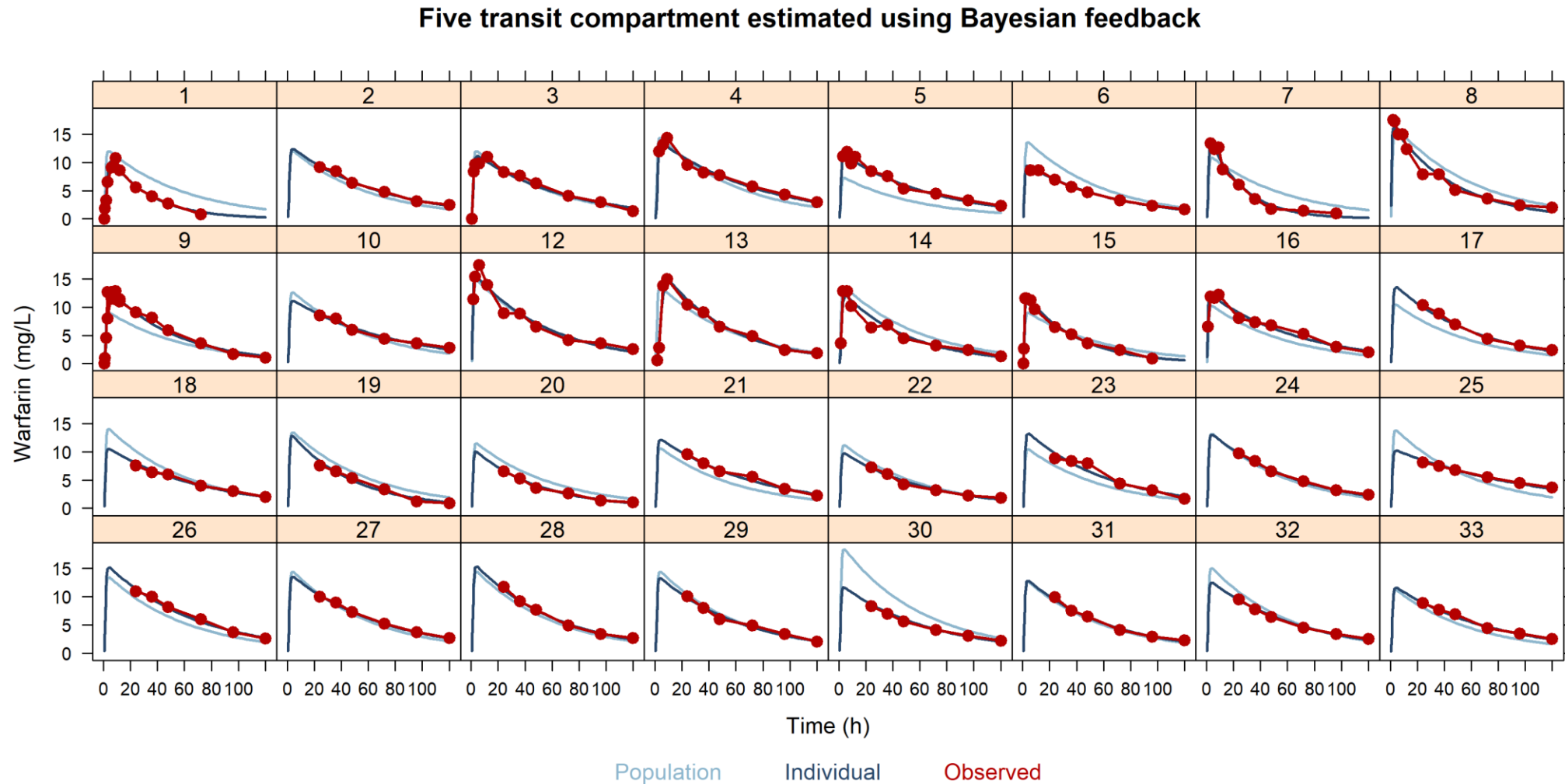
nlmixr can generate empirical Bayes estimates for Bayesian feedback: individual EBEs for a new data set using existing population parameters

```
KA1tr5posthoc <- function() {  
  ini({  
    # Specify previously obtained population estimates (e.g. from NONMEM or nlmixr)  
    lktr <- 1.18994619 #Log ktr (/h)  
    lcl <- -2.01737477 #Log CL (L/h)  
    lv <- 2.06631620 #Log V (L)  
    prop.err <- 0.07883633 #proportional error (SD/mean)  
    add.err <- 0.37249666 #additive error (mg/L)  
    eta.ktr ~ 0.2532964 #IIV ktr  
    eta.cl ~ 0.08073339 #IIV CL  
    eta.v ~ 0.04490733 #IIV V  
  })  
  model({  
    cl <- exp(lcl + eta.cl)  
    v <- exp(lv + eta.v)  
    ktr <- exp(lktr + eta.ktr)  
    d/dt(trns1) = -ktr * trns1  
    d/dt(trns2) = ktr * trns1 - ktr * trns2  
    d/dt(trns3) = ktr * trns2 - ktr * trns3  
    d/dt(trns4) = ktr * trns3 - ktr * trns4  
    d/dt(trns5) = ktr * trns4 - ktr * trns5  
    d/dt(central) = ktr * trns5 - (cl/v) * central  
    cp = central/v  
    cp ~ prop(prop.err) + add(add.err)  
  })  
}
```

```
fitKA1tr5_Fph <- nlmixr(KA1tr5posthoc, PKdata,  
  est = "posthoc") # Specify posthoc as estimation method
```

- Useful in a therapeutic drug monitoring setting
- Or for generating exposure estimates with a particularly nasty model that you do not want to refit on new data 😊

Individual graphs for the five transit compartment model estimated using Bayesian feedback; perfect fit even though there was no actual parameter estimation



Parameterisation and mu-referencing

- For SAEM, parameters must be defined using 'mu-referencing'
- This means that inter-individual variability parameters must be added onto population parameters
- This implies estimating log-parameters with the IIV added on the log-scale
- For FOCEI, mu-referencing is not strictly required, but is shown to provide superior estimation results
- For a binary covariate (e.g. sex 0/1), the back-transformed estimate is a fold-change that can be re-written as a percentage change

Corresponding **nlmixr** code

```
## One compartment transit model with Sex on V
KAtr1_sexV <- function() {
  ini({
    lktr <- log(1.15) #Log k transit (/h)
    lc1 <- log(0.135) #Log CL (L/h)
    lv <- log(8) #Log V (L)
    Sex_V <- 0.1 #Log Sex on v
    prop.err <- 0.15 #proportional error (SD/mean)
    add.err <- 0.6 #additive error (mg/L)
    eta.ktr ~ 0.5 #IIV ktr
    eta.cl ~ 0.1 #IIV CL
    eta.v ~ 0.1 #IIV V
  })
  model({
    #Sex on volume
    cl <- exp(lc1 + eta.cl)
    v <- exp(lv + eta.v + Sex_V * SEX) #the SEX covariate is 0 or 1 in the data set
    ktr <- exp(lktr + eta.ktr)
    d/dt(depot) = -ktr * depot
    d/dt(central) = ktr * trans - (cl/v) * central
    d/dt(trans) = ktr * depot - ktr * trans
    cp = central/v
    cp ~ prop(prop.err) + add(add.err)
  })
}
```

nlmixr output: mu-referenced sex on V (log-scale)

```
> fitKatr1_sexV_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 303.2938 782.601 814.33      -382.3005          55.65776

-- Time (sec; fitKatr1_sexV_F$time): -----
      setup optimize covariance table  other
elapsed 17.224      4.46      4.46  0.03 12.356

-- Population Parameters (fitKatr1_sexV_F$parFixed or fitKatr1_sexV_F$parFixedDf
Registered S3 method overwritten by 'R.oo':
  method      from
throw.default R.methodss3

      Parameter Est.      SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr      log k transit (/h) 0.148 0.332 224      1.16 (0.605, 2.22)      61.4      45.0%
lcl      log CL (L/h) -2.02 0.113 5.62      0.133 (0.107, 0.166)      29.2      1.27%
lv      log V (L) 1.74 0.233 13.4      5.7 (3.61, 8.99)      15.8      15.2%
sex_v      log Sex on v 0.394 0.244 62 0.394 (-0.0847, 0.872)
prop.err      0.101      0.101
add.err      0.47      0.47

Covariance Type (fitKatr1_sexV_F$covMethod): s
Some strong fixed parameter correlations exist (fitKatr1_sexV_F$cor) :
      cor:lcl,lktr      cor:lv,lktr cor:Sex_v,lktr      cor:lv,lcl cor:Sex_v,lcl      cor:Sex_v,lv
      0.0841      -0.139      0.0530      -0.135      0.0152      -0.950

No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKatr1_sexV_F$omega) or correlation (fitKatr1_sexV_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKatr1_sexV_F$shrink
Minimization message (fitKatr1_sexV_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outeropt="bobyqa"))
```

Parameterisation and mu-referencing

- For a binary covariate (e.g. sex 0/1; female/males), the back-transformed estimate is a fold-change that can be re-written as a percentage change
- The estimated 0.394 (95%CI: -0.0847 / 0.872) translates to a fold-change estimate of 1.482 (95%CI: 0.919/ 2.392) which corresponds to a change of 48.2% (95%CI: -8.1% / 139.2%) for males

mu-referencing and allometric scaling

- For a standard allometric equation we would use:
 - $CL_i = CL_{Pop} \cdot (WT_i/70)^{3/4} \cdot e^\eta$
- Take the log on both sides
 - $\log[CL_i] = \log[CL_{Pop}] + \log\left[(WT_i/70)^{3/4}\right] + \log[e^\eta]$
 - $\log[CL_i] = \log[CL_{Pop}] + 3/4 \cdot \log[WT_i/70] + \eta$
- And back-transforming:
 - $CL_i = e^{\log[CL_{Pop}] + 3/4 \cdot \log[WT_i/70] + \eta}$

Corresponding **nlmixr** code

- Code is most stable if transformations are carried out in the data file instead of in the model code, especially for SAEM:

Using standard R syntax:

```
PKdata$logWT70 <- log(PKdata$WT/70)
```

Or using data.table syntax

```
PKdata[,logWT70:=log(WT/70)]
```

Corresponding **nlmixr** code: allometric scaling

```
One.comp.transit.allo <- function() {  
  ini({  
    lktr <- log(1.15)  #Log k transit (/h)  
    lcl  <- log(0.135) #Log CL (L/hr)  
    lv   <- log(8)     #Log V (L)  
    ALLC <- fix(0.75)  #allometric exponent cl  
    ALLV <- fix(1.00)  #allometric exponent v  
    prop.err <- 0.15   #proportional error (SD/mean)  
    add.err <- 0.6     #additive error (mg/L)  
    eta.ktr ~ 0.5     #IIV ktr  
    eta.cl ~ 0.1      #IIV CL  
    eta.v ~ 0.1       #IIV V  
  })  
  model({  
    #Allometric scaling on weight  
    cl <- exp(lcl + eta.cl + ALLC * logWT70)  
    v  <- exp(lv + eta.v + ALLV * logWT70)  
    ktr <- exp(lktr + eta.ktr)  
    d/dt(depot) = -ktr * depot  
    d/dt(central) = ktr * trans - (cl/v) * central  
    d/dt(trans) = ktr * depot - ktr * trans  
    cp = central/v  
    cp ~ prop(prop.err) + add(add.err)  
  })  
}
```

nlmixr output: allometric scaling ODEs using FOCEI

Change in OFV compared to model without allometric scaling: -29.49

```
> fitone.comp.transit.allo_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 291.6498 768.9569 797.1605      -376.4785          38.16301

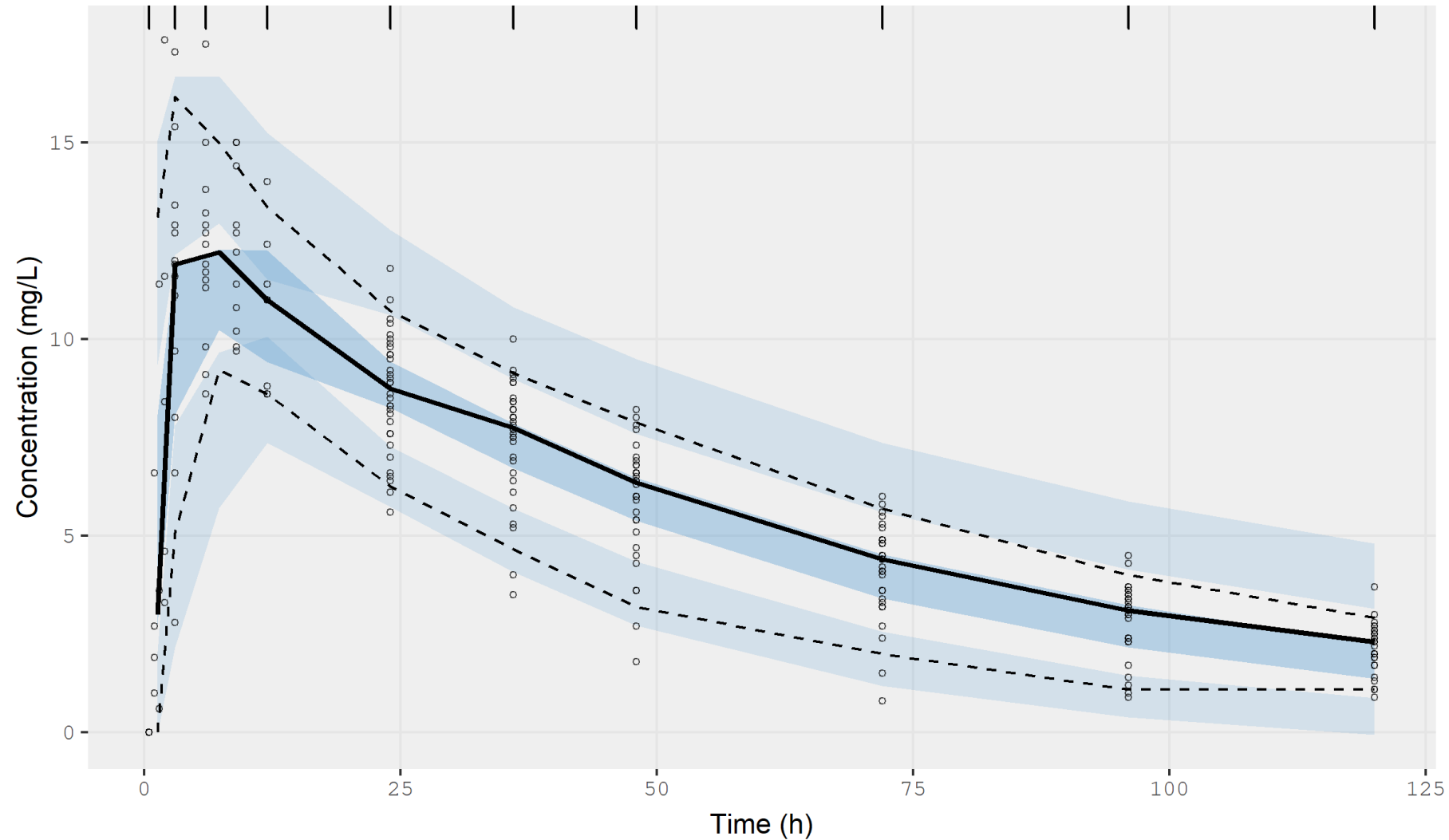
-- Time (sec; fitone.comp.transit.allo_F$time): -----
      setup optimize covariance table  other
elapsed 18.055      3.027      3.027  0.02 13.361

-- Population Parameters (fitone.comp.transit.allo_F$parFixed or fitone.comp.tra
Parameter Est. SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr      log k transit (/h) 0.159 0.381 239      1.17 (0.556, 2.47) 80.2      53.4%
lcl        log cl (L/hr) -2.01 0.124 6.2      0.134 (0.105, 0.171) 30.9      13.0%
lv          log v (L) 2.08 0.0625 3      8.02 (7.1, 9.07) 14.0      19.0%
ALLC      allometric exponent cl 0.75 FIXED FIXED      0.75
ALLV      allometric exponent v 1 FIXED FIXED      1
prop. err      0.101      0.101
add. err      0.466      0.466

Covariance Type (fitone.comp.transit.allo_F$covMethod): s
Fixed parameter correlations in fitone.comp.transit.allo_F$cor
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitone.comp.transit.allo_F$omega) or correlation (fitone.comp.transit.allo_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitone.comp.transit.allo_F$shrink
Minimization message (fitone.comp.transit.allo_F$message):
false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
nlmixr(...,control=list(outeropt="bobyqa"))
```

VPC for one compartment model with a transit compartment and allometric scaling using FOCEI

One transit compartment and allometric scaling FOCEI



nlmixr output: allometric scaling ODEs using FOCEI

Freely estimated exponents: drop of only 2.11 points

```
> fitOne.comp.transit.allo.free_F
-- nlmixr FOCEI (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEI 289.5371 770.8442 806.0987      -375.4221      126.6137

-- Time (sec; fitOne.comp.transit.allo.free_F$time): -----
      setup optimize covariance table other
elapsed 0.841      6.553      6.553 0.01 9.163

-- Population Parameters (fitOne.comp.transit.allo.free_F$parFixed or fitOne.com
Parameter Est. SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr      log k transit (/h) 0.236 0.335 142      1.27 (0.656, 2.44)      64.1      44.5%
lcl      log cl (L/hr) -2.01 0.104 5.19      0.134 (0.11, 0.165)      26.1      0.241%
lv      log v (L) 2.08 0.063 3.03      7.99 (7.07, 9.04)      12.5      17.3%
ALLC      allometric exponent cl 0.618 0.583 94.3      0.618 (-0.524, 1.76)
ALLV      allometric exponent v 0.93 0.398 42.8      0.93 (0.15, 1.71)
prop. err      0.102      0.102
add. err      0.472      0.472

Covariance Type (fitOne.comp.transit.allo.free_F$covMethod): s
Fixed parameter correlations in fitOne.comp.transit.allo.free_F$cor
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitOne.comp.transit.allo.free_F$omega) or correlation (fitOne.comp.transit.allo.free_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitOne.comp.transit.allo.free_F$shrink
Minimization message (fitOne.comp.transit.allo.free_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outerOpt="bobyqa"))
```

Hands-on session IV: running an **nlmixr** posthoc analysis and implementing covariates using mu-referencing

- Examine the code in PAWS_4.R to run a posthoc analysis
- Examine the code in PAWS_4.R to run a covariate analysis with additive on log-scale covariate implementation
 - Binary covariate (sex)
 - Continuous covariate (weight)
- Modify the allometrically scaled model with fixed exponents to examine what happens when you set them free

PKPD analysis with **nlmixr**: sequential estimation

- First approach: use EBEs from a previous PK model to define PK profiles and estimate PKPD relationship
- Extract EBEs from nlmixr object and merge to PKPD data file

```
load(file = "fitOne.comp.transit.allo_F.Rdata")
#Use the one compartment transit model to start the PD analysis
EBEs <- as.data.table(fitOne.comp.transit.allo_F)
EBEs <- EBEs[!duplicated(ID), .(ID = as.numeric(as.character(ID)),
  IKTR = ktr, ICL = cl, IV = v)]
PKPDdata <- fread("warfarin_dat.csv")

#Change variable names to upper case (not strictly necessary)
setnames(PKPDdata, names(dataF), toupper(names(dataF)))

#Generate MDV data items
PKPDdata[, MDV := ifelse(is.na(DV), 1, 0)]
PKPDdata[, MDV := ifelse(AMT > 0, 1, MDV)]

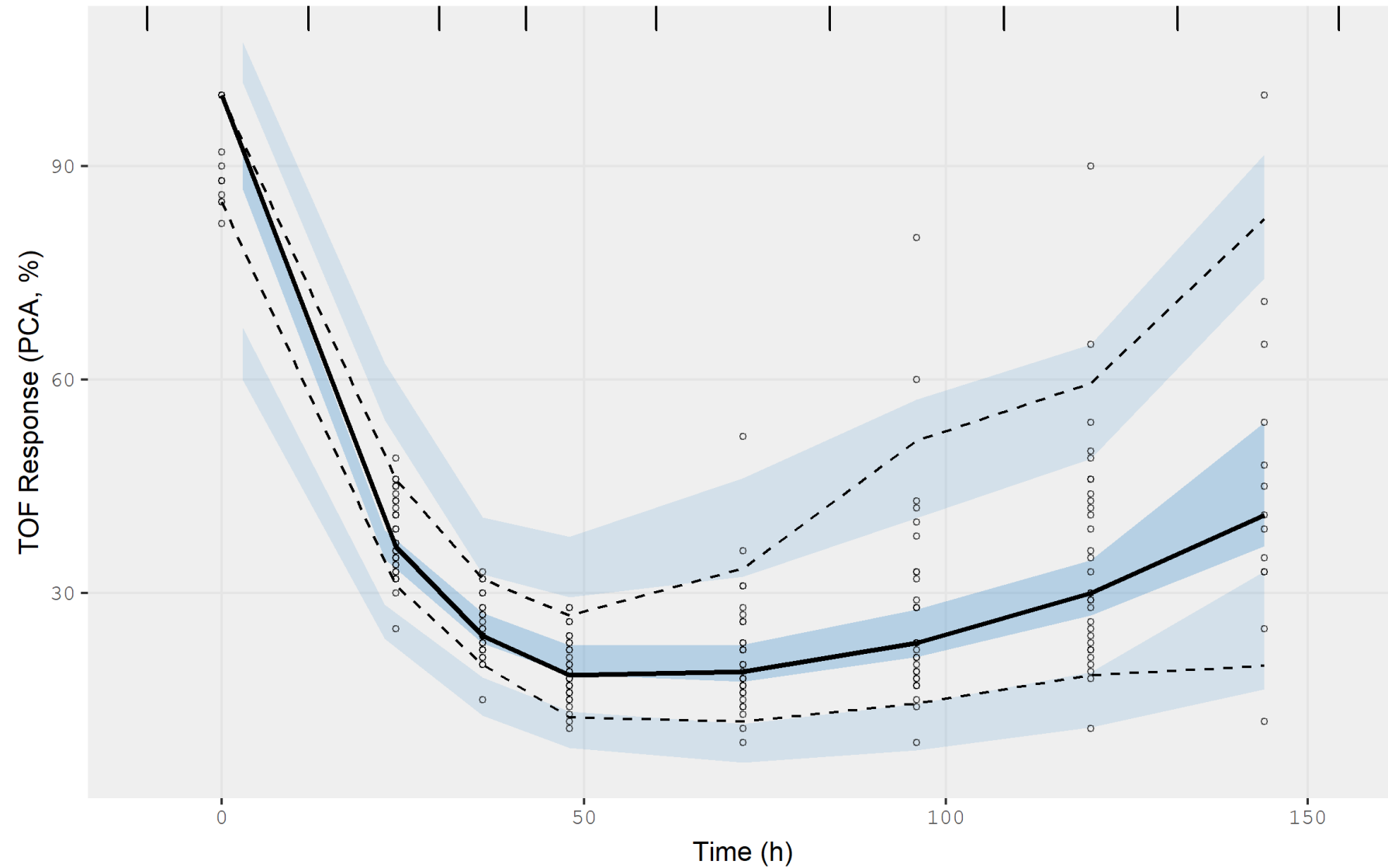
#Merge data with PK EBEs
PDdata <- merge(PKPDdata, EBEs, by = "ID", all.x = TRUE)

#Mark the PK measurements from the file to not be analysed
PDdata[, MDV := ifelse(DVID == 1 & AMT == 0, 1, MDV)]
```

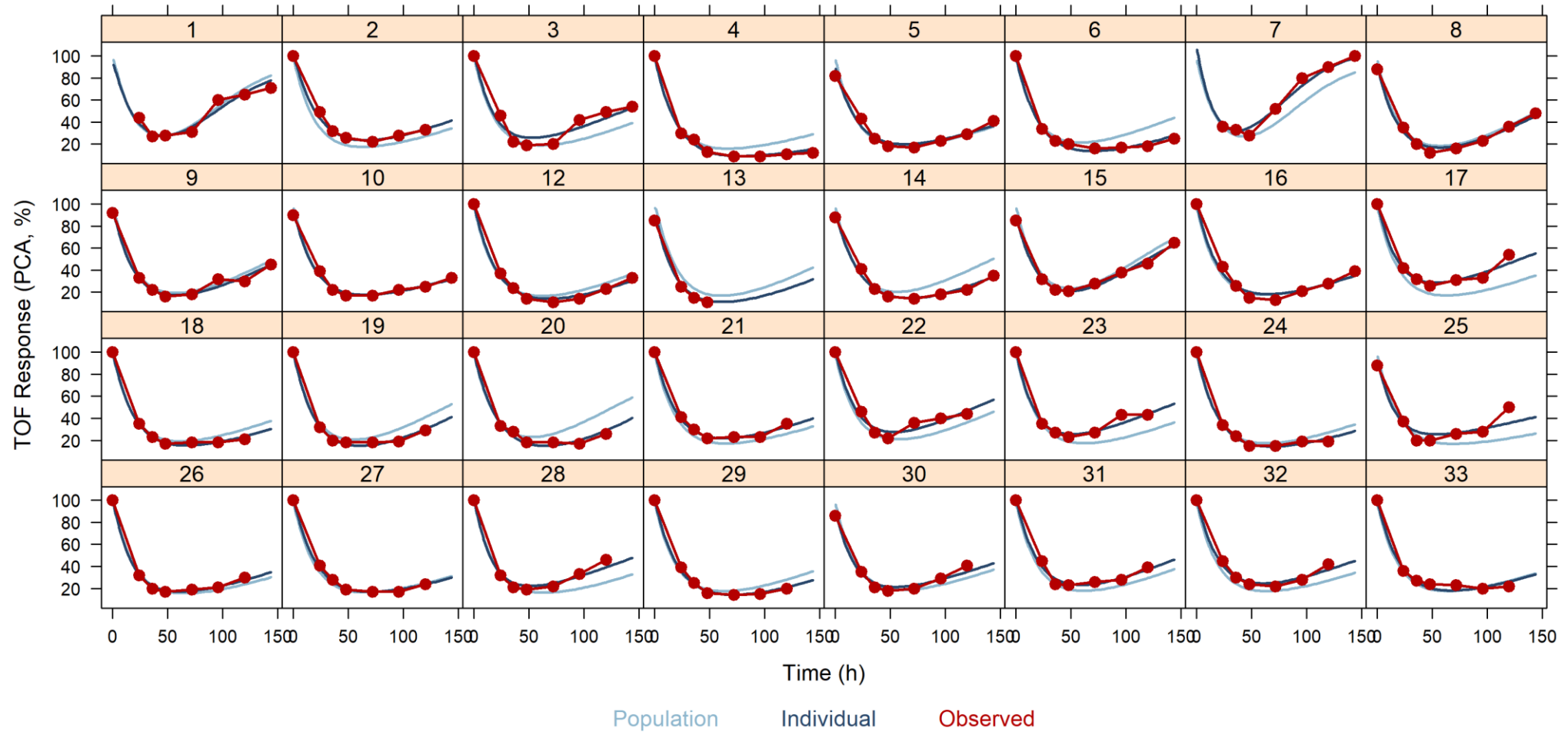
PKPD analysis with **nlmixr**: define turnover model using PK EBEs

```
KA1tr1IPP_PDtoemax1 <- function() {  
  ini({  
    tc50 <- log(1)    #Log ec50 (mg/L)  
    tkout <- log(0.05) #Log tkout (/h)  
    te0 <- log(100)   #Log e0  
    eta.c50 ~ .5  
    eta.kout ~ .1  
    eta.e0 ~ .1  
    eps.pdadd <- 100  
  })  
  model({  
    c50 = exp(tc50 + eta.c50)  
    kout = exp(tkout + eta.kout)  
    e0 = exp(te0 + eta.e0)  
    # PK parameters from input data set  
    ktr = IKTR  
    cl = ICL  
    v = IV  
    cp = central/v  
    PD = 1 - cp/(c50 + cp)  
    effect(0) = e0  
    kin = e0 * kout  
    d/dt(depot) = -ktr * depot  
    d/dt(central) = ktr * trans - cl/v * central  
    d/dt(trans) = ktr * depot - ktr * trans  
    d/dt(effect) = kin * PD - kout * effect  
    effect ~ add(eps.pdadd)  
  })  
}
```

VPC turnover Emax PD model with PK using EBEs



Individual graphs for turnover Emax PD model with PK using EBEs



Hands-on session V: running a sequential **nlmixr** PKPD analysis

- Examine the code in PAWS_5.R to generate a data file with EBEs to describe the PK-part of the PKPD model
- Run the sequential PKPD analysis

PKPD analysis with **nlmixr**: simultaneous estimation

- Second approach: estimate PK and PD simultaneously
- The source of observations is identified using either a CMT data item or a DVID data item
- CMT defines the compartment where data are observed
- DVID is coded as 1, 2 to identify the first and second type of observation as presented in the model code

Immediate effect simultaneous PKPD analysis with **nlmixr**: **ini** block

```
#Immediate effect
KA1tr1_PDimmemax1 <- function() {
  ini({
    ## PK
    tktr <- log(1)    # log ktr (/h)
    tcl  <- log(0.1)  # log CL (L/h)
    tv   <- log(8)    # log Vc (L)
    eta.ktr ~ 1
    eta.cl ~ 0.1
    eta.v ~ 0.1
    eps.pkprop <- 0.1 #proportional error (SD/mean)
    eps.pkadd <- 0.4  #additive error (mg/L)

    ## PD
    tc50 <- log(1)    #log ec50 (mg/L)
    te0  <- log(100)  #log e0
    eta.c50 ~ .5
    eta.e0 ~ .1
    eps.pdadd <- 100
  })
  model({
  })
}
```

Immediate effect simultaneous PKPD analysis with **nlmixr**: **model** block

```
#Immediate effect
model({
  ktr <- exp(tktr + eta.ktr)
  cl  <- exp(tc1 + eta.cl)
  v   <- exp(tv + eta.v)

  c50 = exp(tc50 + eta.c50)
  e0  = exp(te0 + eta.e0)

  cp          = central/v
  d/dt(depot) = -ktr * depot
  d/dt(central) = ktr * trans - cl * cp
  d/dt(trans)  = ktr * depot - ktr * trans
  effect      = e0 * (1 - cp/(c50 + cp))

  cp ~ prop(eps.pkprop) + add(eps.pkadd) | central
  effect ~ add(eps.pdadd) | effect
})
}
```

Simultaneous PKPD analysis with **nlmixr**: examine defined model

```
nlmixr(KA1tr1_PDimmemax1) # Show initial estimates and model:
```

```
-- Multiple Endpoint Model ($multipleEndpoint): -----
```

variable	cmt	dvid*
cp ~ ...	cmt='central' or cmt=2	dvid='central' or dvid=1
effect ~ ...	cmt='effect' or cmt=4	dvid='effect' or dvid=2

* If dvids are outside this range, all dvids are re-numbered sequentially, ie 1,7, 10 becomes 1,2,3 etc

```
-- mu-referencing ($muRefTable): -----
```

theta	eta
tktr	eta.ktr
tc1	eta.c1
tv	eta.v
tc50	eta.c50
te0	eta.e0

```
-- Model: -----
```

```
ktr <- exp(tktr + eta.ktr)
c1  <- exp(tc1 + eta.c1)
v   <- exp(tv + eta.v)

c50 = exp(tc50 + eta.c50)
e0  = exp(te0 + eta.e0)

cp      = central/v
d/dt(depot) = -ktr * depot
d/dt(central) = ktr * trans - c1 * cp
d/dt(trans)  = ktr * depot - ktr * trans
effect      = e0 * (1 - cp/(c50 + cp))

cp ~ prop(eps.pkprop) + add(eps.pkadd) | central
effect ~ add(eps.pdadd) | effect
```

nlmixr output: immediate effect simultaneous PKPD model

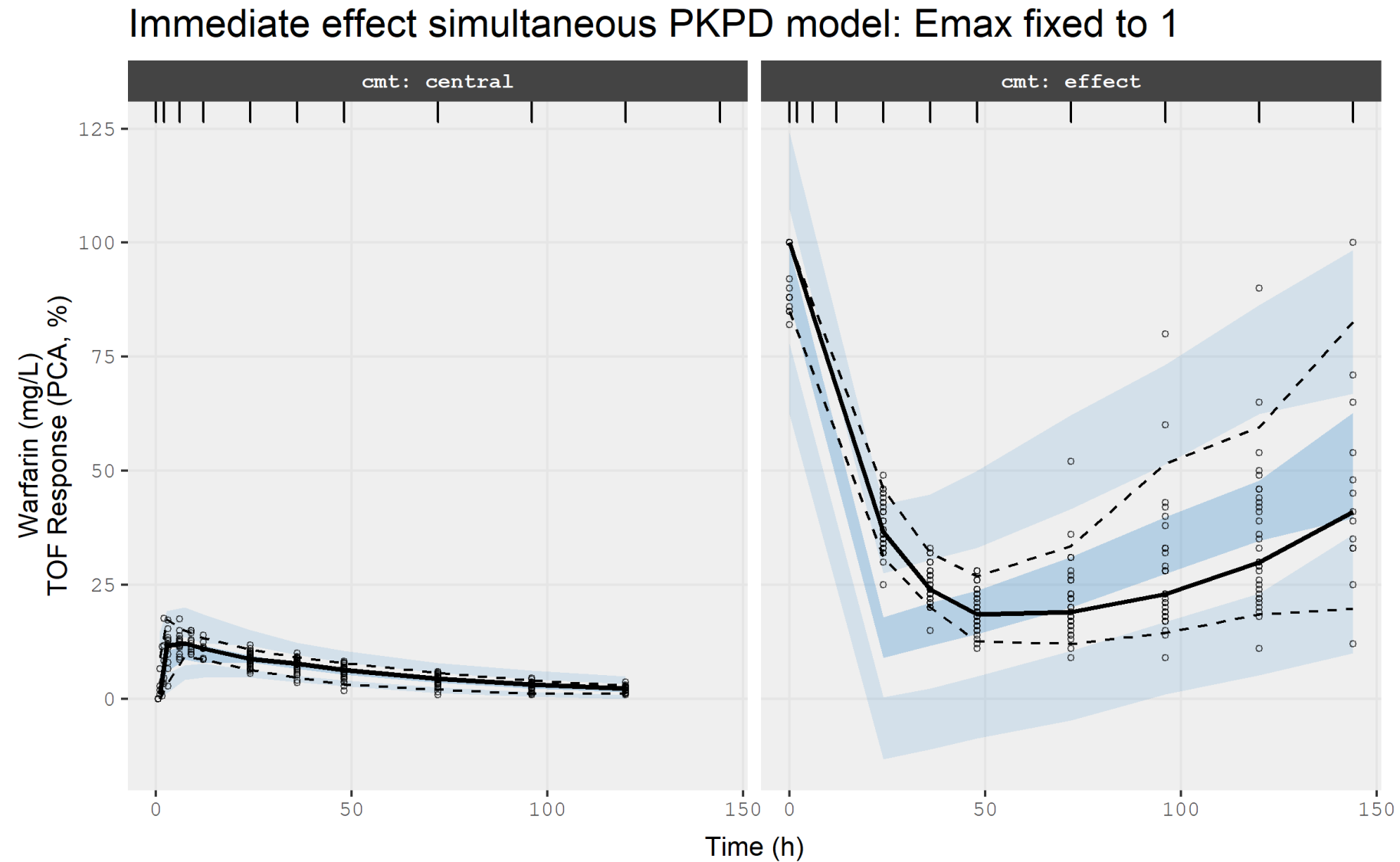
```
> fitKA1tr1_PDimmemax1_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 1753.422 2667.117 2721.457      -1320.558      430.4273

-- Time (sec; fitKA1tr1_PDimmemax1_F$time): -----
      setup optimize covariance table  other
elapsed 33.241    12.68    12.68  0.03 88.409

-- Population Parameters (fitKA1tr1_PDimmemax1_F$parFixed or fitKA1tr1_PDimmemax
Parameter Est. SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr      log ktr (/h) 0.0799 0.221 277      1.08 (0.702, 1.67) 67.2 48.1%
tc1       log CL (L/h) -2.03 0.0616 3.03      0.131 (0.116, 0.148) 28.2 0.818%
tv        log Vc (L) 2.11 0.0488 2.32      8.21 (7.47, 9.04) 23.4 6.34%
eps.pkprop      0.108      0.108
eps.pkadd       0.451      0.451
tc50      log ec50 (mg/L) 0.352 0.0911 25.9      1.42 (1.19, 1.7) 35.3 34.5%
te0        log e0 4.53 0.0151 0.332      92.6 (89.9, 95.3) 9.58 63.3%
eps.pdadd      12.2      12.2

Covariance Type (fitKA1tr1_PDimmemax1_F$covMethod): r,s
Fixed parameter correlations in fitKA1tr1_PDimmemax1_F$cor
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKA1tr1_PDimmemax1_F$omega) or correlation (fitKA1tr1_PDimmemax1_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKA1tr1_PDimmemax1_F$shrink
Minimization message (fitKA1tr1_PDimmemax1_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outerOpt="bobyqa"))
```

VPC for immediate effect simultaneous PKPD model



Effect compartment simultaneous PKPD analysis with **nlmixr**: **ini** block

```
#Effect compartment model
KA1tr1_PDceemax <- function() {
  ini({
    ## PK
    tktr <- log(1)    # log ktr (/h)
    tc1  <- log(0.1)  # log CL (L/h)
    tv   <- log(8)    # log Vc (L)
    eta.ktr ~ 1
    eta.cl ~ 0.1
    eta.v ~ 0.1
    eps.pkprop <- 0.1 #proportional error (SD/mean)
    eps.pkadd <- 0.4  #additive error (mg/L)

    ## PD
    tc50 <- log(1)    #log ec50 (mg/L)
    tkout <- log(0.05) #log tkout (/h)
    te0  <- log(100)  #log e0
    eta.c50 ~ .5
    eta.kout ~ .1
    eta.e0 ~ .1
    eps.pdadd <- 100
  })
  model({
  })
}
```

Effect compartment simultaneous PKPD analysis with **nlmixr**: **model** block

```
#Effect compartment model
model({
  ktr <- exp(tktr + eta.ktr)
  cl  <- exp(tc1 + eta.cl)
  v   <- exp(tv + eta.v)

  c50 = exp(tc50 + eta.c50)
  kout = exp(tkout + eta.kout)
  e0   = exp(te0 + eta.e0)
  emax = 1

  cp          = central/v
  d/dt(depot) = -ktr * depot
  d/dt(central) = ktr * trans - cl * cp
  d/dt(trans)  = ktr * depot - ktr * trans
  d/dt(ce)     = kout * (cp - ce)

  effect      = e0 * (1 - emax * ce/(c50 + ce))

  cp ~ prop(eps.pkprop) + add(eps.pkadd) | central
  effect ~ add(eps.pdadd) | effect
})
}
```

nlmixr output: effect compartment simultaneous PKPD model

```
> fitKA1tr1_PDceemax_F
-- nlmixr FOCEi (outer: nlminb) fit -----
      OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 1521.036 2438.73 2501.43      -1204.365          603.1391

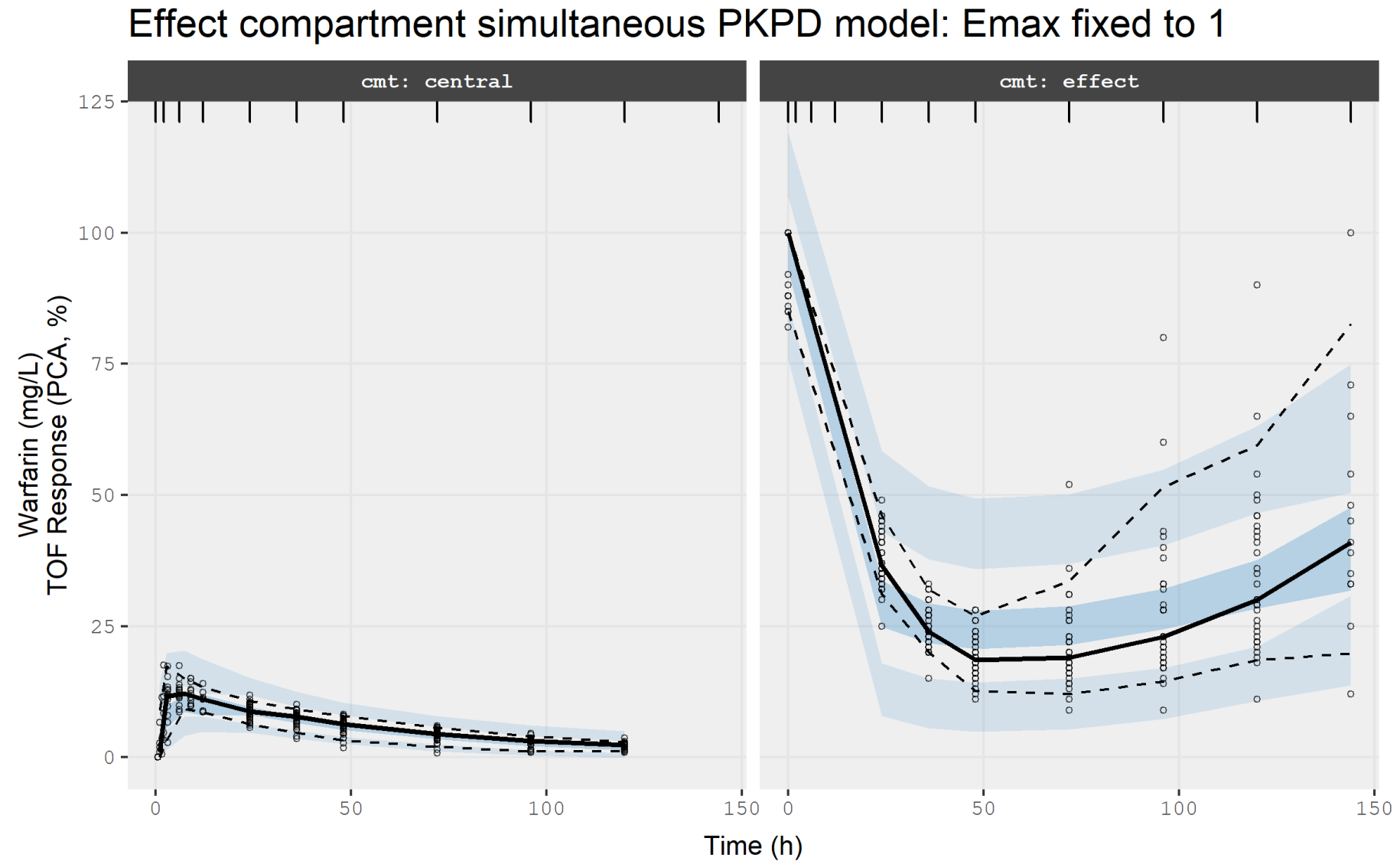
-- Time (sec; fitKA1tr1_PDceemax_F$time): -----
      setup optimize covariance table      other
elapsed 41.047   24.594       24.594  0.03 132.905

-- Population Parameters (fitKA1tr1_PDceemax_F$parFixed or fitKA1tr1_PDceemax_F$
Parameter      Est.      SE      %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr      log ktr (/h) 0.000614 0.315 5.13e+004      1 (0.539, 1.86) 68.1 47.2%
tc1      log CL (L/h) -2.01 0.0216 1.07 0.134 (0.128, 0.139) 29.2 1.44%
tv      log vc (L) 2.08 0.0688 3.31 7.99 (6.98, 9.14) 23.7 9.08%
eps.pkprop      0.106      0.106
eps.pkadd      0.444      0.444
tc50      log ec50 (mg/L) 0.507 0.0653 12.9 1.66 (1.46, 1.89) 34.4 20.5%
tkout      log tkout (/h) -3.85 0.0704 1.83 0.0214 (0.0186, 0.0245) 35.3 34.1%
te0      log e0 4.57 0.0162 0.356 96.5 (93.5, 99.6) 8.45 48.3%
eps.pdadd      6.42      6.42

Covariance Type (fitKA1tr1_PDceemax_F$covMethod): r,s
Some strong fixed parameter correlations exist (fitKA1tr1_PDceemax_F$cor) :
cor:tc1,tktr      cor:tv,tktr      cor:tc50,tktr      cor:tkout,tktr      cor:te0,tktr      cor:tv,tc1      cor:tc50,tc1      cor:tkout,tc1      cor:te0,tc1      cor:tc50,tv
-0.244      -0.262      -0.153      -0.297      0.165      0.370      -0.292      0.154      0.0369
0.342
cor:te0,tv      cor:tkout,tc50      cor:te0,tc50      cor:te0,tkout
0.134      0.792      -0.0510      0.0673

No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKA1tr1_PDceemax_F$omega) or correlation (fitKA1tr1_PDceemax_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKA1tr1_PDceemax_F$shrink
Minimization message (fitKA1tr1_PDceemax_F$message):
false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
nlmixr(...,control=list(outeropt="bobyqa"))
```


VPC for effect compartment simultaneous PKPD model



Turnover model simultaneous PKPD analysis with **nlmixr**: **ini** block

```
#Turnover simultaneous PKPD model
KA1tr1IPP_PDtoemax <- function() {
  ini({
    ## PK
    tktr <- log(1) # log ktr (/h)
    tc1 <- log(0.1) # log CL (L/h)
    tv <- log(8) # log Vc (L)
    eta.ktr ~ 1
    eta.cl ~ 0.1
    eta.v ~ 0.1
    eps.pkprop <- 0.1 # proportional error (SD/mean)
    eps.pkadd <- 0.4 # additive error (mg/L)

    ## PD
    tc50 <- log(1) #log ec50 (mg/L)
    tkout <- log(0.05) #log tkout (/h)
    te0 <- log(100) #log e0
    eta.c50 ~ .5
    eta.kout ~ .1
    eta.e0 ~ .1
    eps.pdadd <- 100
  })
  model({
  })
}
```

Turnover model simultaneous PKPD analysis with `nlmixr: model` block

```
#Turnover simultaneous PKPD model
model({
  ktr <- exp(tktr + eta.ktr)
  cl  <- exp(tc1 + eta.cl)
  v   <- exp(tv + eta.v)
  c50 = exp(tc50 + eta.c50)
  kout = exp(tkout + eta.kout)
  e0   = exp(te0 + eta.e0)
  emax = 1

  cp          = central/v
  d/dt(depot) = -ktr * depot
  d/dt(central) = ktr * trans - cl * cp
  d/dt(trans)  = ktr * depot - ktr * trans
  effect(0)    = e0
  kin          = e0 * kout
  PD           = 1 - emax * cp/(c50 + cp)
  d/dt(effect) = kin * PD - kout * effect

  cp ~ prop(eps.pkprop) + add(eps.pkadd) | central
  effect ~ add(eps.pdadd) | effect
})
}
```

nlmixr output: turnover simultaneous PKPD model

```
> fitKA1tr1IPP_PDtoemax_F
-- nlmixr FOCEi (outer: nlminb) fit -----
              OBJF      AIC      BIC Log-likelihood Condition Number
FOCEi 1330.843 2248.537 2311.238      -1109.269      163.6949

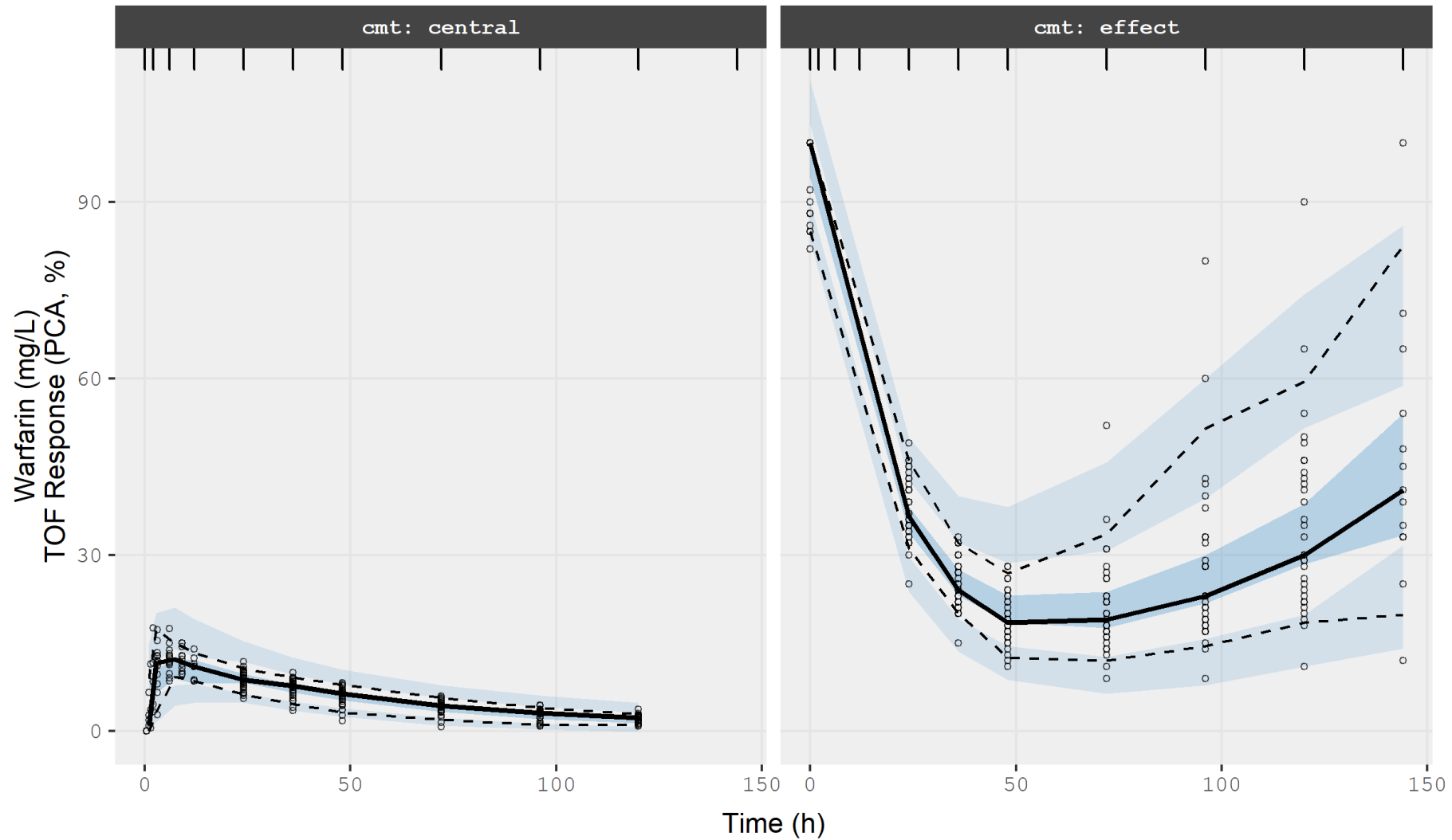
-- Time (sec; fitKA1tr1IPP_PDtoemax_F$time): -----
      setup optimize covariance table  other
elapsed 47.905      30      30 0.04 234.965

-- Population Parameters (fitKA1tr1IPP_PDtoemax_F$parFixed or fitKA1tr1IPP_PDtoe
Parameter      Est.      SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr      log ktr (/h) 0.098 0.195 199      1.1 (0.753, 1.62)      67.4      48.1%
tc1      log CL (L/h) -2.02 0.0706 3.5      0.133 (0.116, 0.153)      29.9      4.24%
tv      log vc (L) 2.05 0.101 4.9      7.79 (6.4, 9.49)      22.4      4.21%
eps.pkprop      0.0989      0.0989
eps.pkadd      0.419      0.419
tc50      log ec50 (mg/L) 0.157 0.0958 60.9      1.17 (0.97, 1.41)      44.0      3.48%
tkout      log tkout (/h) -2.95 0.0463 1.57 0.0523 (0.0477, 0.0572)      10.0      31.0%
te0      log e0 4.57 0.0246 0.538      96.5 (91.9, 101)      5.95      21.3%
eps.pdadd      3.73      3.73

Covariance Type (fitKA1tr1IPP_PDtoemax_F$covMethod): r,s
Fixed parameter correlations in fitKA1tr1IPP_PDtoemax_F$cor
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fitKA1tr1IPP_PDtoemax_F$omega) or correlation (fitKA1tr1IPP_PDtoemax_F$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fitKA1tr1IPP_PDtoemax_F$shrink
Minimization message (fitKA1tr1IPP_PDtoemax_F$message):
  false convergence (8)
In an ODE system, false convergence may mean "useless" evaluations were performed.
See https://tinyurl.com/yyrrwkce
It could also mean the convergence is poor, check results before accepting fit
You may also try a good derivative free optimization:
  nlmixr(...,control=list(outeropt="bobyqa"))
```

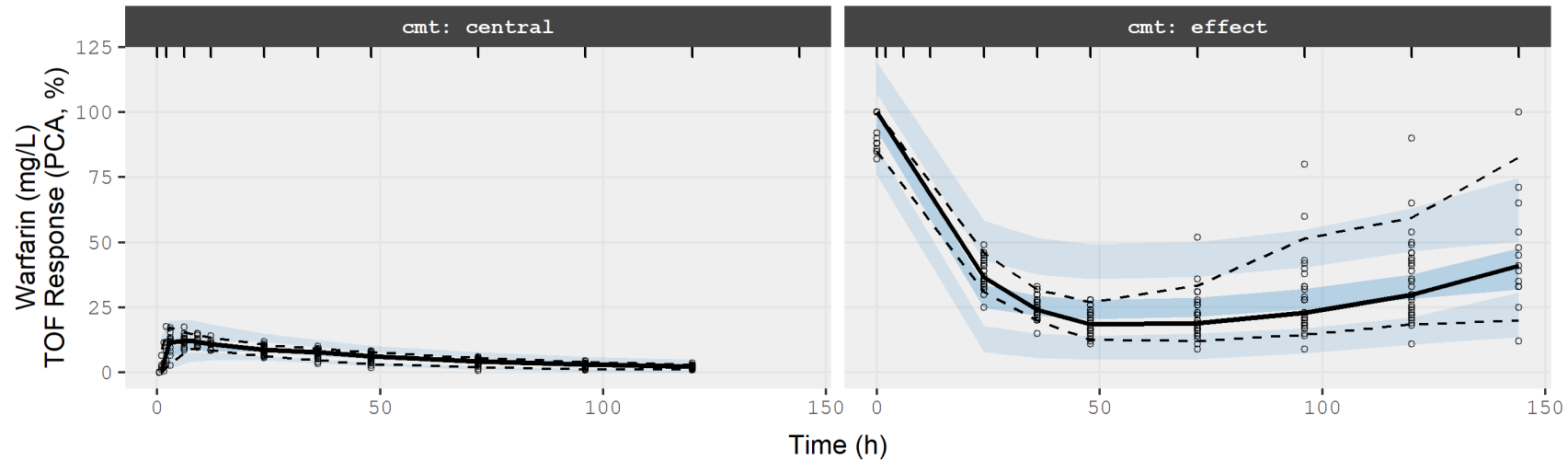
VPC for turnover simultaneous PKPD model

Turnover simultaneous PKPD model: Emax fixed to 1

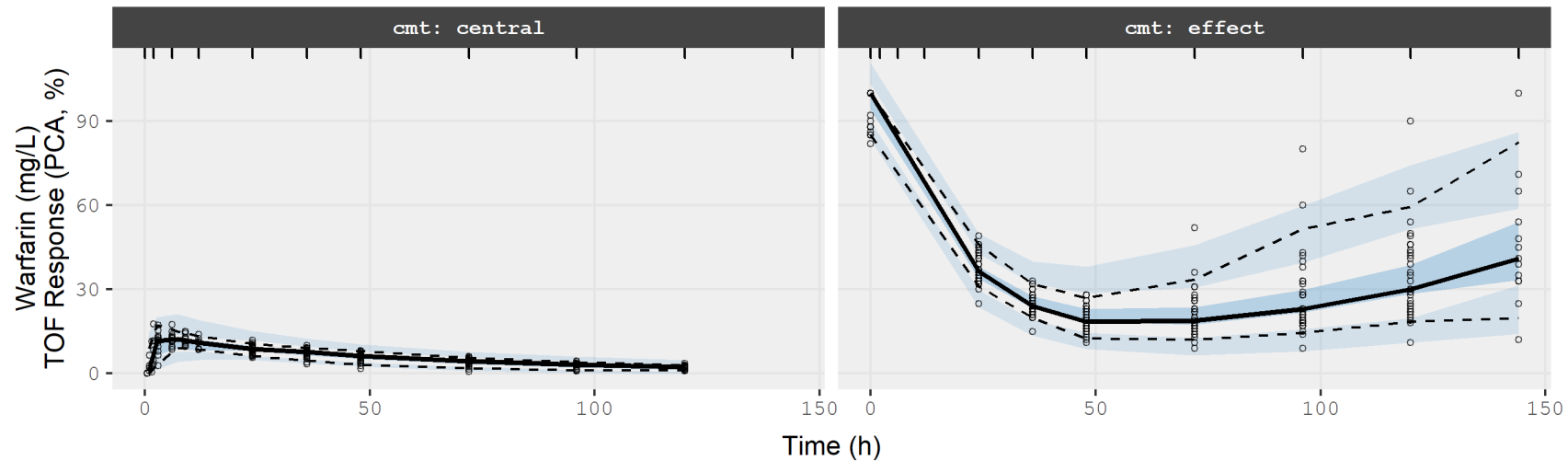


VPCs to compare the effect compartment simultaneous model (top row, OFV=1521) with the turnover simultaneous PKPD model (bottom row, OFV=1331)

Effect compartment simultaneous PKPD model: Emax fixed to 1



Turnover simultaneous PKPD model: Emax fixed to 1



Hands-on session VI: running a simultaneous **nlmixr** PKPD analysis

- Examine the code in PAWS_5.R to run one of the simultaneous PKPD analyses

Individual graphs with multiple endpoints currently require advanced RxODE tricks: eventTable for multiple subjects

- RxODE examples so far have only been for a single subject
- The eventTable can define multiple subjects as well: you can even use a NONMEM data set for this (it has time points and doses)

- Start with a table of doses

#Generate a data.table of doses

```
Doses <- dataF[AMT > 0, .(ID, TIME, AMT, EVID = 1)]
```

- Create an eventTable using `et(Doses)` and cook these together with simulation time points using `et(0, 150, by = 0.5)` with the help of magrittr piping (`%>%`)

#Generate an eventTable by combining the doses with the same set of fine-meshed sampling points for all subjects using the et function and magrittr (%>%) piping

```
evt <- et(Doses) %>% et(0, 150, by = 0.5)
```

- Create a table of EBEs

#Generate a data.table of EBEs: the nlmixr fit object contains EBEs

```
EBEs_KA1tr1_PDimmemax1_F <- data.table(fitKA1tr1_PDimmemax1_F)
```

#Take only the first record for each ID:

```
EBEs_KA1tr1_PDimmemax1_F <-  
  EBEs_KA1tr1_PDimmemax1_F[!duplicated(ID), .(ID, ktr, cl, v, c50, e0)]
```


Individual graphs with multiple endpoints currently require advanced RxODE tricks: simulate...

```
#Define the RxODE model
modPKPD0 <- RxODE({
  cp          = central/v
  d/dt(depot) = -ktr * depot
  d/dt(trans) = ktr * depot - ktr * trans
  d/dt(central)= ktr * trans - cl * cp
  effect      = e0 * (1 - cp/(c50 + cp))
})

#Solve the system:
res1 <- data.table(rxSolve(modPKPD0, EBEs_KA1tr1_PDimmemax1_F, evt))

#Merge with observed data points
Rdata <- PKPDdata[MDV == 0, .(id = factor(ID), time = TIME, DVID, DV)]
xx1 <- merge(res1, Rdata, by = c("id", "time"), all.x = TRUE)
```

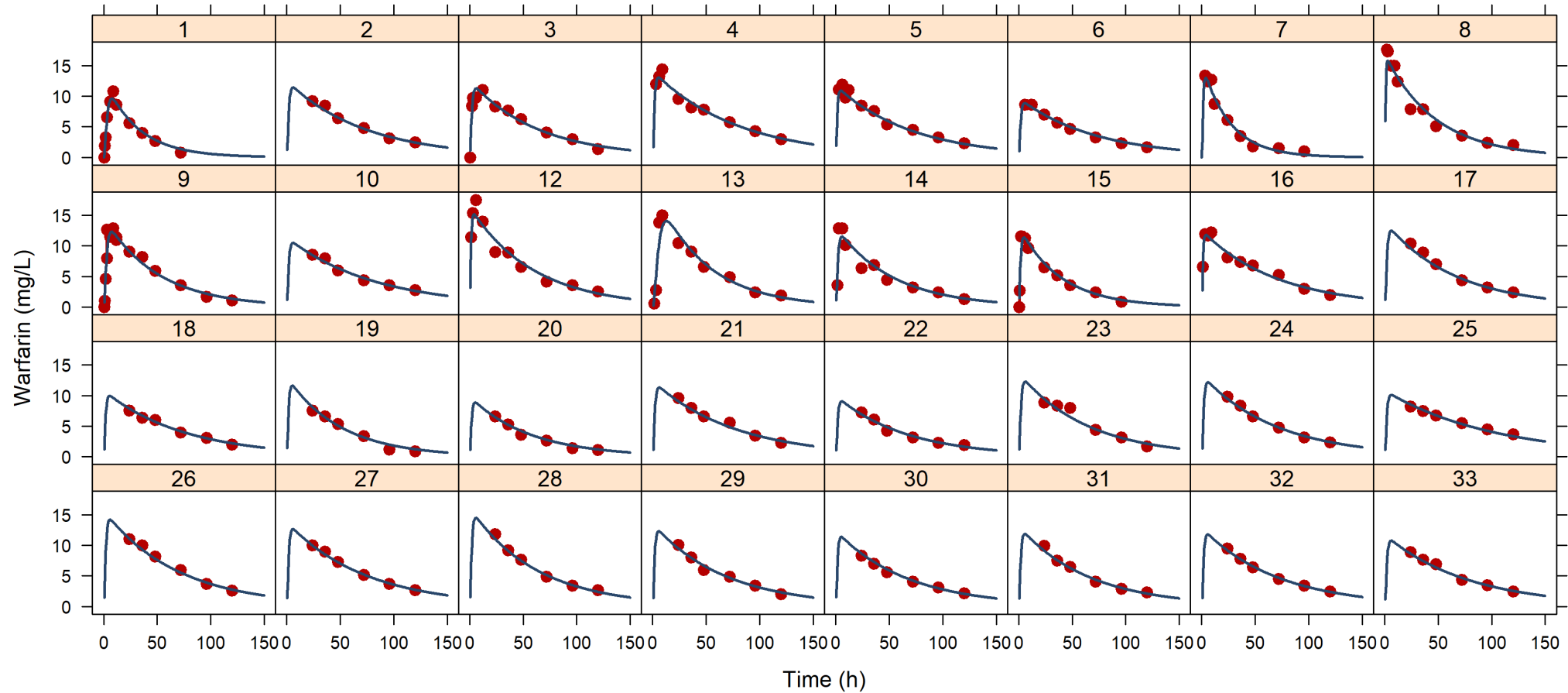
...and plot

```
#And plot:
xyplot(
  DV + effect ~ time | id,
  data = xx1[DVID == 2 | is.na(DVID)],
  type = c("b", "l"),
  col = nlmixCOLS[c(3, 2)],
  main = "PCA profiles for the immediate effect simultaneous PKPD model",
  cex = c(1, 0.1),
  layout=c(8,4),
  lwd = 2,
  pch = c(19, 1),
  xlab = "Time (h)\n",
  ylab = "TOF Response (PCA, %)",
  as.table = TRUE,
  scales = list(alternating = 1)
)
```

- But if I know Matt, he'll soon update **augPred** to cover multiple endpoints 😊

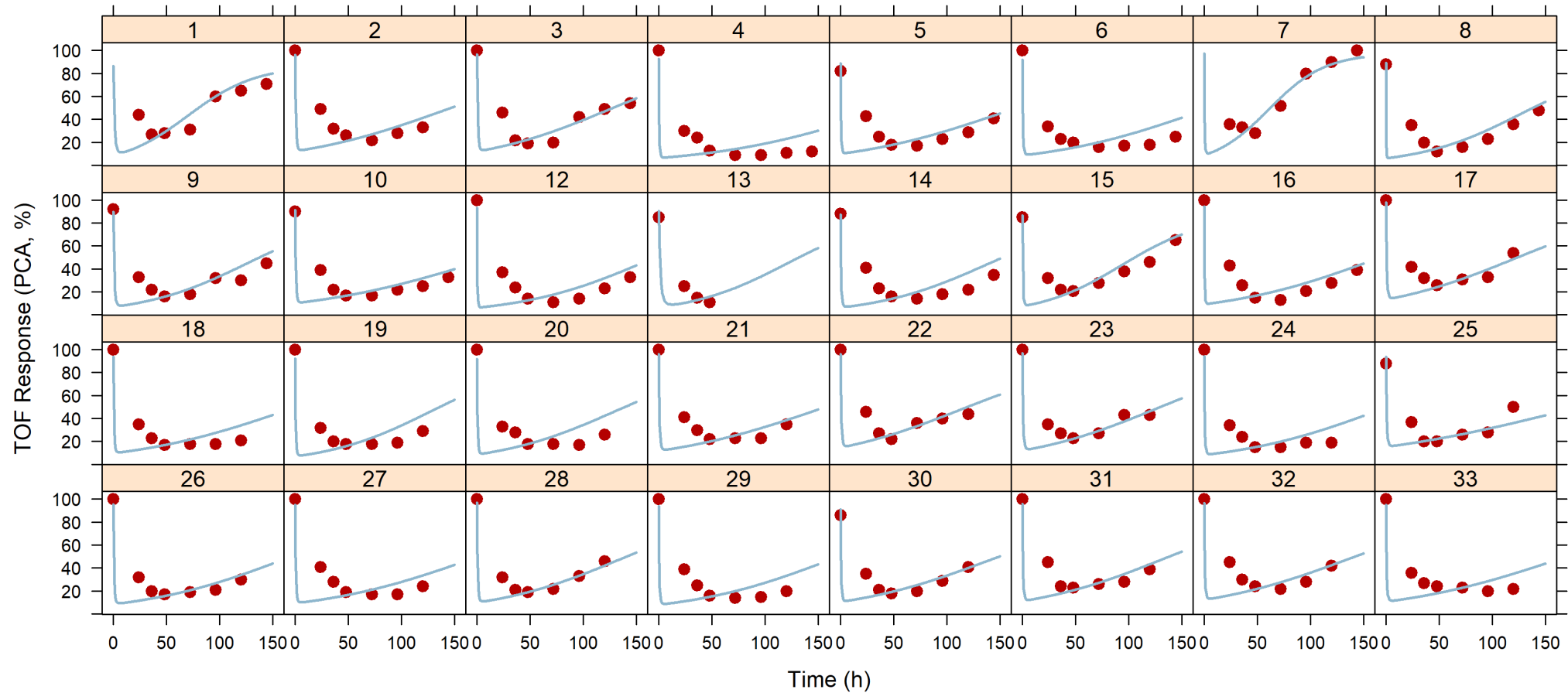
Individual graphs for warfarin PK of the immediate effect simultaneous PKPD model

Warfarin profiles for the immediate effect simultaneous PKPD model with E_{max} fixed to 1



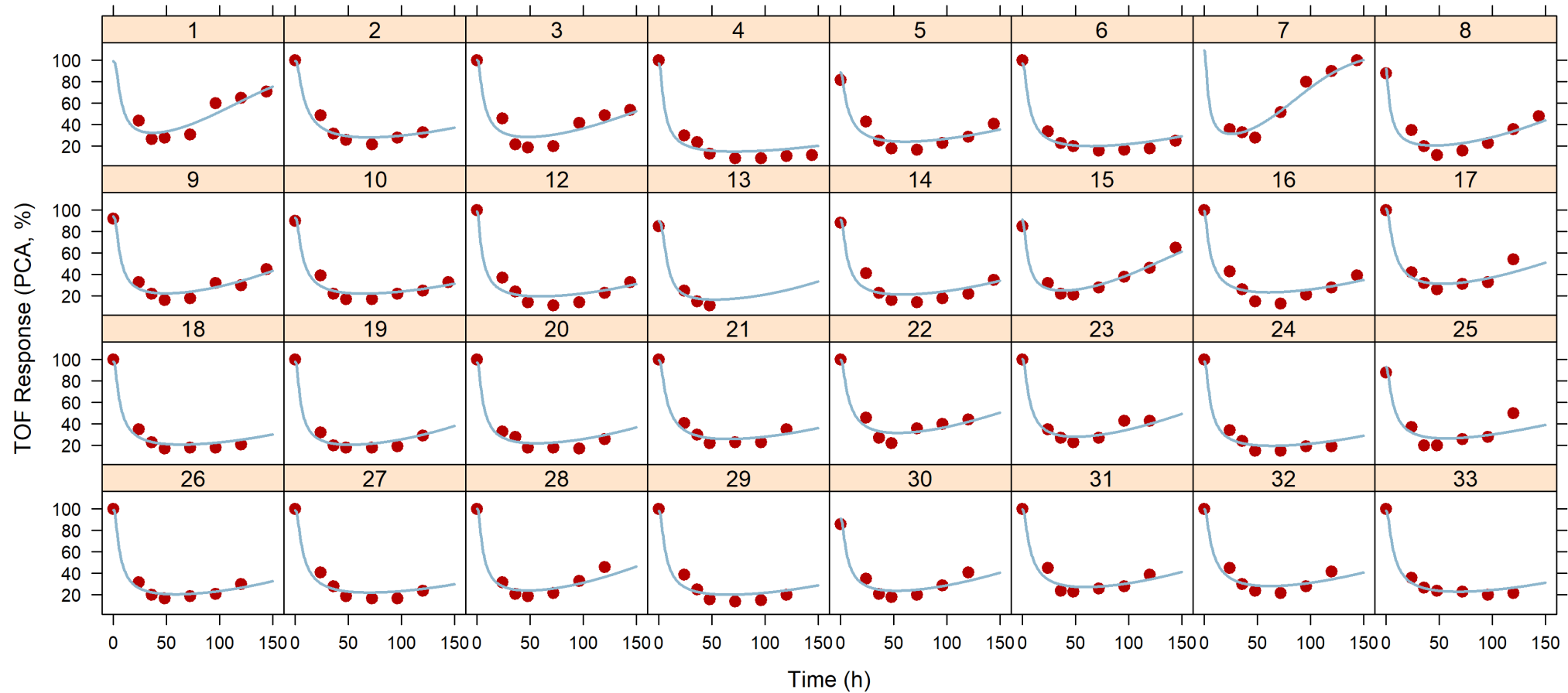
Individual graphs for PCA profiles for the immediate effect simultaneous PKPD model

PCA profiles for the immediate effect simultaneous PKPD model with E_{max} fixed to 1



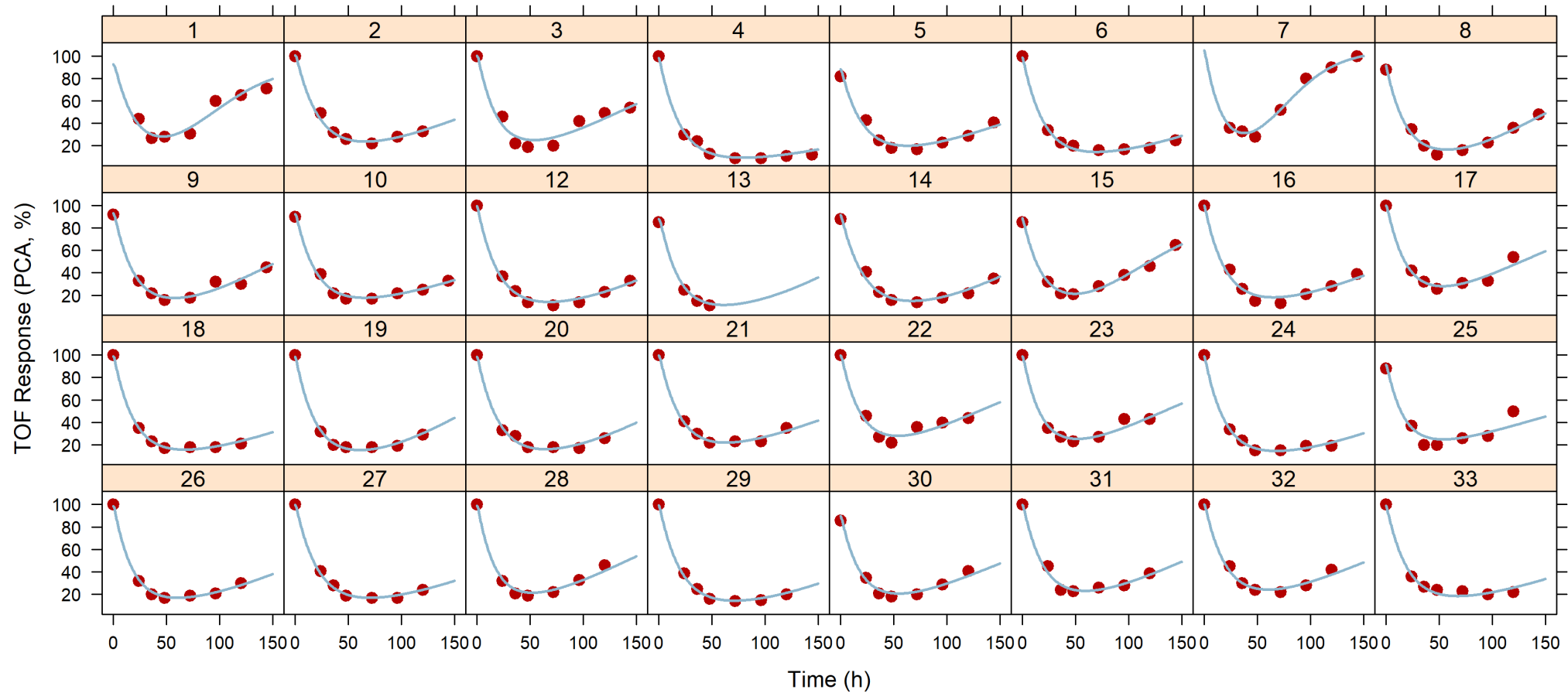
Individual graphs for PCA profiles for the simultaneous PKPD effect-compartment model

PCA profiles for simultaneous PKPD effect-compartment model with E_{max} fixed to 1



Individual graphs for PCA profiles for the simultaneous PKPD turnover model

PCA profiles for simultaneous PKPD turnover model with Emax fixed to 1



Hands-on session VII: simulate and plot individual profiles using RxODE

- Examine the code in PAWS_5.R to extract EBEs from the simultaneous PKPD model you ran
- Generate a fine-meshed population eventTable
- Update the RxODE model to match the model you actually ran
- Simulate and plot the results

Upcoming developments

- Implementation of transit models
- Use of a new CRAN package SymEngine to implement symbolic mathematics, currently still requiring Python/SymPy
 - Easier installation without Python dependency
 - More efficient calculations, likely increase in speed
 - Implementation of solved equations for FOCEI
- Inter-occasion variability
- Mixture models
- Categorical data models (ordered categorical, count data...)
- Parallelisation...
- ... 😊