Simulation and parameter estimation with rxode2 and nlmixr^2

rxode2/nlmixr2 Course PMx Africa

Kampala, Uganda, 2025

Rik Schoemaker, PhD

On behalf of the nlmixr development team:

Matt Fidler, Bill Denney, Richard Hooijmaijers, Rik Schoemaker, Mirjam Trame, Max Taubert, Theo Papathanasiou, Justin Wilkins, John Harrold, Anne Keunecke

Course outline

- Introduction to rxode2 simulation
- Introduction to nlmixr^2 and parameter estimation
- Hands on with the warfarin PK models (part 1)
- Advanced capabilities nlmixr^2
- Hands on with the warfarin PK and PKPD models (part 2)

rxode2 simulation software in R

- rxode2 is pharmacometric simulation software using ordinary differential equations (ODEs) as an open-source R package
- Rewritten by Matt Fidler after Wenping Wang's original RxODE package, available on CRAN[a] and GitHub[b], and described in a tutorial in CPT:PSP[c] and with online documentation[d]
- Simulation of ODEs was already possible in R (using deSolve), but was slow and virtually impossible to code with flexible dosing history
- rxode2 has rapid execution due to compilation in C
- rxode2 allows fully flexible dosing history
- Stable and mature software for Windows, OS X (in docker), Linux
- Requires external compilers (provided by Rtools on Windows)
- rxode2 is an evolution of the original RxODE package: after a full re-write of the code backward compatibility could not be guaranteed

[a] CRAN: https://cran.r-project.org/web/packages/rxode2
[b] GitHub: https://github.com/nlmixr2/rxode2
[c] Wang W et al. CPT:PSP (2016) 5, 3–10.
[d] RxODE packagedown: https://nlmixr2.github.io/rxode2

Basic example load the library and define the ODEs

```
library(rxode2)

## 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(rxode2)

## 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 <- rxode2(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
```

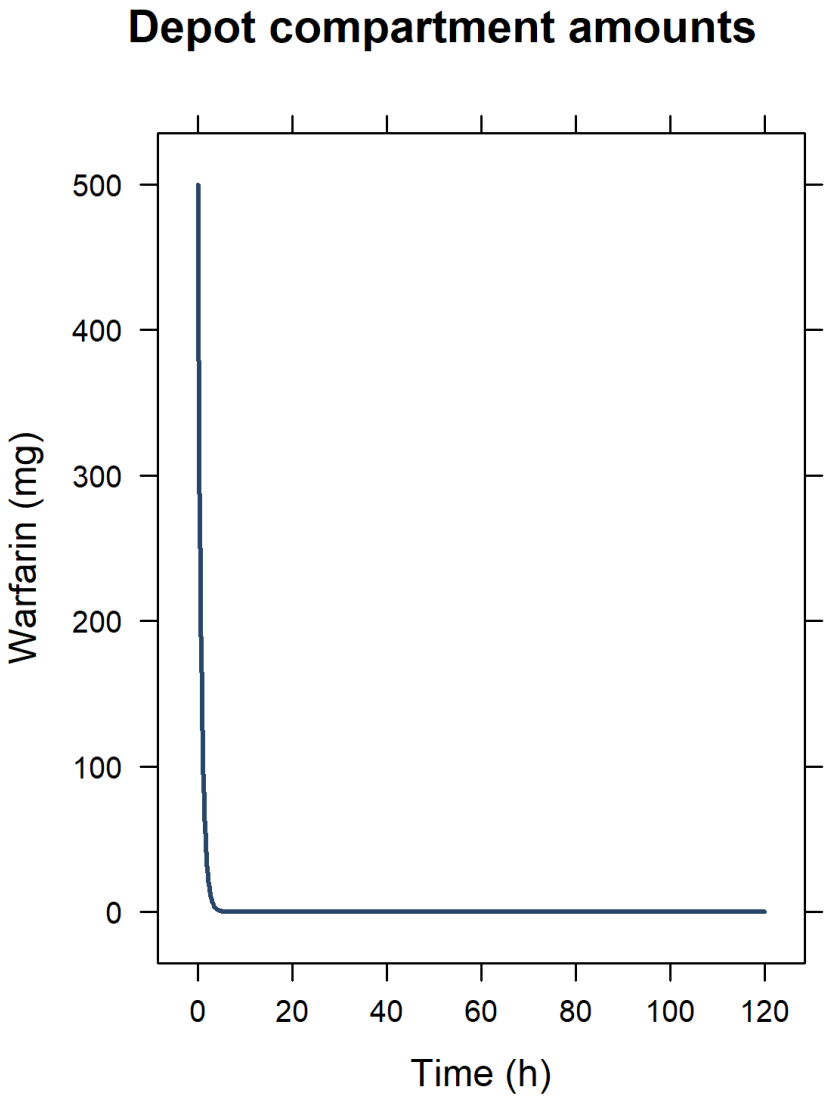
Create an eventTable that defines the doses and the sampling times

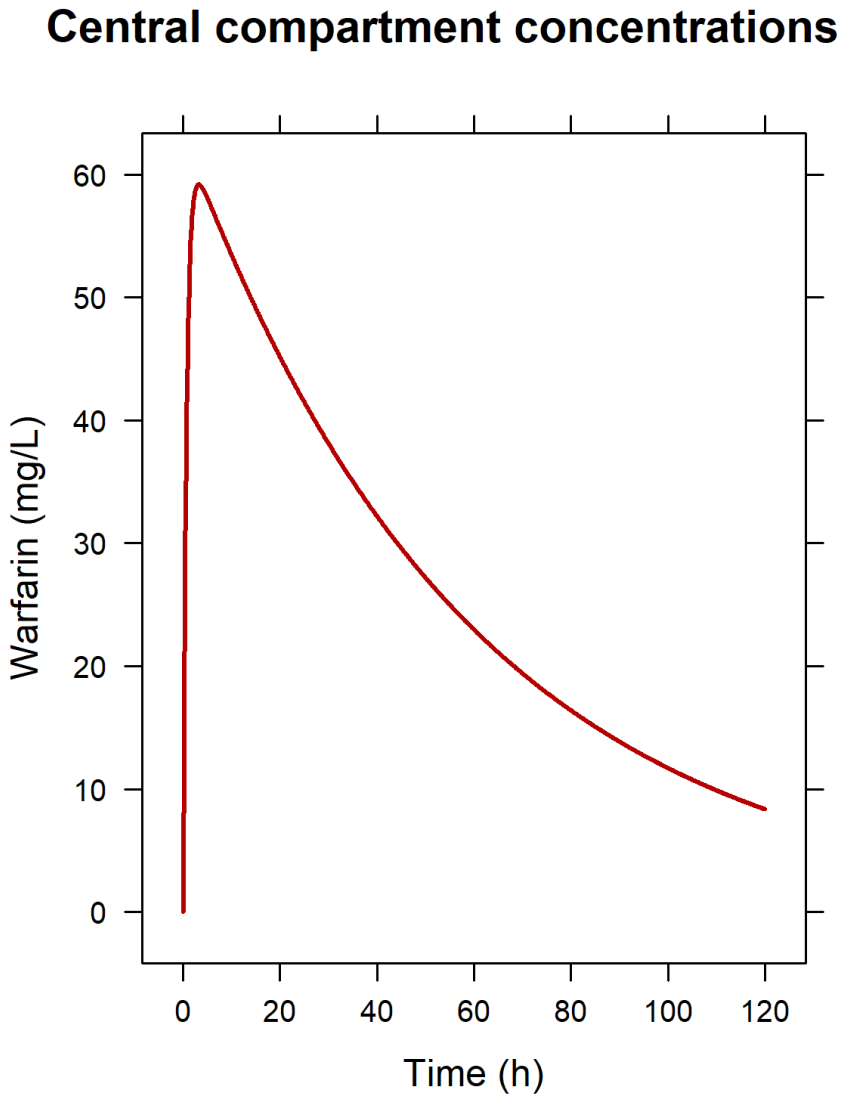
```
library(rxode2)
## set up the system of differential equations (ODEs)
odeKA1 <- "
 d/dt(depot)   = -ka*depot;                # This is compartment number 1 (depot)
 d/dt(central) =  ka*depot-(cl/v)*central; # This is compartment number 2 (central)
 C1=central/v;                             # Calculates concentration from amount
"
## compile the model
modKA1 <- rxode2(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))
```

Solve the equations and plot the results

```
library(rxode2)
## 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 <- rxode2(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

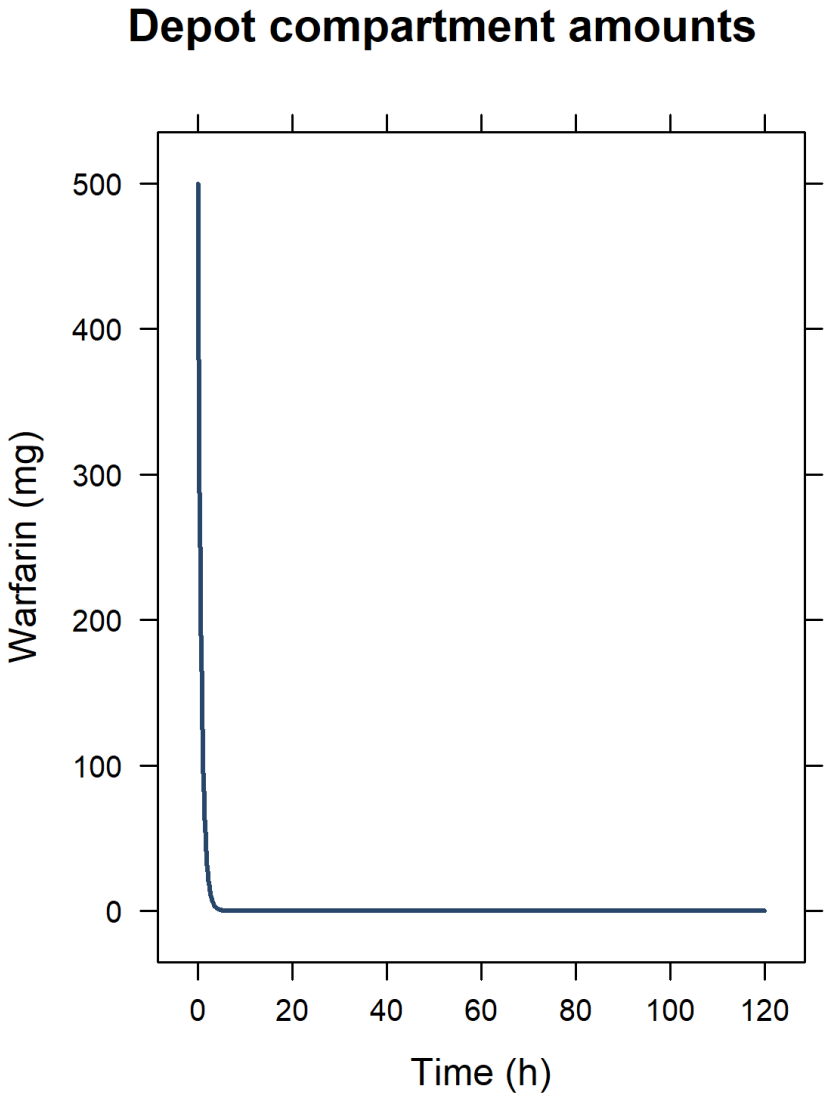


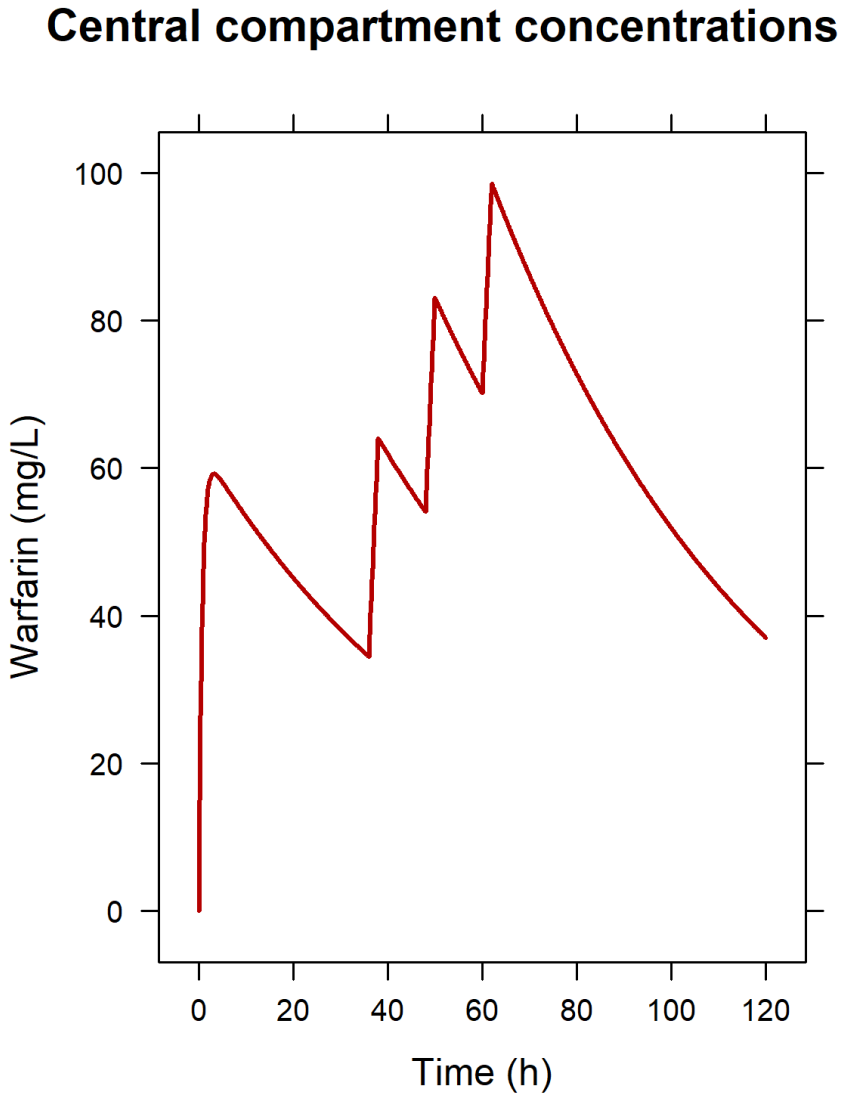


Adding extra doses (expand the existing eventTable): three additional infusions in the central compartment

```
## 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

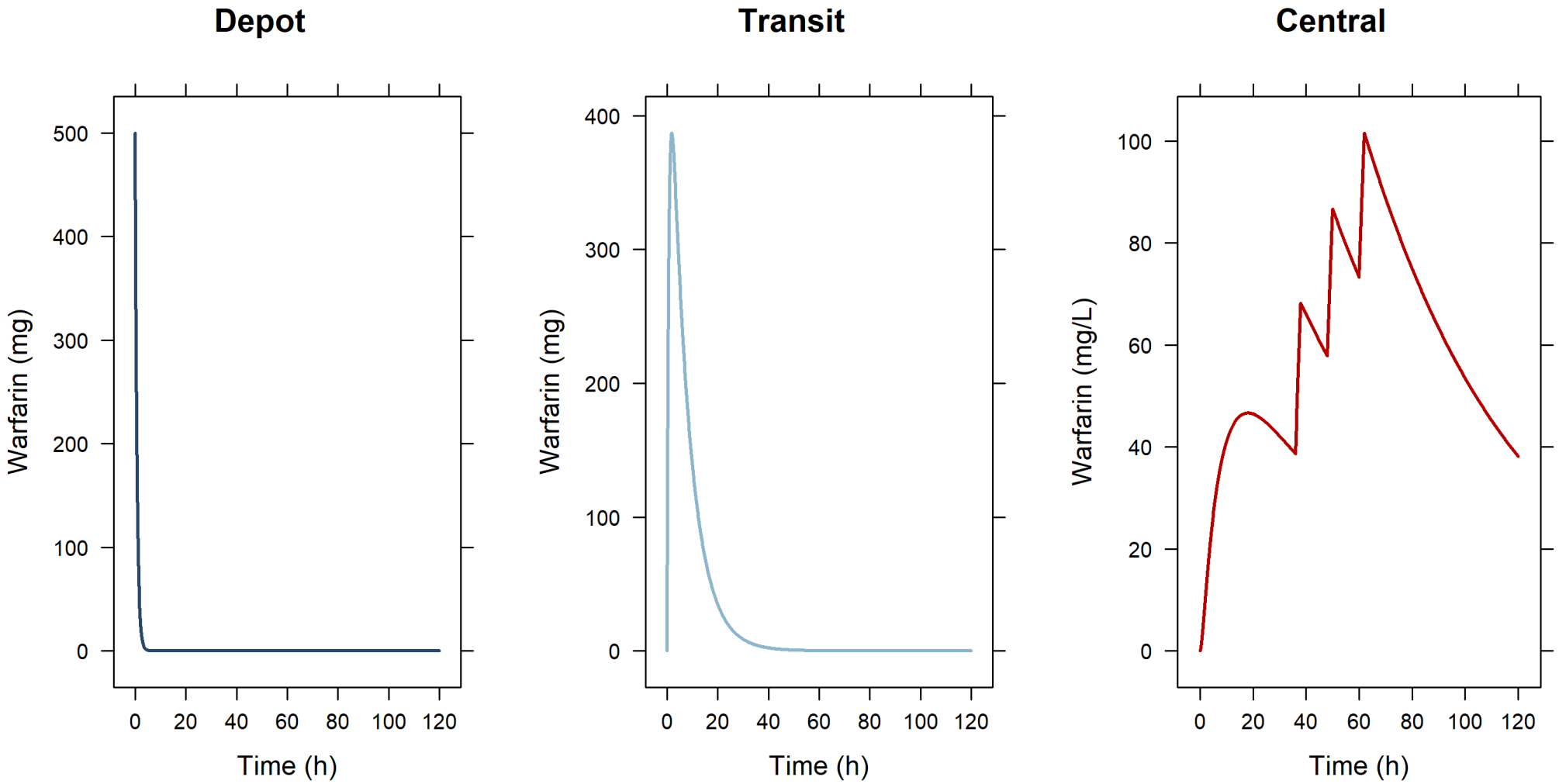




Extend the model: add a transit compartment…

```
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 <- rxode2(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
```

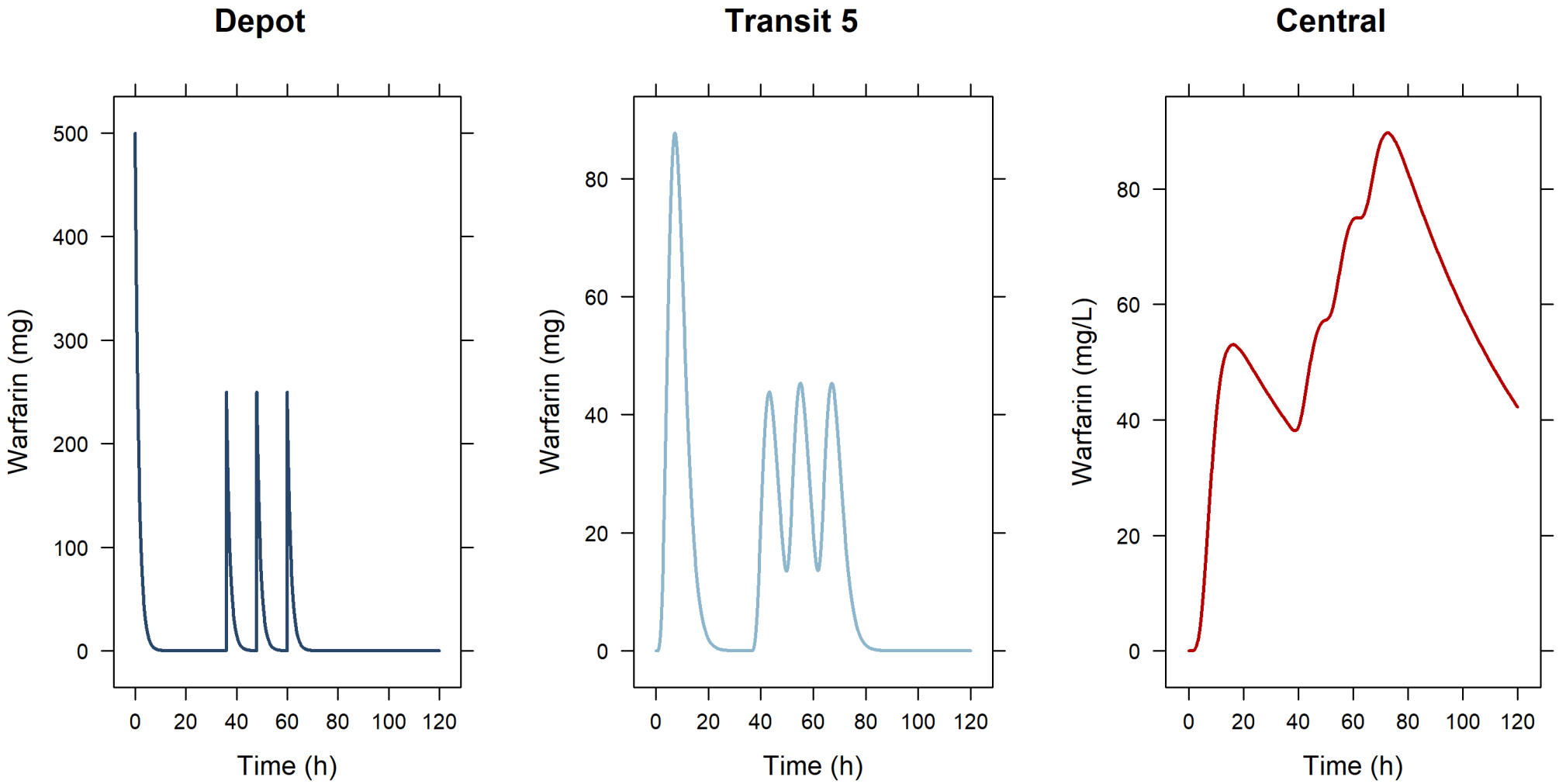
...adding a transit compartment between depot and central



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 <- rxode2(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



Alternatively: use NONMEM-style dataset as input

- Instead of eventTables you can also use a NONMEM dataset structure to simulate

Hands-on session I: rxode2 simulations

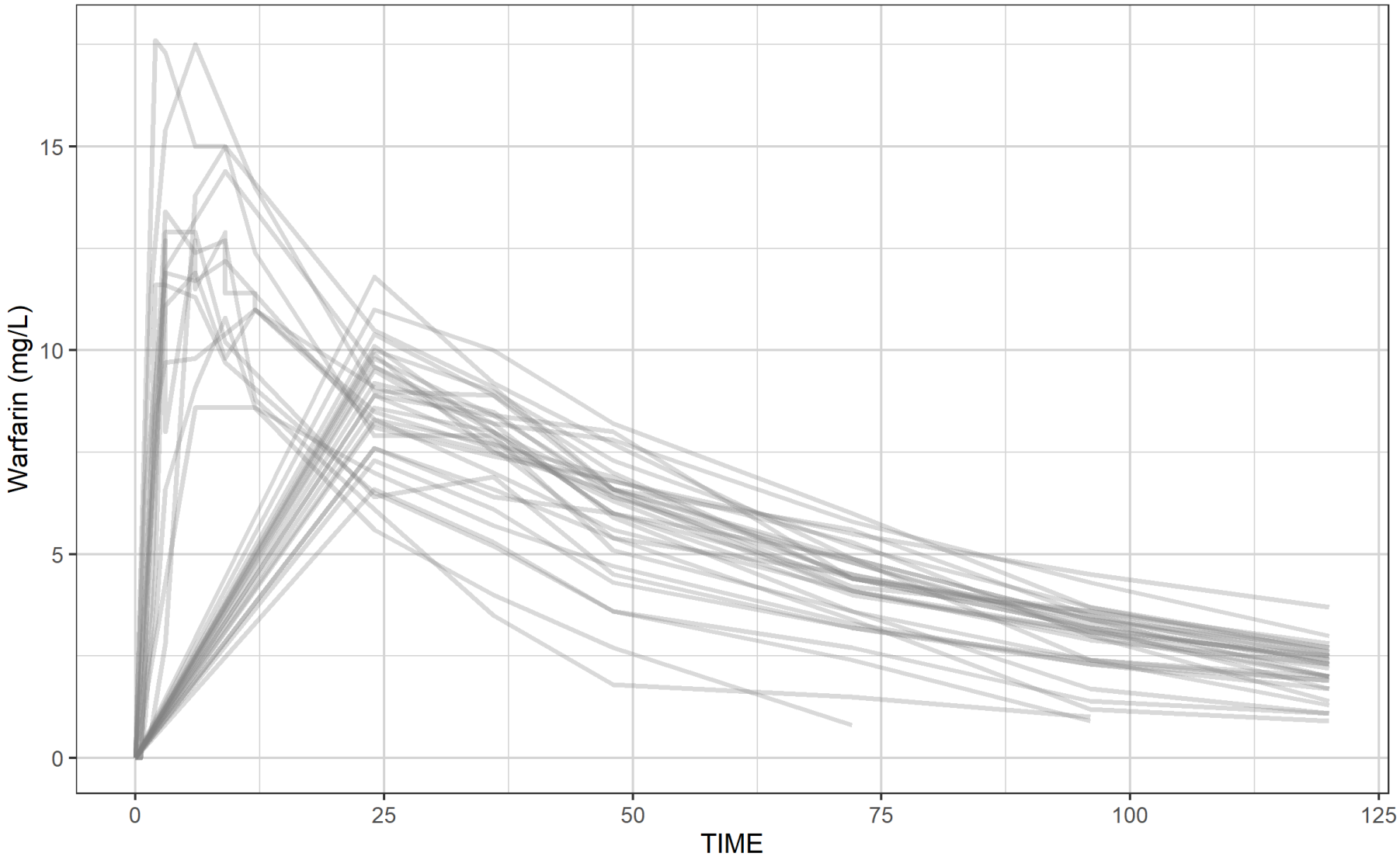
- Examine the code in HandsOn_1.R to run pre-programmed simulations and try out your own variations

Assess the data before you start analysis

- Classic data sets used by Nick Holford in his courses:
  - O'Reilly RA, Aggeler PM. Studies on coumarin anticoagulant drugs initiation of warfarin therapy without a loading dose. Circulation 1968;38:169-177
  - O'Reilly RA, Aggeler PM, Leong LS. Studies of the coumarin anticoagulant drugs: The pharmacodynamics of warfarin in man. Journal of Clinical Investigation 1963;42(10):1542-1551

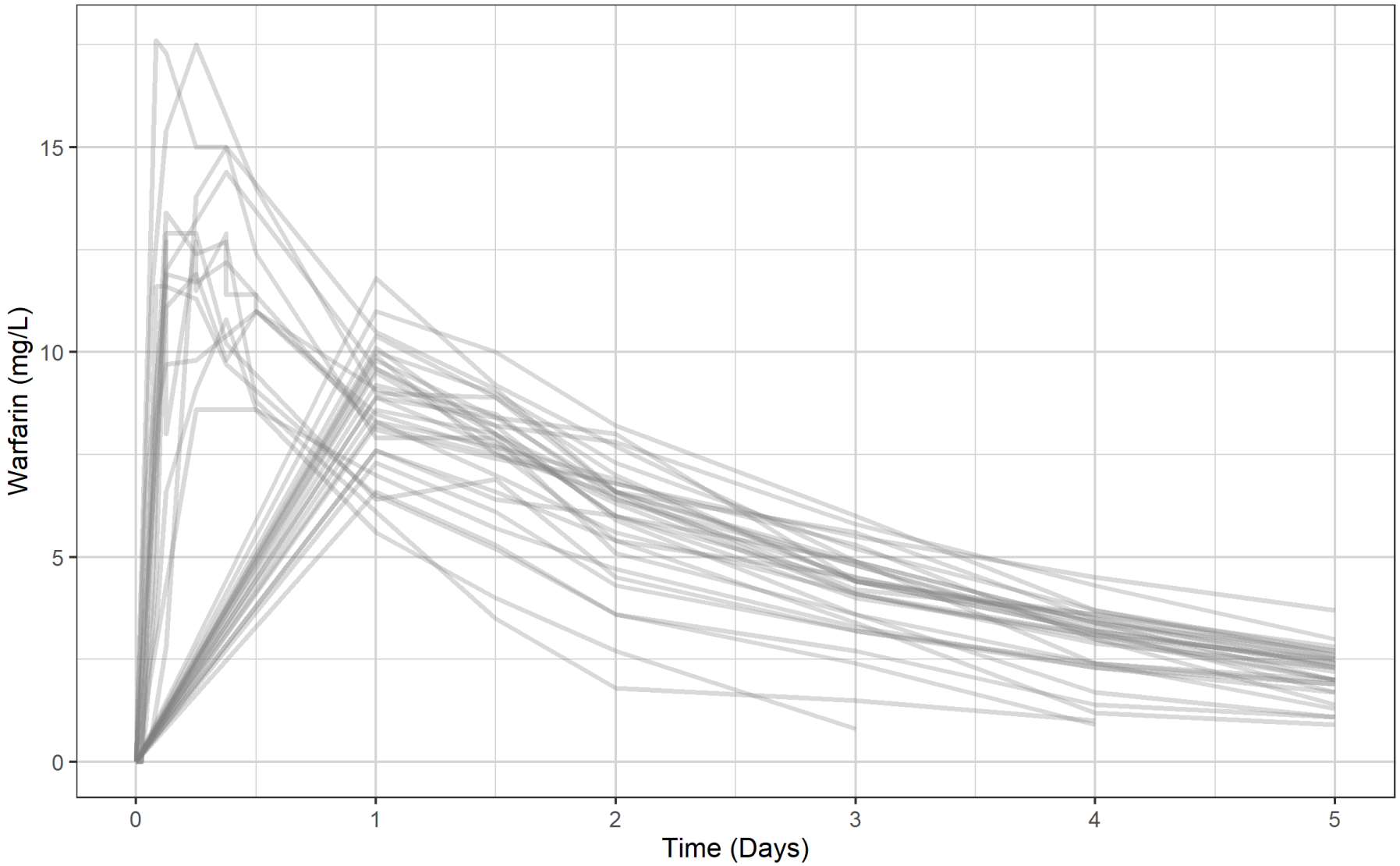
- Some simple plots to get a feel of the data
- Additional xgxr functionality to improve and summarize the profiles

Our warfarin data file: a simple ggplot to provide an impression



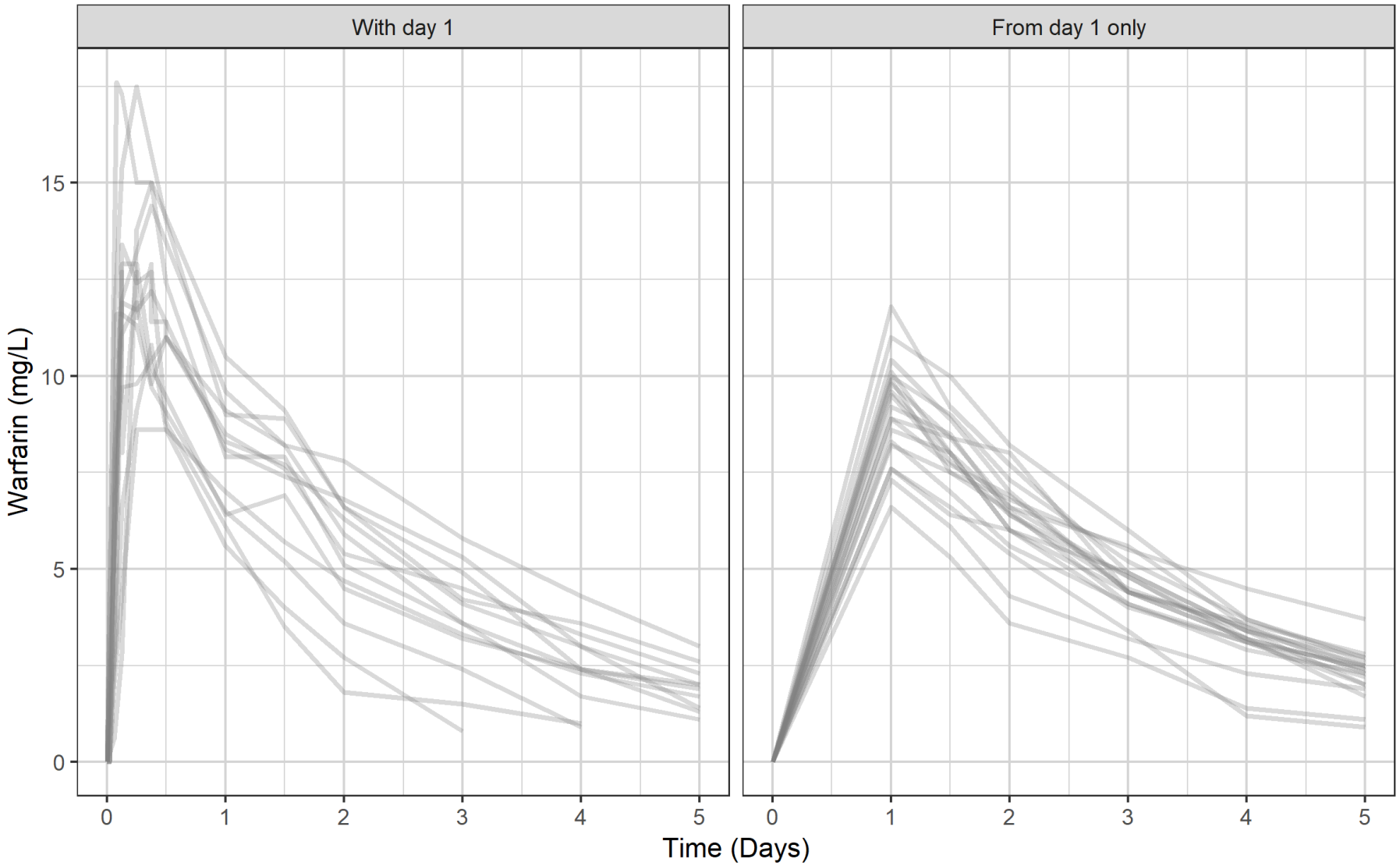
Change the x-axis from hours to days and add a proper label using the xgx helper

```
xgx_scale_x_time_units(units_dataset = "hours", units_plot = "days")
```

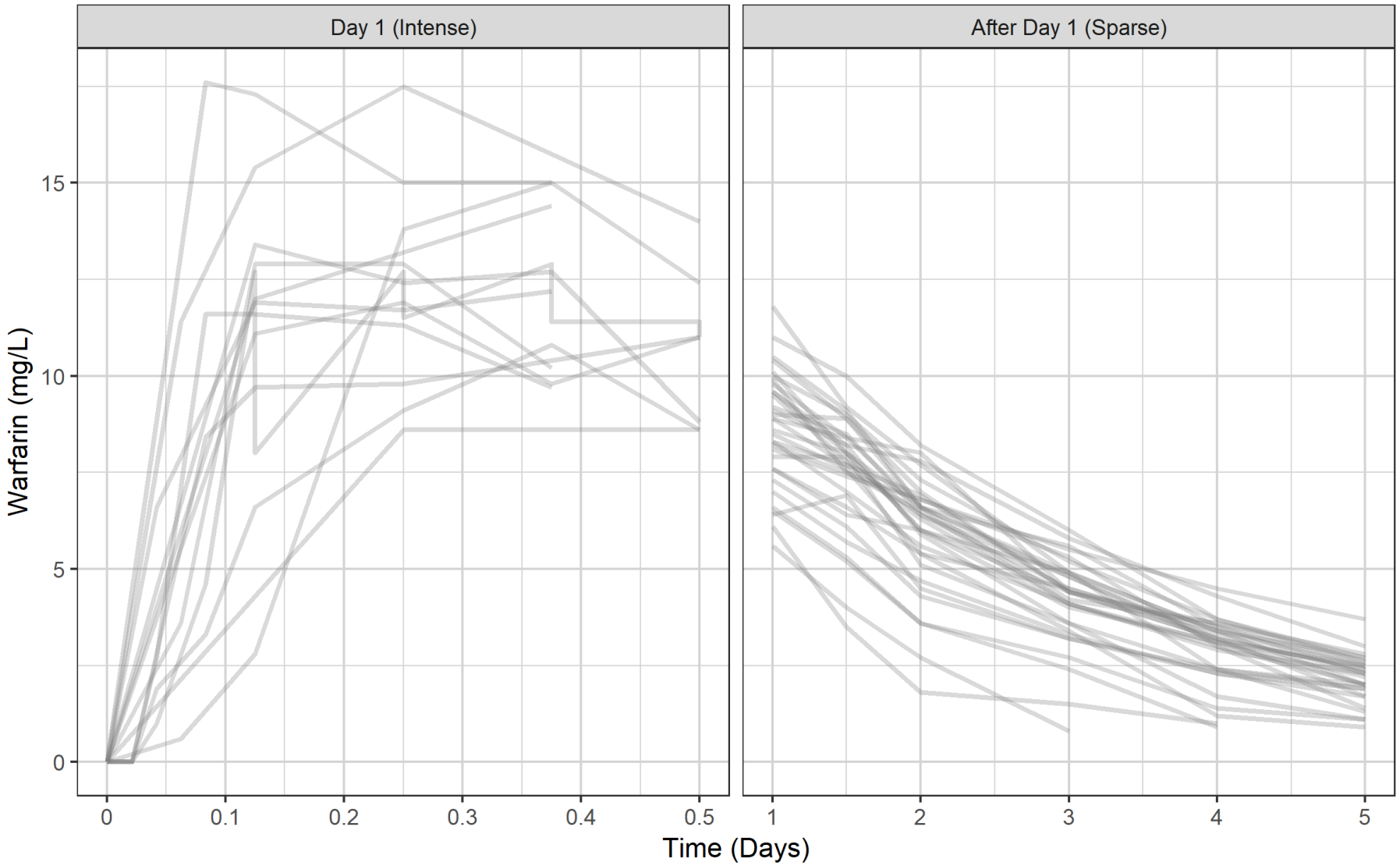


nlmixr2

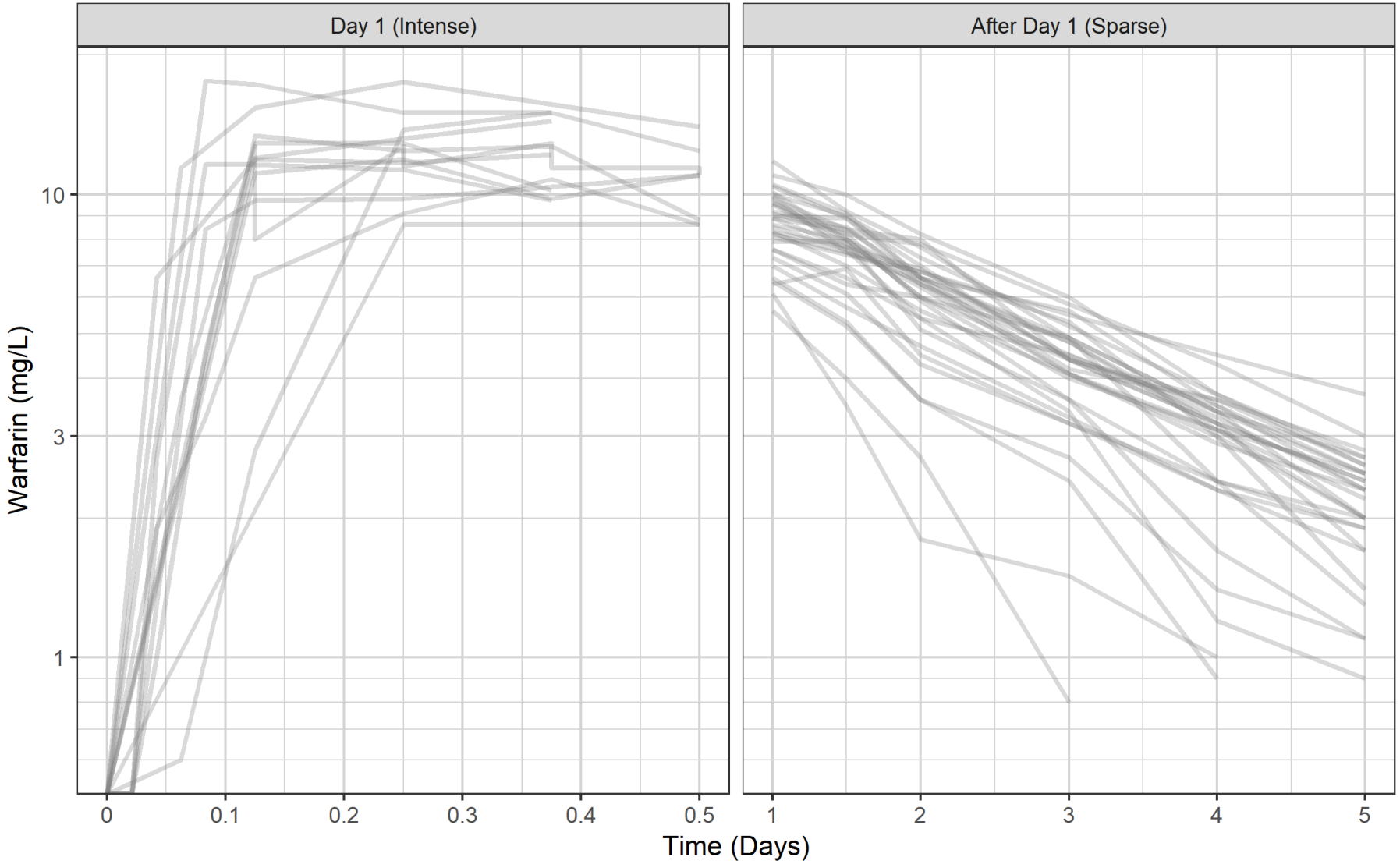
The data set has two types of profiles…



...but you can also group by rich day 1 and sparse later days...



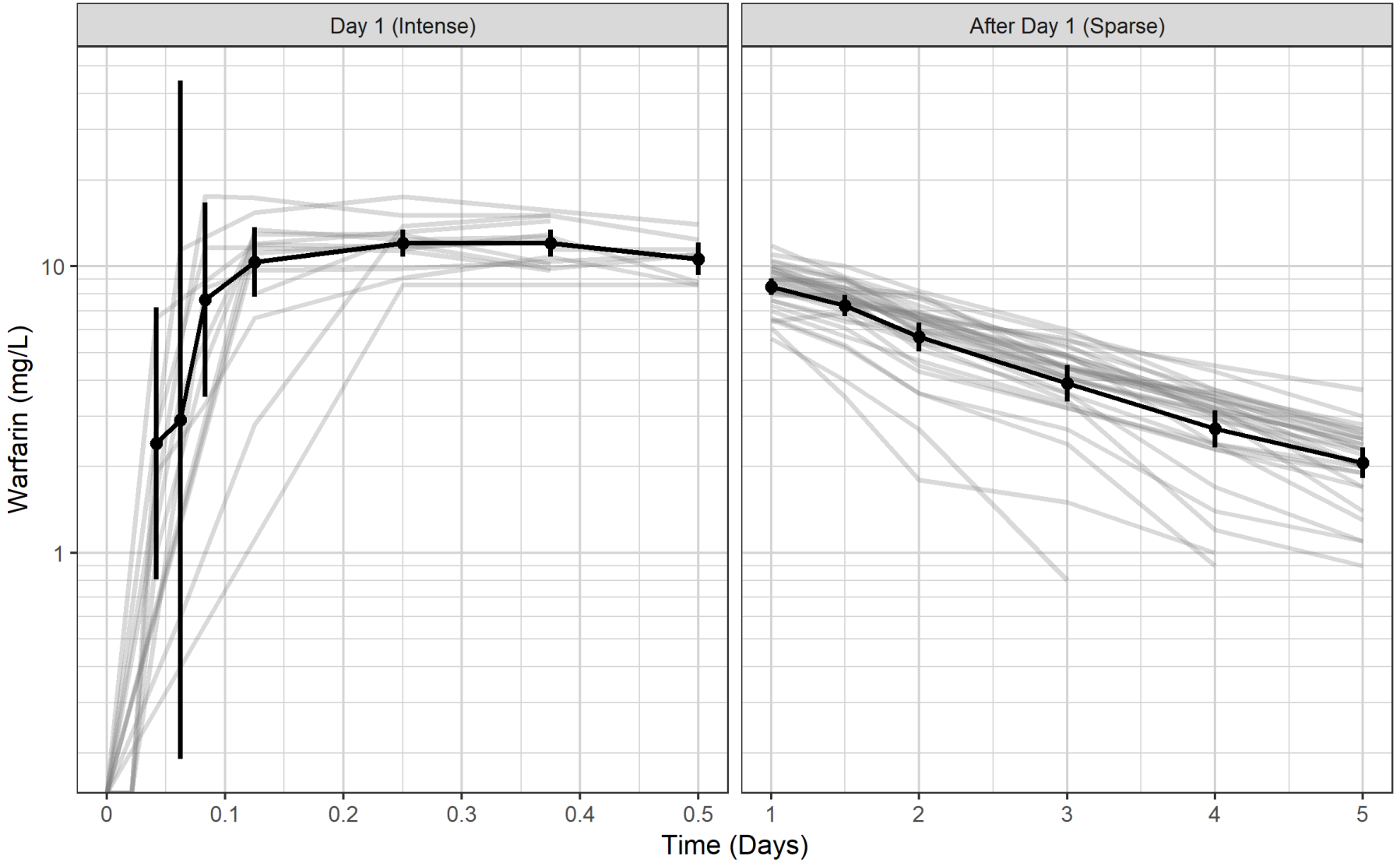
Switch to semi-log scale using xgx helper `xgx_scale_y_log10()`
Any clues to what model we should use?



nlmixr²

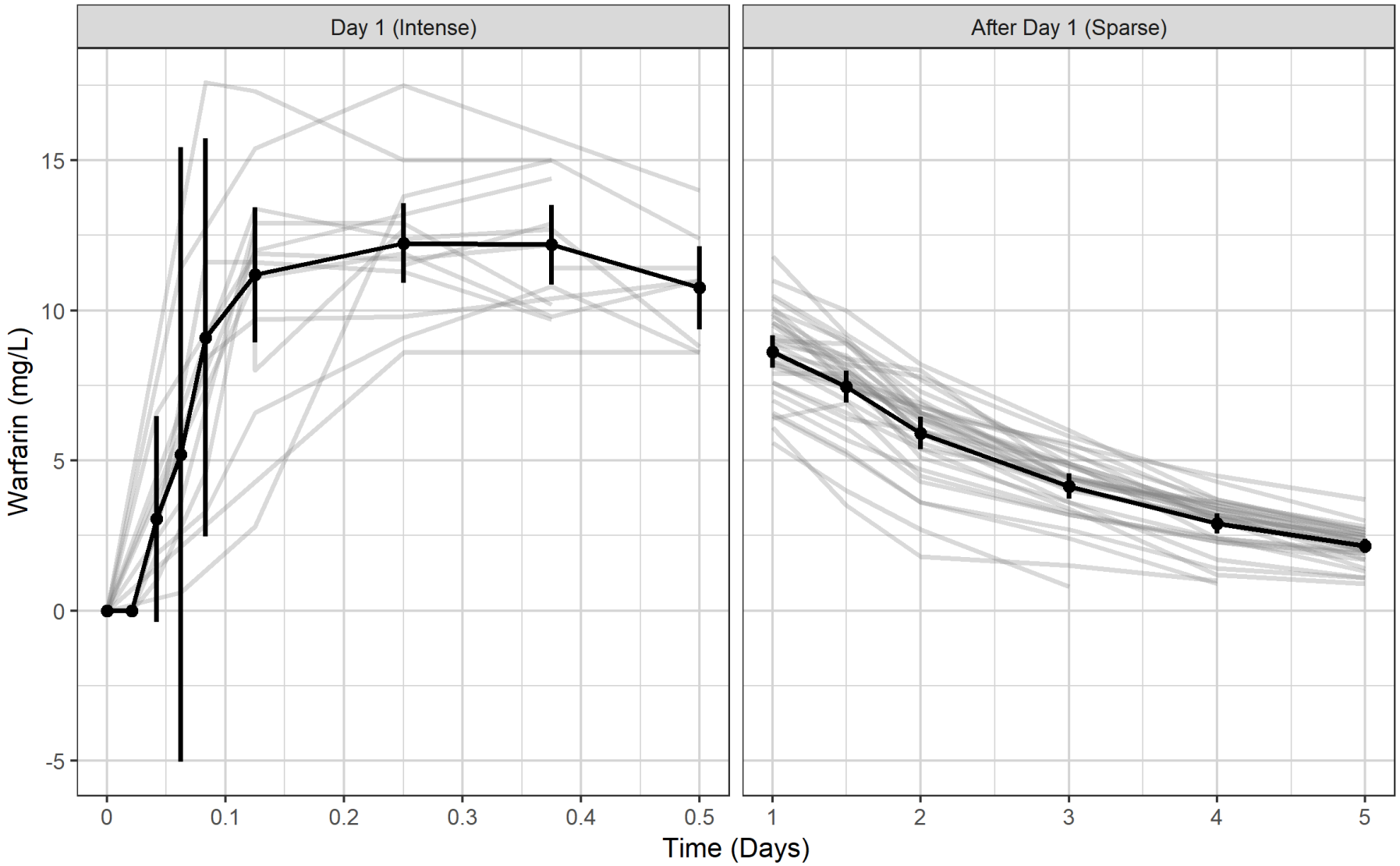
xgx can also add nice summary information if data has nominal times: summaries of mean plus 95% CI

```
xgx_geom_ci(aes(x = TIME, color = NULL, group = NULL, shape = NULL), conf level = 0.95)
```



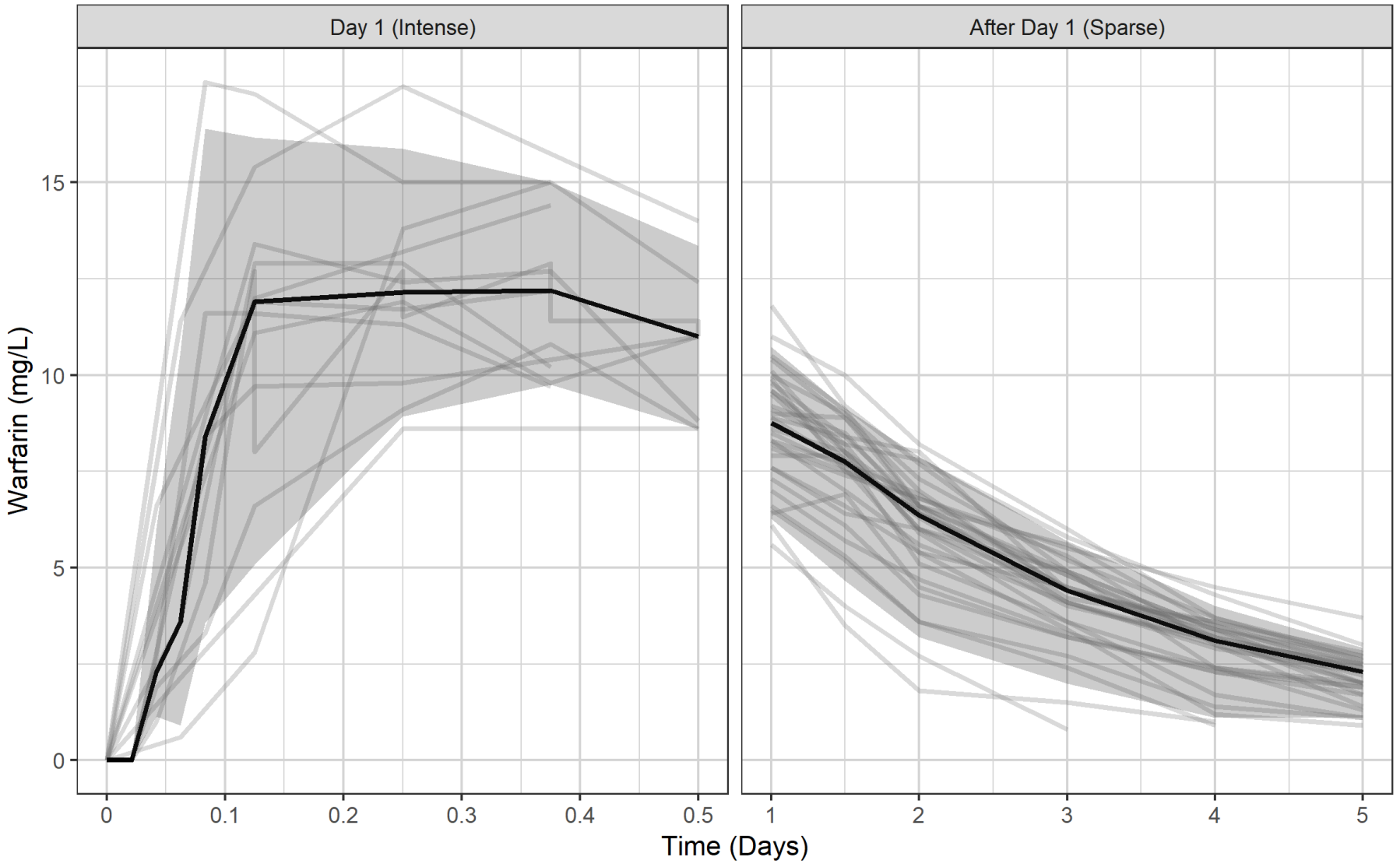
nlmixr²

On linear scale this would result in a CI crossing zero because CIs are assumed symmetrical



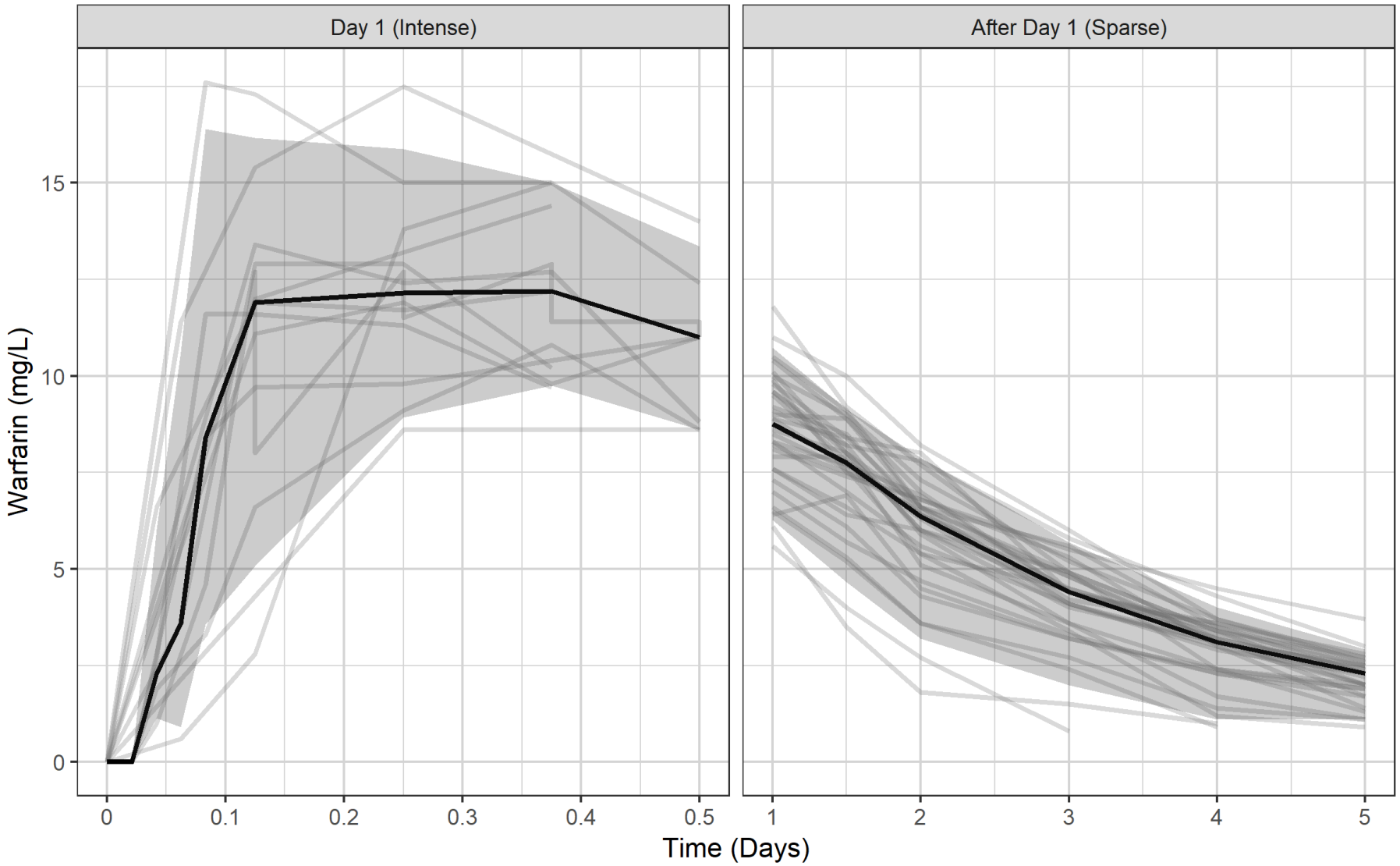
...so perhaps a median and 95% of the data would be more suitable

```
xgx_geom_pi(aes(x = TIME, color = NULL, group = NULL, shape = NULL)
```



nlmixr²

There appears to be a delay in absorption…



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:

nlmixr²!

nlmixr^2 is an open-source R package

- Written by Wenping Wang and Matt Fidler, and available on CRAN[a] and GitHub[b]:
  - builds on rxode2
  - combined with SAEM, and FOCEi estimation routines, provides an R package for parameter estimation in nonlinear mixed effect models
  - under very active development!
- nlmixr^2 is completely free and open source, and does not depend on any other commercial tool such as NONMEM or Monolix
- nlmixr^2 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

[a] https://cran.r-project.org/web/packages/nlmixr2
[b] https://github.com/nlmixr2/nlmixr2

nlmixr² is a well-embedded open-source R package

- nlmixr² is fully supported by ggPMX[a]
- Additionally, xpose.nlmixr2[b] written by Justin Wilkins provides linkage to the new xpose package[c], written by Ben Guiastrennec, feeding the uniform output into a highly flexible diagnostics package
- The shinyMixR[d] project management tool written by Richard Hooijmaijers and Teun Post provides an interface to nlmixr² from both the R command line and a user-friendly browser-based Shiny dashboard application
- nlmixr² requires access to compilers, easily installed using Rtools on Windows[e]
- Installation is very simple from CRAN:

```
install.packages("nlmixr2",dependencies=TRUE)
```

- Documentation is available in the form of a bookdown (nlmixr2.org)[f] with a dedicated nlmixr blog (blog.nlmixr2.org)
- Runs on Windows, Linux, and OS X, and a Docker install is available

[a] https://cran.r-project.org/web/packages/ggPMX/ and https://github.com/ggPMXdevelopment/ggPMX
[b] https://github.com/nlmixr2/xpose.nlmixr2 and https://cran.r-project.org/packages/xpose.nlmixr2
[c] https://cran.r-project.org/packages/xpose and https://uupharmacometrics.github.io/xpose/
[d] https://github.com/RichardHooijmaijers/shinyMixR and https://richardhooijmaijers.github.io/shinyMixR/articles/shinyMixR-vignette.html
[e] https://cran.r-project.org/bin/windows/Rtools/rtools42/rtools.html
[f] https://nlmixr2.github.io/nlmixr2

… and nlmixr² has some amazing new features!

- nlmixr² can now use NONMEM for its parameter estimation when using the babelmixr2[a] package
- Additionally, babelmixr2 can use the PKNCA package to provide starting values for PK analysis when rich profiles are available, and can keep track of units
- The nonmem2rx[b] package allows reading in of NONMEM output and translation of NONMEM models (with a single epsilon error component) to rxode2 and nlmixr² objects
- This allows checking of NONMEM models, additional calculation of covariance matrices where NONMEM failed, and nlmixr² functionality like creation of VPCs and smooth predicted individual profiles

[a] https://cran.r-project.org/web/packages/babelmixr2/index.html
[b] https://nlmixr2.github.io/nonmem2rx/index.html

… and nlmixr^2 has some more amazing new features!

- Just released: nlmixr^2 is now at version 3.0 on CRAN hugely improving the stability of the installation[a]!

- The monolix2rx[b] package allows reading in of Monolix output and translation of Monolix models to rxode2 and nlmixr^2 objects
- Now using babelmixr2 you can import Monolix and NONMEM models to do optimal design with PopED[c]

[a] https://blog.nlmixr2.org/blog/2024-09-18-nlmixr2-3.0.0-release/
[b] https://cran.r-project.org/web/packages/nonmem2rx/index.html
[c] https://cran.r-project.org/web/packages/PopED/index.html

nlmixr performance: do I get the same results when I switch from my usual estimation tool to nlmixr?

- Publication in CPT:PSP: *Performance of the SAEM and FOCEI algorithms in the open-source non-linear mixed effect modelling tool nlmixr*[a]

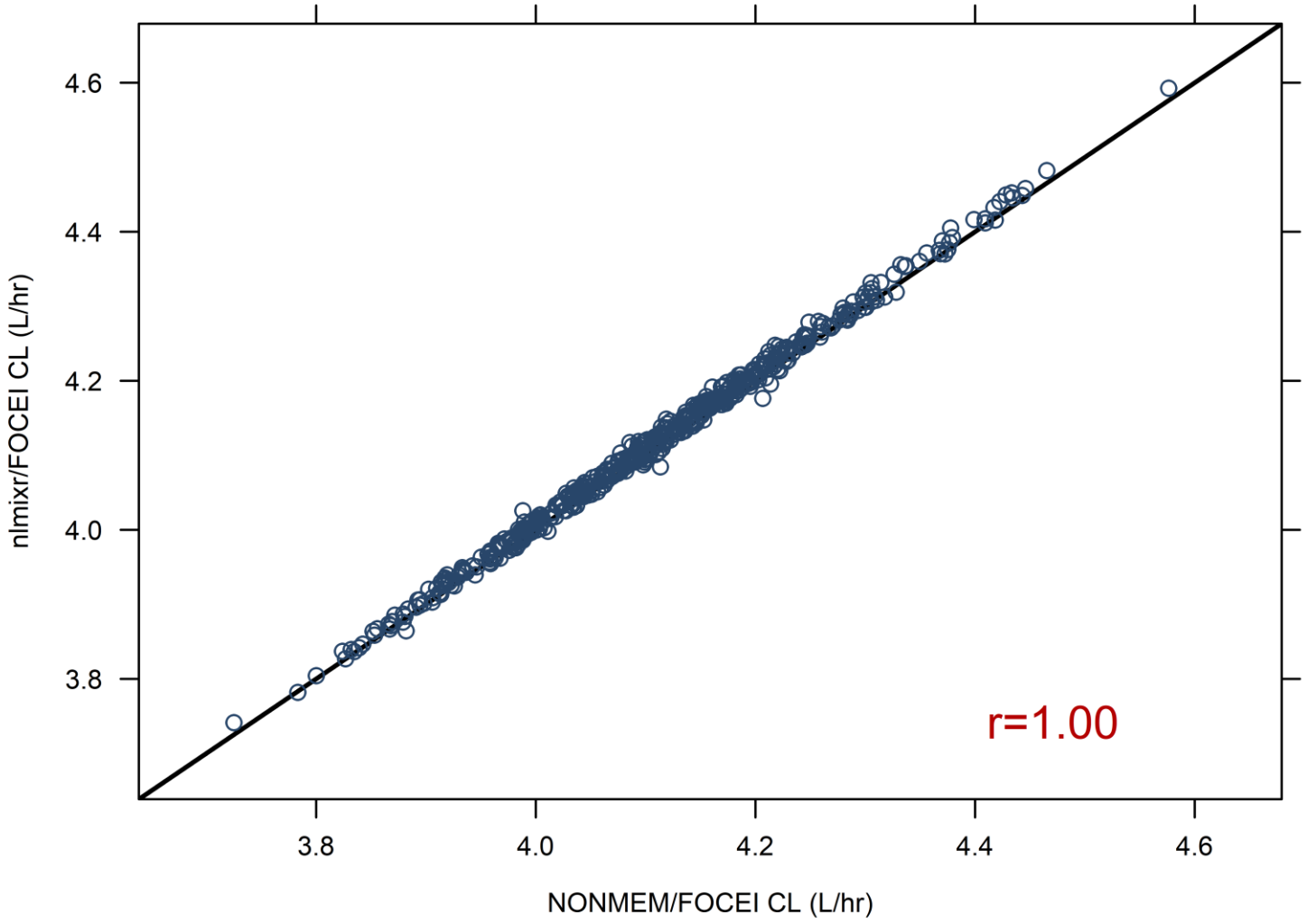
- nlmixr/FOCEi compared to NONMEM/FOCEi
- nlmixr/SAEM compared to Monolix/SAEM

- Repeated sparse data sets for a single model
- A wide range of models and inputs using single rich data sets

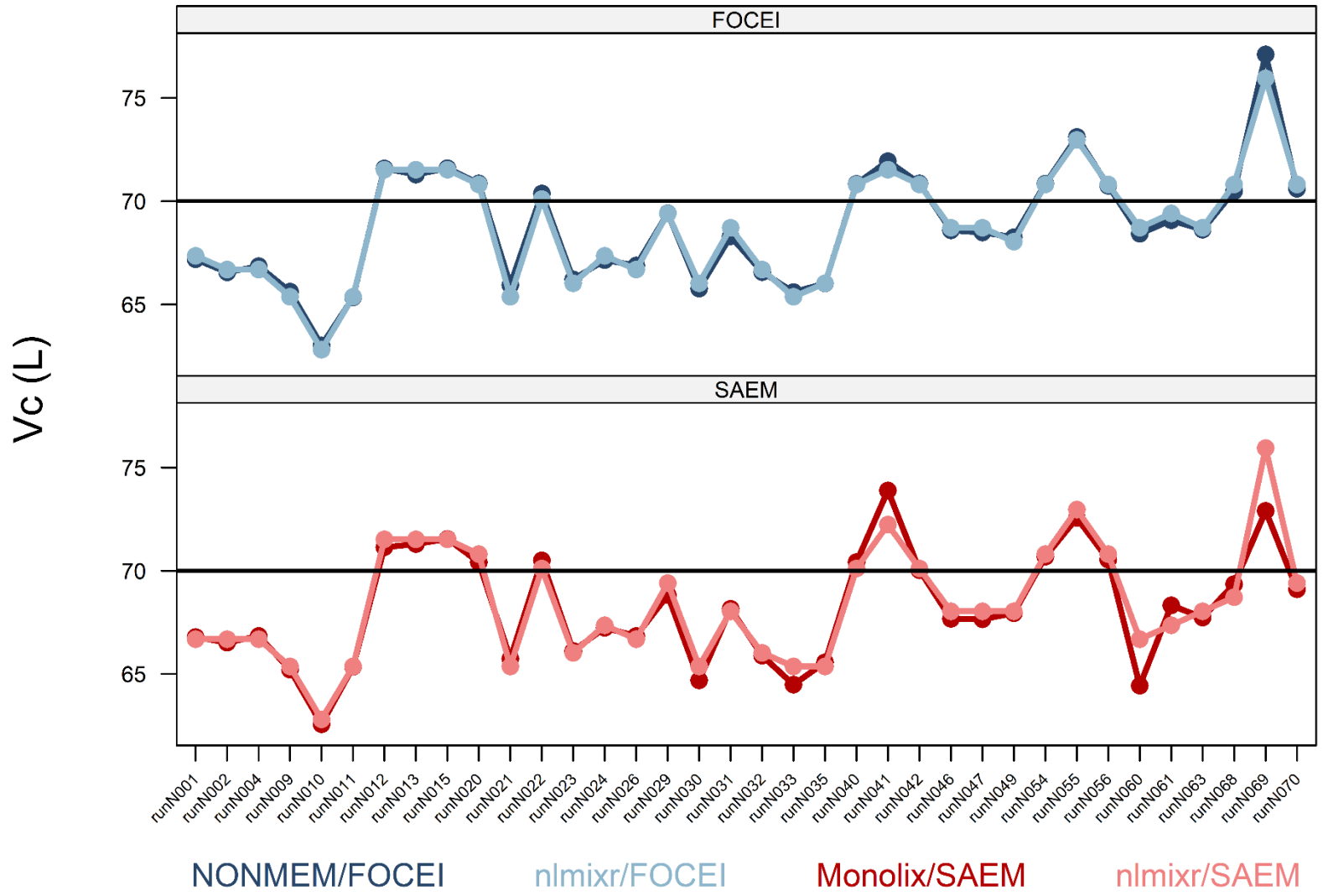
- Results for both FOCEi and SAEM are perfectly comparable to NONMEM and Monolix

[a] https://ascpt.onlinelibrary.wiley.com/doi/10.1002/psp4.12471

500 sparse data sets are analysed using both NONMEM/FOCEi and nlmixr/FOCEi. Each marker is a single paired data set result for clearance for NONMEM on the x-axis and nlmixr/FOCEi on the y-axis: results are highly correlated (r=1.00), and lie on the line of identity (diagonal black line)



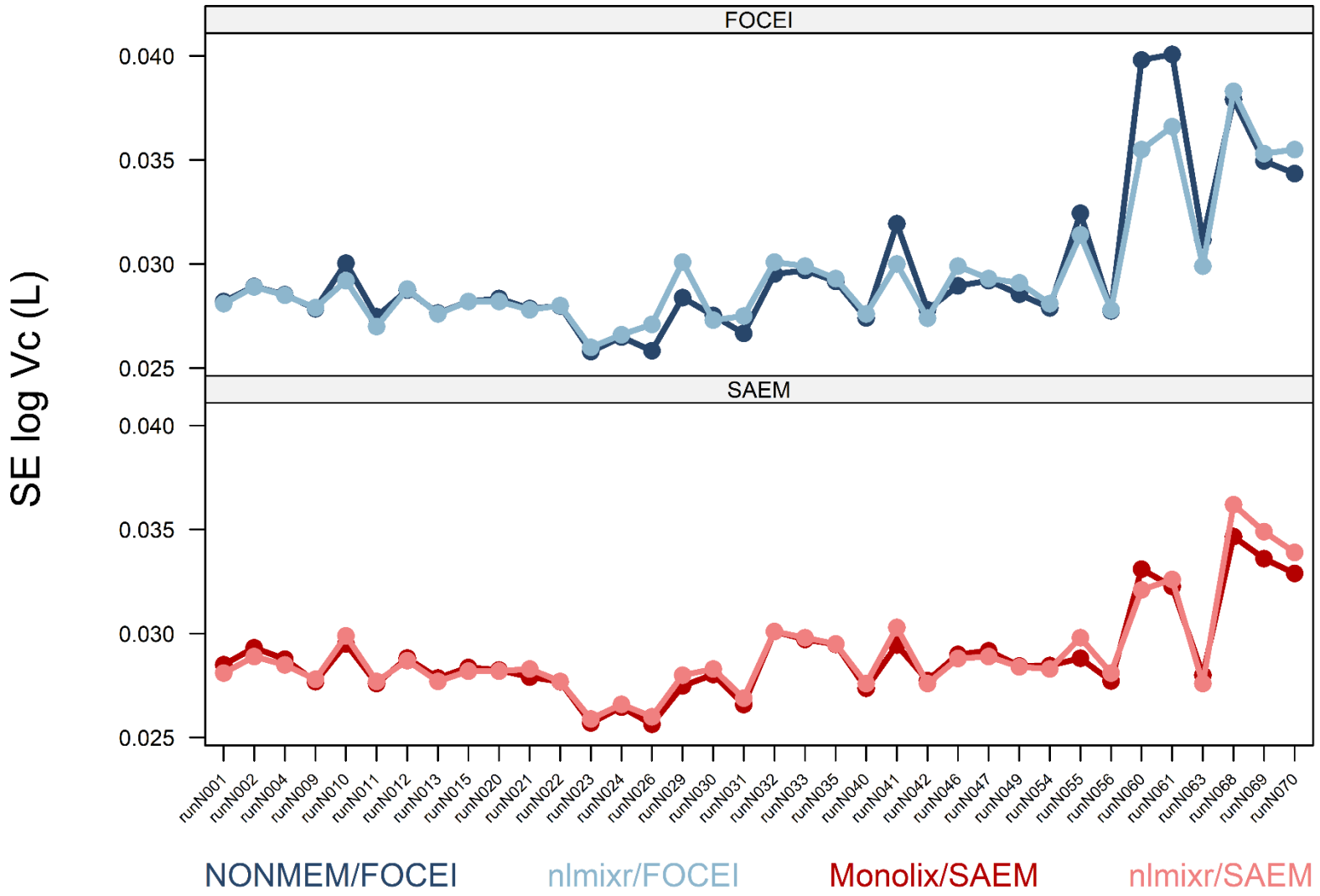
Central volume (Vc): Top panel: FOCEi estimates, bottom panel: SAEM estimates
Dark lines: NONMEM/MONOLIX, light lines: nlmixr



SE of theta estimates for Vc: very good match across software packages

Top panel: FOCEi estimates, bottom panel: SAEM estimates

Dark lines: NONMEM/MONOLIX, light lines: nlmixr



Defining nlmixr^2 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^2 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 for thetas and errors (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
    eta.cl ~ 0.1
    eta.v  ~ 0.1
  })
  model({

})
}
```

Defining nlmixr^2 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^2 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
    eta.cl ~ 0.1
    eta.v  ~ 0.1
})
  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^2 models: check the model code

```
## Check the model and some of the assumptions made by nlmixr

nlmixr2(One.comp.KA.solved)
```

```
> nlmixr2(One.comp.KA.solved)
i parameter labels from comments will be replaced by 'label()'
 ── rxode2-based solved PK 1-compartment model with first-order absorption ──────────────────────
 ── Initalization: ──
Fixed Effects ($theta):
       lka        lcl         lv   prop.err    add.err
 0.1397619 -2.0024805  2.0794415  0.1500000  0.6000000

Omega ($omega):
       eta.ka eta.cl eta.v
eta.ka    0.5    0.0   0.0
eta.cl    0.0    0.1   0.0
eta.v     0.0    0.0   0.1
 ── μ-referencing ($muRefTable): ──
  theta    eta level
1   lcl eta.cl    id
2    lv  eta.v    id
3   lka eta.ka    id
```

Running nlmixr² models: check the model code

```
## Check the model and some of the assumptions made by nlmixr2

nlmixr2(One.comp.KA.solved)
```

```
 ── Model (Normalized Syntax): ──
function() {
    ini({
        lka <- 0.139761942375159
        label("log ka (1/h)")
        lcl <- -2.00248050054371
        label("log Cl (L/h)")
        lv <- 2.07944154167984
        label("log V (L)")
        prop.err <- c(0, 0.15)
        label("proportional error (SD/mean)")
        add.err <- c(0, 0.6)
        label("additive error (mg/L)")
        eta.ka ~ 0.5
        eta.cl ~ 0.1
        eta.v ~ 0.1
    })
    model({
        cl <- exp(lcl + eta.cl)
        v <- exp(lv + eta.v)
        ka <- exp(lka + eta.ka)
        linCmt() ~ prop(prop.err) + add(add.err)
    })
}
```

Running nlmixr² models with the nlmixr2 command

```
## estimate parameters using nlmixr:
run001S <-
  nlmixr2(
    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
                 npde = TRUE)  #and npde values for diagnostics
 )

## results are stored in the nlmixr2 object and can be viewed:
run001S
```

nlmixr² output for SAEM

```
> run001s
── nlmixr² SAEM OBJF by FOCEi approximation ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 454.9677 932.2748 960.4784      -458.1374        19.65728         1.52084

── Time (sec run001s$time): ──

        setup optimize covariance saem table compress
elapsed 0.001    0.001      0.001 9.42 11.07     0.07

── Population Parameters (run001s$parFixed or run001s$parFixedDf): ──

                           Parameter   Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lka                     log ka (1/h) -0.605  0.205   34    0.546 (0.365, 0.817)     54.6      46.5%
lcl                     log Cl (L/h)  -1.99 0.0492 2.47     0.136 (0.124, 0.15)     23.4      15.5%
lv                         log V (L)   2.05 0.0473  2.3       7.77 (7.08, 8.52)     16.2      35.3%
prop.err proportional error (SD/mean)   0.17                              0.17
add.err         additive error (mg/L)  0.818                             0.818

  Covariance Type (run001s$covMethod): linFim
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run001s$omega) or correlation (run001s$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run001s$shrink
  Censoring (run001s$censInformation): No censoring

── Fit Data (object run001s is a modified tibble): ──
# A tibble: 251 × 26
  ID     TIME    DV EPRED  ERES   NPDE   NPD    PDE     PD  PRED   RES  WRES IPRED   IRES  IWRES CPRED  CRES CWRES eta.ka eta.cl
  <fct> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
1 1       0.5   0    3.57 -3.57  0.643 -2.33 0.74   0.01    3.06 -3.06 -1.76  1.45 -1.45  -1.69   2.56 -2.56 -2.28 -0.724  0.413
2 1       1     1.9  5.70 -3.80 -0.117 -1.79 0.453  0.0367  5.36 -3.46 -1.36  2.69 -0.794 -0.846  4.66 -2.76 -1.74 -0.724  0.413
3 1       2     3.3  8.49 -5.19 -1.71  -2.05 0.0433 0.02    8.38 -5.08 -1.61  4.70 -1.40  -1.22   7.74 -4.44 -1.96 -0.724  0.413
# i 248 more rows
# i 6 more variables: eta.v <dbl>, cl <dbl>, v <dbl>, ka <dbl>, tad <dbl>, dosenum <dbl>
# i Use `print(n = ...)` to see more rows
```

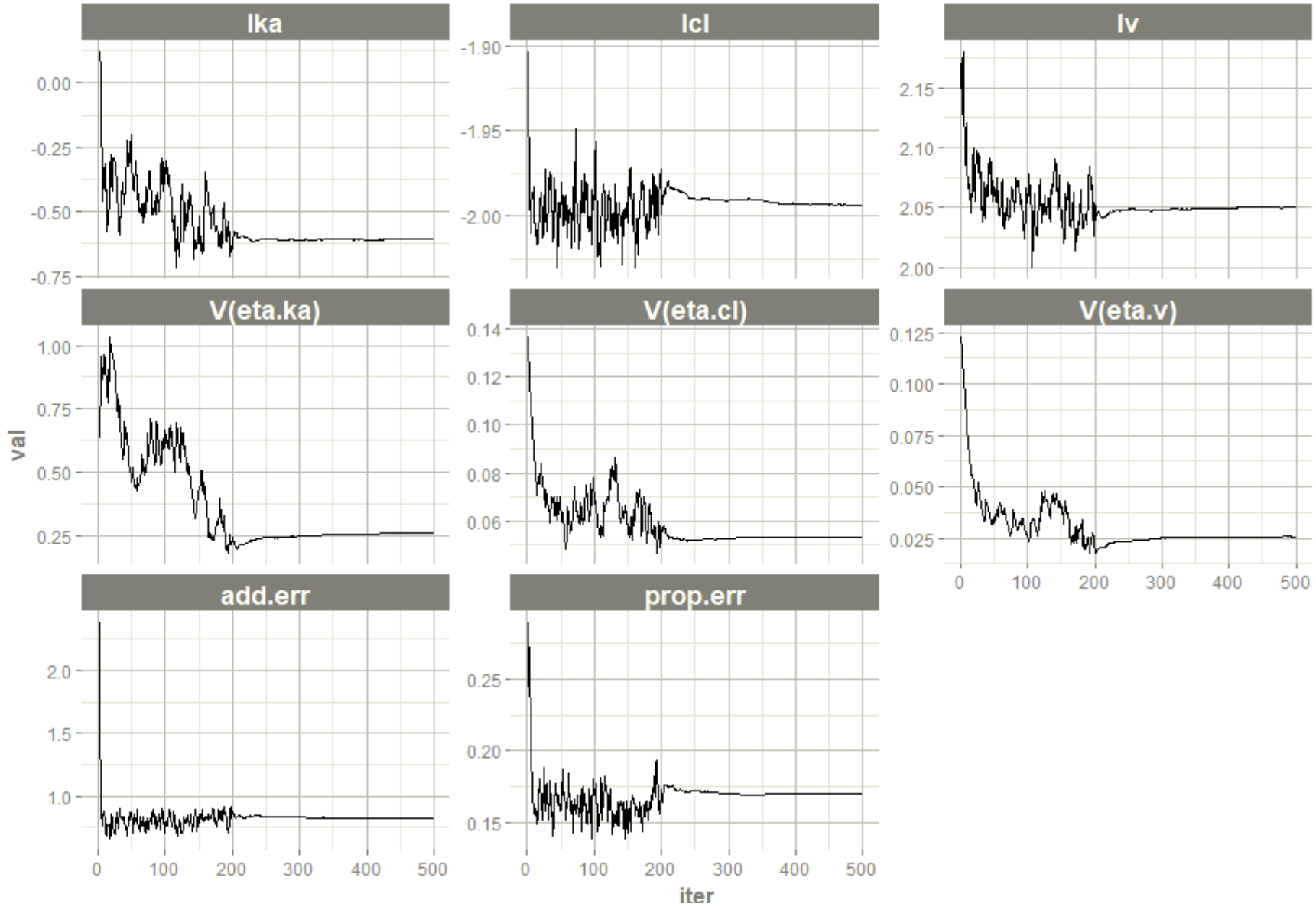
The labels in the ini block end up in the output, and the log-transformed parameters are returned with a back-transformation and 95%CIs

```
> run001s
── nlmixr² SAEM OBJF by FOCEi approximation ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 454.9677 932.2748 960.4784      -458.1374        19.65728         1.52084

── Time (sec run001s$time): ──

        setup optimize covariance saem table compress
elapsed 0.001    0.001      0.001 9.42 11.07     0.07

── Population Parameters (run001s$parFixed or run001s$parFixedDf): ──

                            Parameter   Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lka                     log ka (1/h) -0.605  0.205    34    0.546 (0.365, 0.817)     54.6      46.5%
lcl                     log Cl (L/h)  -1.99 0.0492 2.47     0.136 (0.124, 0.15)     23.4      15.5%
lv                        log V (L)   2.05 0.0473  2.3       7.77 (7.08, 8.52)     16.2      35.3%
prop.err proportional error (SD/mean)   0.17                                0.17
add.err          additive error (mg/L)  0.818                               0.818

  Covariance Type (run001s$covMethod): linFim
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run001s$omega) or correlation (run001s$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run001s$shrink
  Censoring (run001s$censInformation): No censoring

── Fit Data (object run001s is a modified tibble): ──
# A tibble: 251 × 26
  ID     TIME    DV EPRED  ERES   NPDE   NPD    PDE     PD  PRED   RES  WRES IPRED   IRES  IWRES CPRED  CRES CWRES eta.ka eta.cl
  <fct> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
1 1       0.5   0    3.57 -3.57  0.643 -2.33 0.74    0.01    3.06 -3.06 -1.76  1.45 -1.45  -1.69   2.56 -2.56 -2.28 -0.724  0.413
2 1       1     1.9  5.70 -3.80 -0.117 -1.79 0.453   0.0367  5.36 -3.46 -1.36  2.69 -0.794 -0.846  4.66 -2.76 -1.74 -0.724  0.413
3 1       2     3.3  8.49 -5.19 -1.71  -2.05 0.0433  0.02    8.38 -5.08 -1.61  4.70 -1.40  -1.22   7.74 -4.44 -1.96 -0.724  0.413
# i 248 more rows
# i 6 more variables: eta.v <dbl>, cl <dbl>, v <dbl>, ka <dbl>, tad <dbl>, dosenum <dbl>
# i Use `print(n = ...)` to see more rows
```

Running nlmixr2 models: save the object, and examine parameter trace plots when using SAEM to check convergence

```
## results are stored in the nlmixr2 object and can be viewed:
run001S

## and saved for future use or reference:
save(run001S, file = "run001S.Rdata")

## and for SAEM, convergence can be checked using a parameter trace plot:
traceplot(run001S)
```

Traceplot for SAEM parameter estimates using `traceplot` command

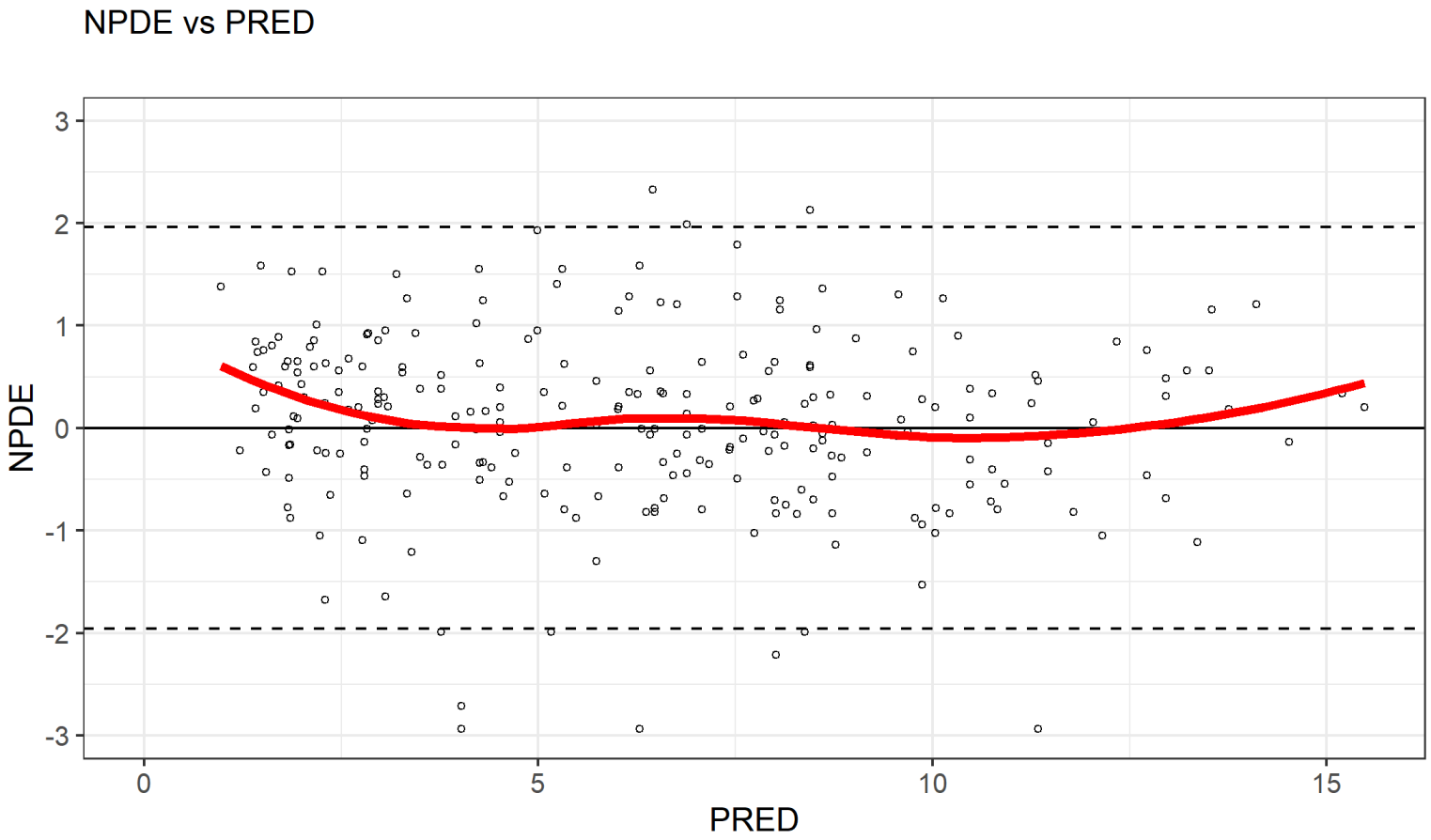


nlmixr^2 is linked to ggPMX

```
# Model Diagnostics with ggPMX
# The controller is first constructed
ctr001S <- pmx_nlmixr(run001S,
                  conts = c("WT","AGE"),
                  cats=c("SEX","SPARSE"),
                  vpc=FALSE,
                  settings=pmx_settings(is.draft=FALSE))

# and can then be piped into a specific plot
# syntax for npde vs pred plot
ctr001S %>% pmx_plot_npde_pred
#alternatively:
pmx_plot_npde_pred(ctr001S)
```

nlmixr² is linked to ggPMX

```
# Model Diagnostics with ggPMX
# The controller is first constructed
ctr001S <- pmx_nlmixr(run001S,
                      conts = c("WT","AGE"),
                      cats=c("SEX","SPARSE"),
                      vpc=FALSE,
          settings=pmx_settings(is.draft=FALSE))

# and can then be piped into a specific plot
# syntax for npde vs pred plot
ctr001S %>% pmx_plot_npde_pred
#alternatively:
pmx_plot_npde_pred(ctr001S)
```

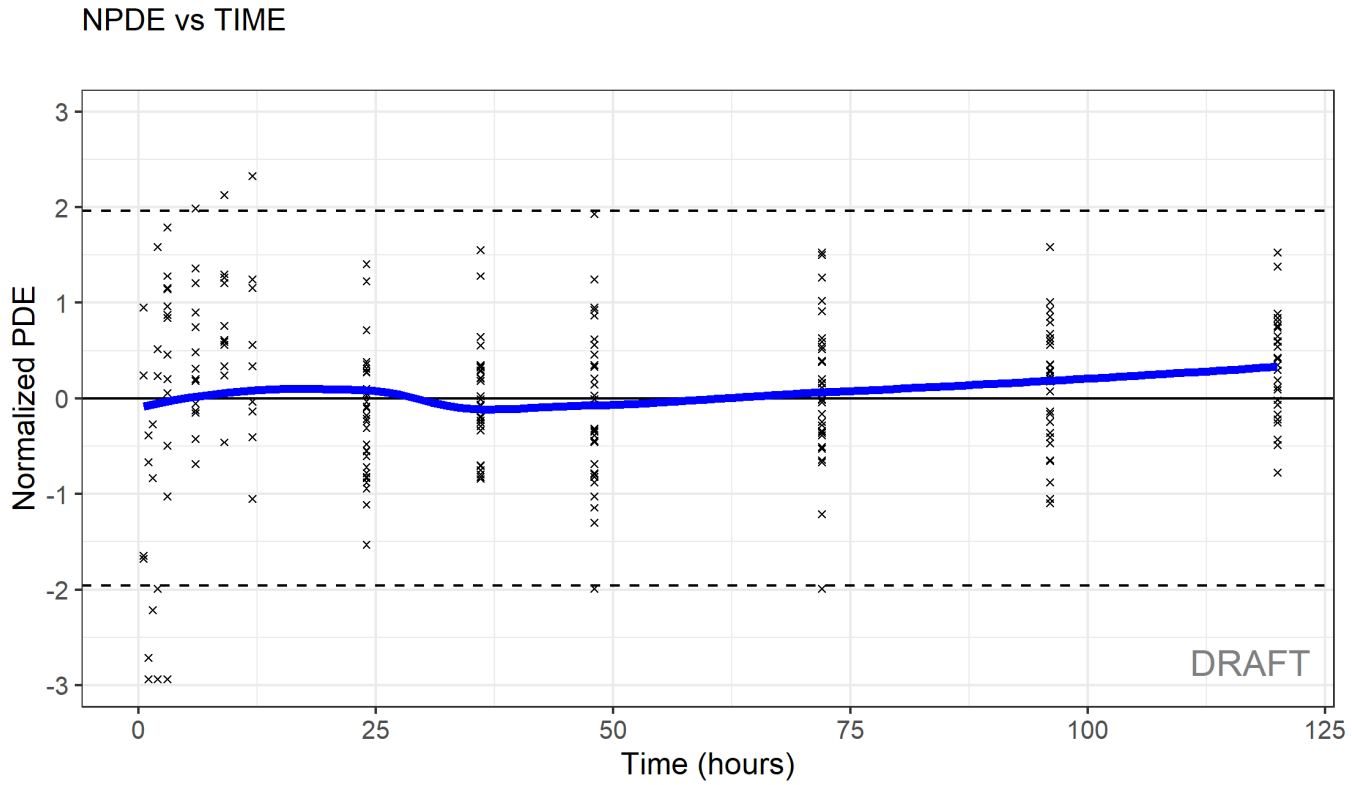


nlmixr² is linked to ggPMX

```
## Modify graphical options and add DRAFT label:
ctr001S %>% pmx_plot_npde_time(smooth = list(color="blue"),
                               point = list(shape=4),
                               s.draft=TRUE,
                               labels = list(x = "Time (hours)",
                                             y = "Normalized PDE"))
```

nlmixr² is linked to ggPMX

```
## Modify graphical options and add DRAFT label:
ctr001S %>% pmx_plot_npde_time(smooth = list(color="blue"),
                               point = list(shape=4),
                               s.draft=TRUE,
                               labels = list(x = "Time (hours)",
                                             y = "Normalized PDE"))
```



nlmixr² is linked to ggPMX

```
## DV vs IPRED plot
ctr001S %>% pmx_plot_dv_ipred(scale_x_log10=TRUE, scale_y_log10=TRUE)

#You can filter to restrict the values of IPRED for instance:
ctr001S %>% pmx_plot_dv_ipred(scale_x_log10=TRUE, scale_y_log10=TRUE,filter=(IPRED>1))

## DV vs PRED plot
ctr001S %>% pmx_plot_dv_pred(scale_x_log10=TRUE, scale_y_log10=TRUE)

## Absolute individual weighted residuals to investigate the residual error model
ctr001S %>% pmx_plot_abs_iwres_ipred
#again, alternatively:
#pmx_plot_abs_iwres_ipred(ctr)


ctr001S %>% pmx_plot_iwres_dens
ctr001S %>% pmx_plot_eta_qq
ctr001S %>% pmx_plot_eta_box
ctr001S %>% pmx_plot_eta_hist
ctr001S %>% pmx_plot_eta_matrix
```

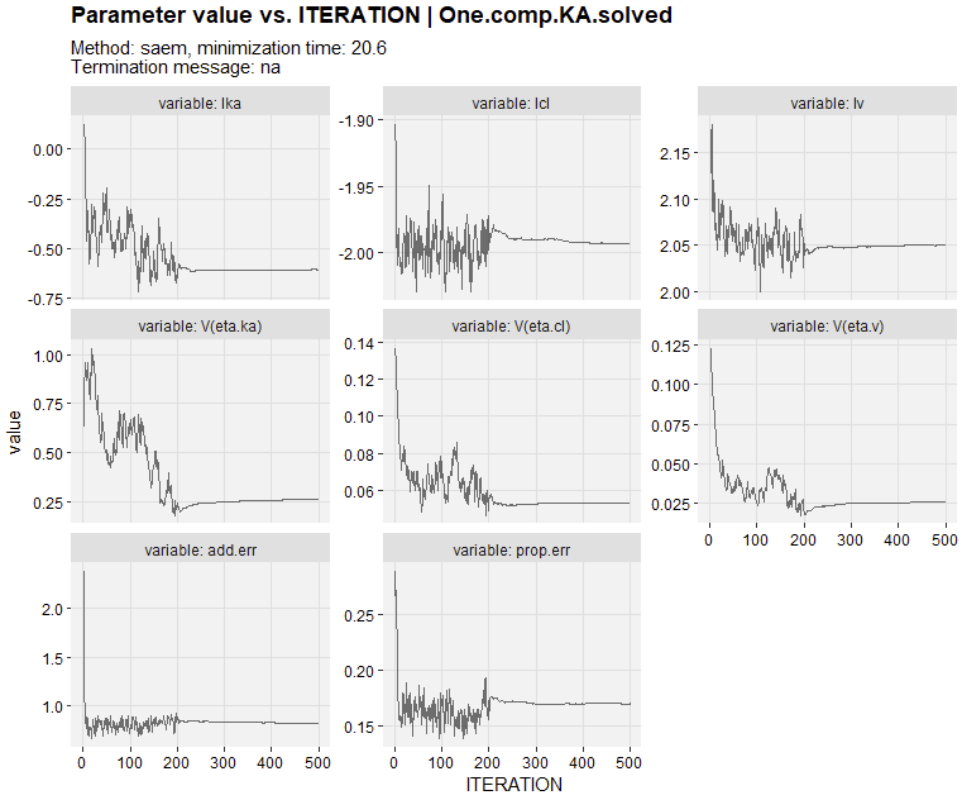
nlmixr^2 is linked to Ben Guiastrennec's xpose* package that uses ggplot2

```
## 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(run001S)

## 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/

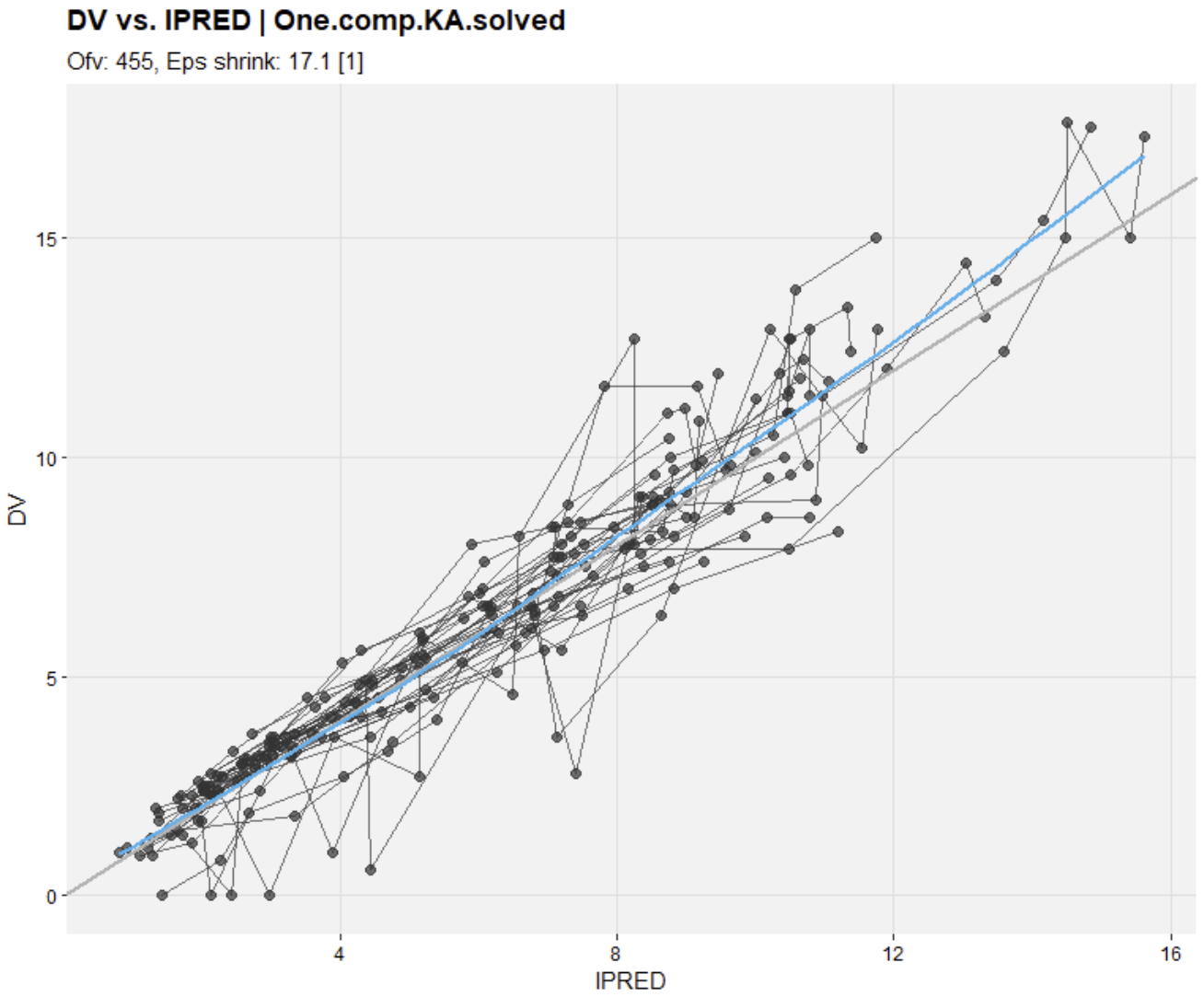
nlmixr² is linked to Ben Guiastrennec's xpose* package that uses ggplot2

```
## the nlmixr2 object can be transformed into an xpose object to allow diagnostics with the new xpose package
## the link between nlmixr2 and xpose is provided by the xpose.nlmixr2 package
## only xpose_data_nlmixr is from xpose.nlmixr2
## all further commands (see cheatsheet)  are from the xpose package

xpdb.1s <- xpose_data_nlmixr(run001S)

## 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/

DV vs IPRED using xpose



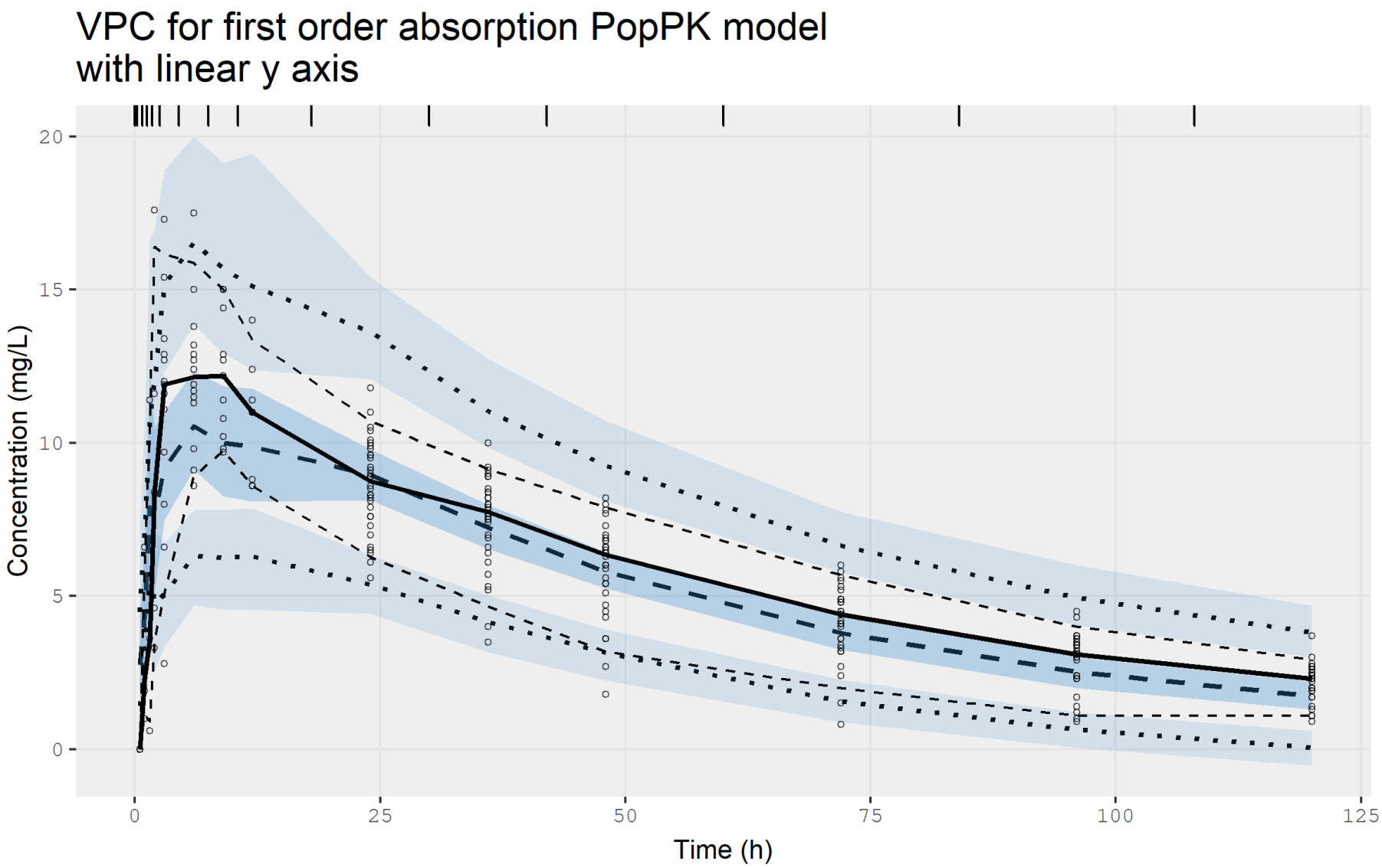

```
xpdb.1s <- xpose_data_nlmixr(run001S)
## dv vs ipred plot:
dv_vs_ipred(xpdb.1s,
            caption = NULL)
```

nlmixr² is linked to Ron Keizer's vpc* package

```
## nlmixr2 comes with its own built-in vpc functionality that uses Ron Keizer's vpc package
## see the cheatsheet for further options

## because the data set uses nominal time points, it is nice to have the bins surround these time points
## so that each time point falls in a bin

bin_mids <- sort(unique(PKdata$TIME))
bin_edges <- bin_mids - c(0, diff(bin_mids) / 2)

vpcPlot(
  run001S,                        #the nlmixr object
  n = 500,                        #number of trials simulated
  bins = bin_edges,
  show = list(obs_dv = TRUE,      #additional items to show, like the observations
              obs_median = TRUE,
              sim_median = TRUE,
              sim_median_ci = TRUE,
              obs_ci = TRUE,
              pi = TRUE
  ),
  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/

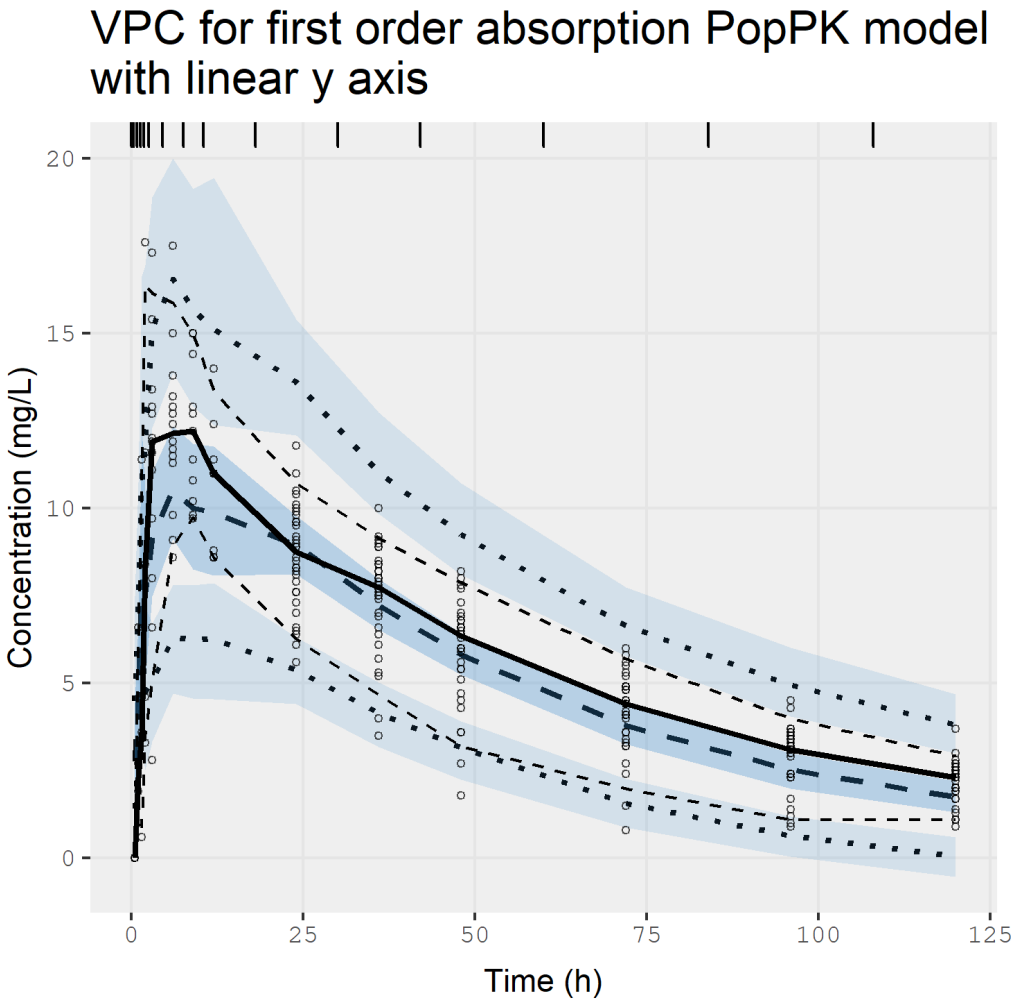
VPC for the base model on linear scale...

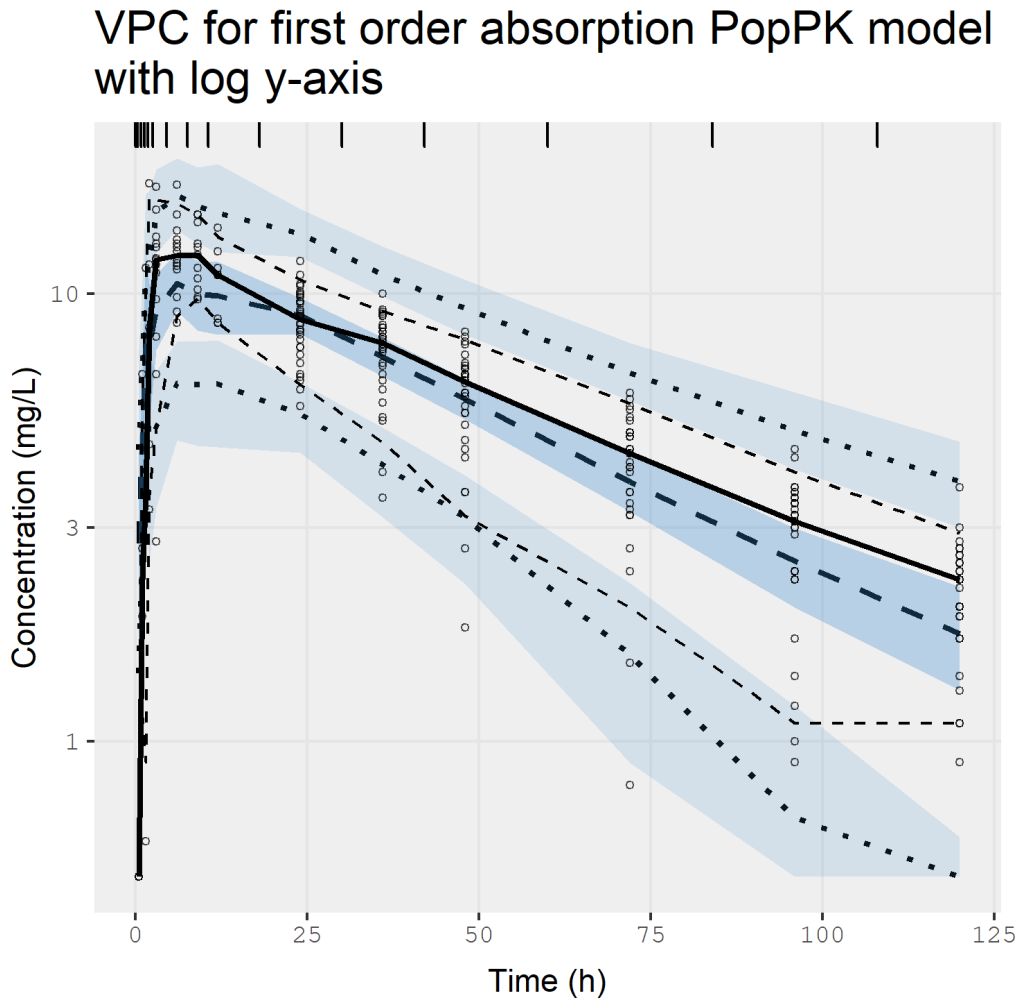


...and on log scale

```
## or with a log y-axis starting at 0.5
vpcPlot(
  run001S,                        #the nlmixr object
  n = 500,                        #number of trials simulated
  bins = bin_edges,
  show = list(obs_dv = TRUE,      #additional items to show, like the observations
              obs_median = TRUE,
              sim_median = TRUE,
              sim_median_ci = TRUE,
              obs_ci = TRUE,
              pi = TRUE
  ),
  xlab = "Time (h)",              #x-axis label
  ylab = "Concentration (mg/L)",  #y-axis label
  title = "VPC for first order absorption PopPK model"
  log_y = TRUE,            #to request a log y-axis
  log_y_min = 0.5          #starting at 0.5
)
```

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





Alternatively using direct Ron Keizer code so you only need to simulate once

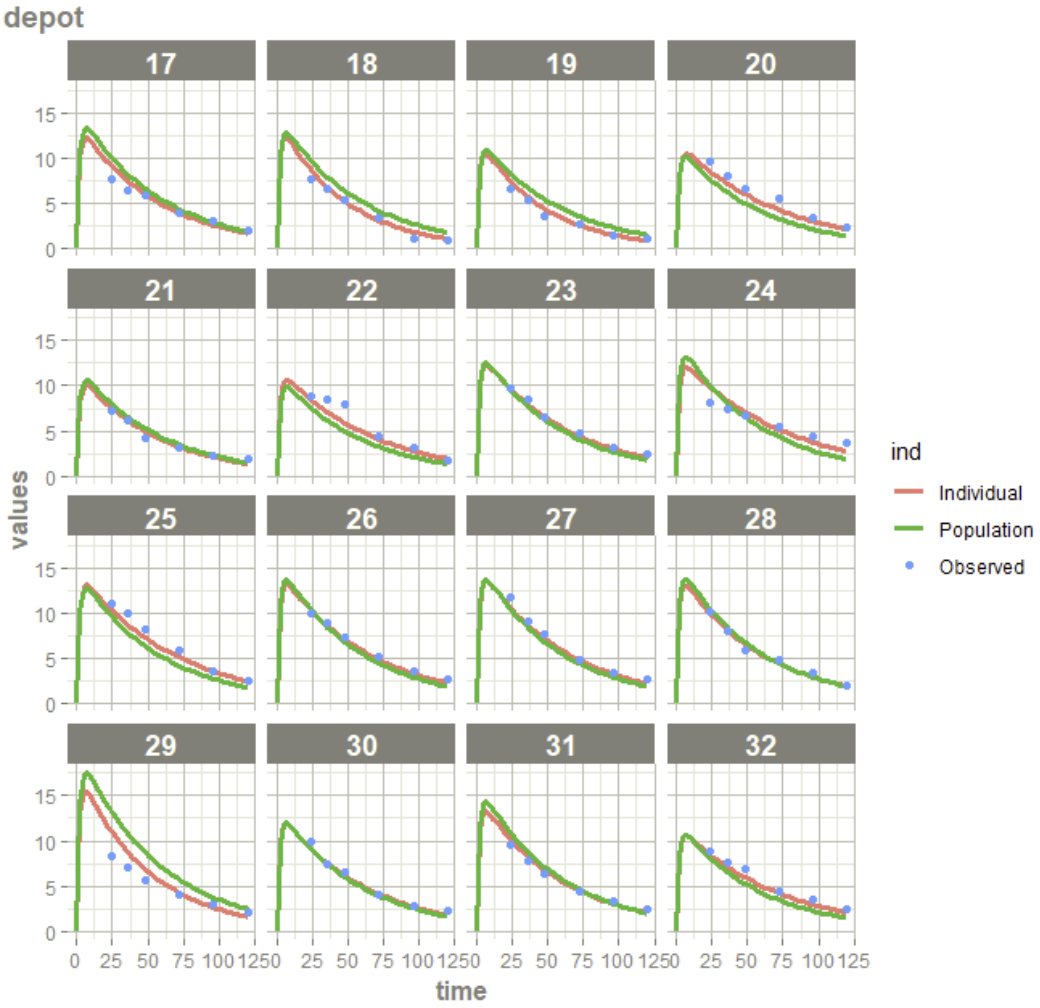
```
library(vpc)

SimVPC <- vpcSim(run001S,                ## the nlmixr2 object
                 n = 500)

setnames(SimVPC, c("id","time","sim"), c("ID","TIME","DV"))

vpc_vpc(
  sim  = SimVPC,
  obs  = Pkdata,
  bins = bin_edges,
  show = list(obs_dv = TRUE,
              obs_median = TRUE,
              sim_median = TRUE,
              sim_median_ci = TRUE,
              obs_ci = TRUE,
              pi = TRUE
  ),
  xlab = "Time (h)",               #x-axis label
  ylab = "Concentration (mg/L)",   #y-axis label
  title = "VPC for first order absorption PopPK model\nwith linear y axis"
)
```

nlmixr² can generate individual graphs using `augPred`

```
## Individual fits can 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
```

nlmixr² can generate individual graphs using `augPred`

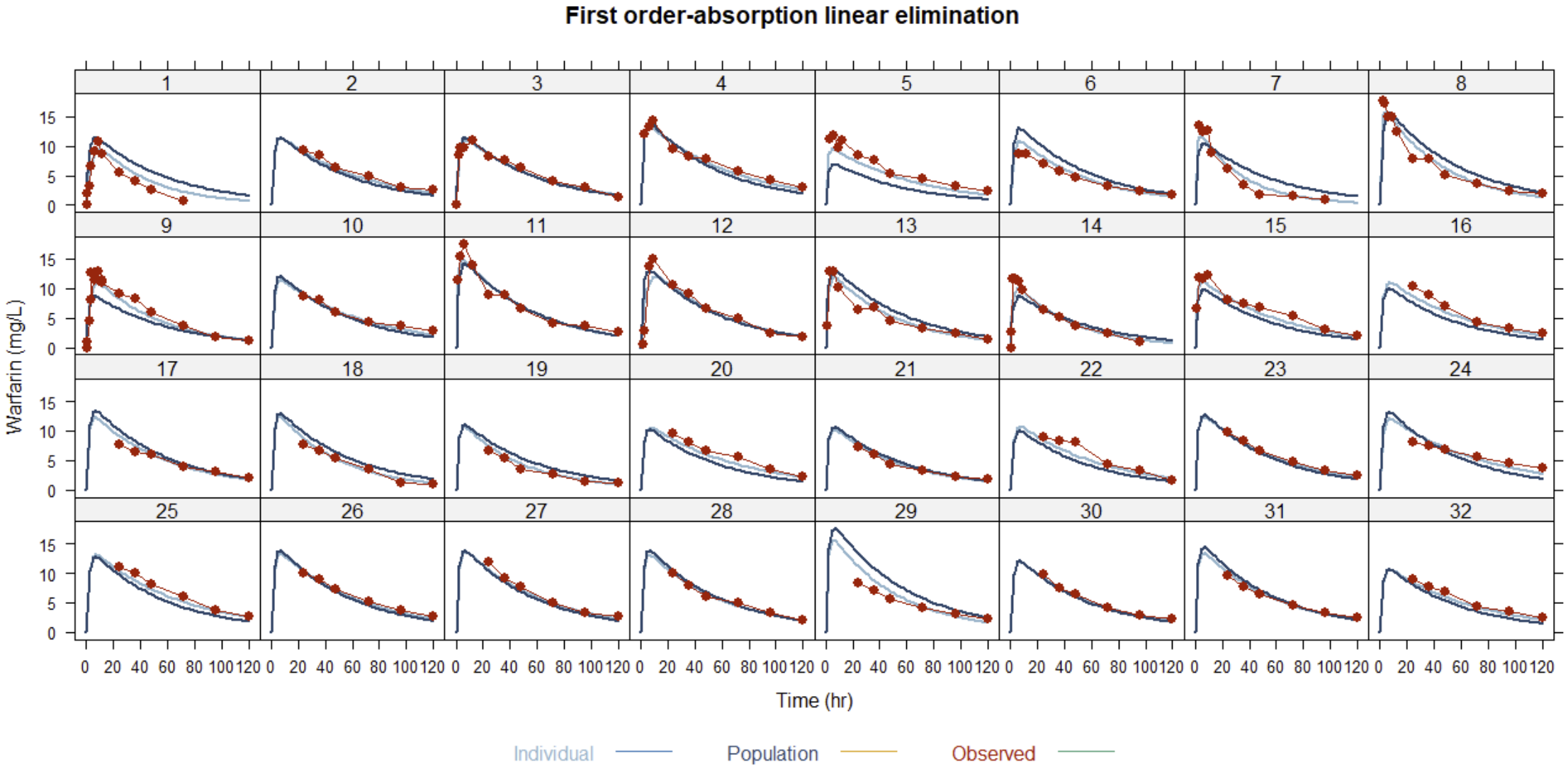
```
## Individual fits can be generated using augPred (augmented predictions)
## that provides smooth profiles by interpolating the predictions between observations:
plot(augPred(run001S))
## ...use the arrows in the plot window to examine the earlier curves
```



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(run001S)
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



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:

```
# rxode2-style differential equation definition
d/dt(depot)     = -ka * depot
d/dt(central) =  ka * depot - (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 run-time consuming

Running nlmixr^2 for a system of ODEs using FOCEi

```
One.comp.KA.ODE <- 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
    eta.cl ~ 0.1
    eta.v  ~ 0.1
  })
  model({
    # Where the model is specified
    cl <- exp(lcl + eta.cl)
    v  <- exp(lv + eta.v)
    ka <- exp(lka + eta.ka)
    # rxode2-style differential equation definition
    d/dt(gut)    = -ka * depot
    d/dt(center) =  ka * depot - (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)
  })
}

run002F <- nlmixr2(One.comp.KA.ODE, PKdata, est = "focei“, foceiControl(print = 5))
```

nlmixr² output for FOCEi with ODEs

```
> run002F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF     AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 425.4509 902.758 930.9616       -443.379        25.84916        2.290799

── Time (sec run002F$time): ──

        setup optimize covariance table compress  other
elapsed 0.001    1.835      1.835  0.06     0.02 15.959

── Population Parameters (run002F$parFixed or run002F$parFixedDf): ──

                           Parameter   Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lka                    log ka (1/h) -0.351 0.477  136      0.704 (0.276, 1.8)     100.      49.4%
lcl                    log Cl (L/h)     -2  0.12    6    0.136 (0.108, 0.172)     29.8      6.08%
lv                        log V (L)   2.07 0.115 5.55       7.95 (6.35, 9.97)     25.0      19.1%
prop.err proportional error (SD/mean)  0.125                           0.125
add.err         additive error (mg/L)  0.657                           0.657

  Covariance Type (run002F$covMethod): s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run002F$omega) or correlation (run002F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run002F$shrink
  Information about run found (run002F$runInfo):
   • gradient problems with initial estimate; see $scaleInfo
   • using S matrix to calculate covariance, can check sandwich or R matrix with $covRS and $covR
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run002F$censInformation): No censoring
  Minimization message (run002F$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:
    nlmixr2(...,control=list(outerOpt="bobyqa"))
```

Hands-on session II: running nlmixr^2 and diagnostics

- Examine the code in HandsOn_2.R to run a pre-programmed FOCEi analysis with a solved system and its diagnostics
- Examine the code in HandsOn_2.R to run a pre-programmed FOCEi analysis with ODEs

- Examine the VPCs and implement alternative models for absorption (like one or more transit compartments…)

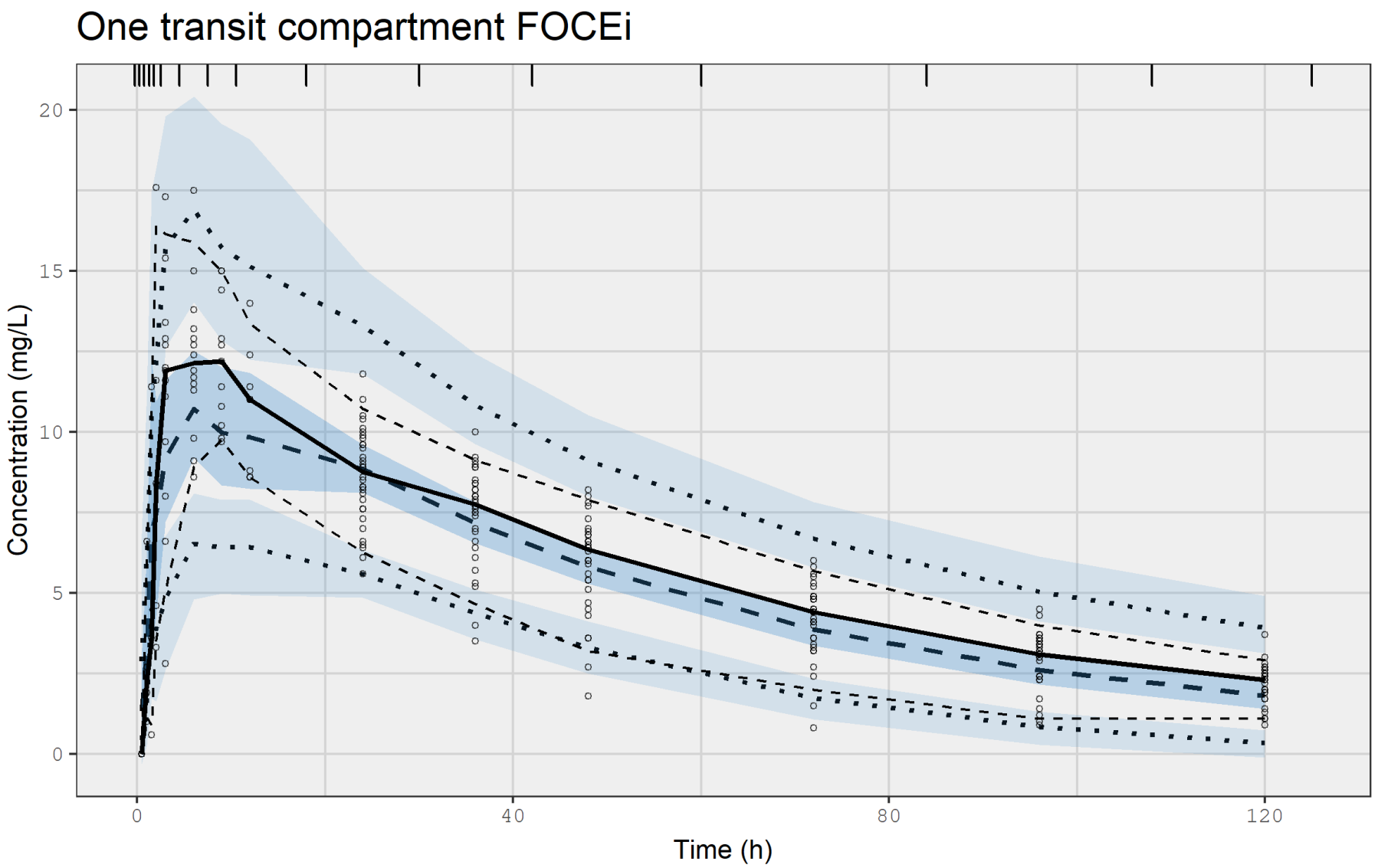
Running nlmixr²: 1 transit compartment

```
## 1 transit compartment
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
    eta.cl ~ 0.1
    eta.v  ~ 0.1
  })
  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)
  })}
```

nlmixr² output: 1 transit compartment with ODEs using FOCEi

```
> run003F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 321.1837 798.4908 826.6945      -391.2454        18.54498        1.596483

── Time (sec run003F$time): ──

        setup optimize covariance table compress  other
elapsed 0.017    2.688      2.688  0.06     0.03 12.867

── Population Parameters (run003F$parFixed or run003F$parFixedDf): ──

                           Parameter  Est.     SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr               log k transit (/h) 0.161  0.073  45.5       1.17 (1.02, 1.35)     64.0      45.9%
lcl                     log Cl (L/hr) -2.03 0.0226  1.11    0.132 (0.126, 0.138)     30.8      5.53%
lv                         log V (L)  2.07 0.0179 0.862       7.94 (7.66, 8.22)     26.4      17.6%
prop.err proportional error (SD/mean) 0.099                                0.099
add.err         additive error (mg/L) 0.489                                0.489

  Covariance Type (run003F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run003F$omega) or correlation (run003F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run003F$shrink
  Information about run found (run003F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run003F$censInformation): No censoring
```

VPC for one compartment model with a transit compartment using FOCEi



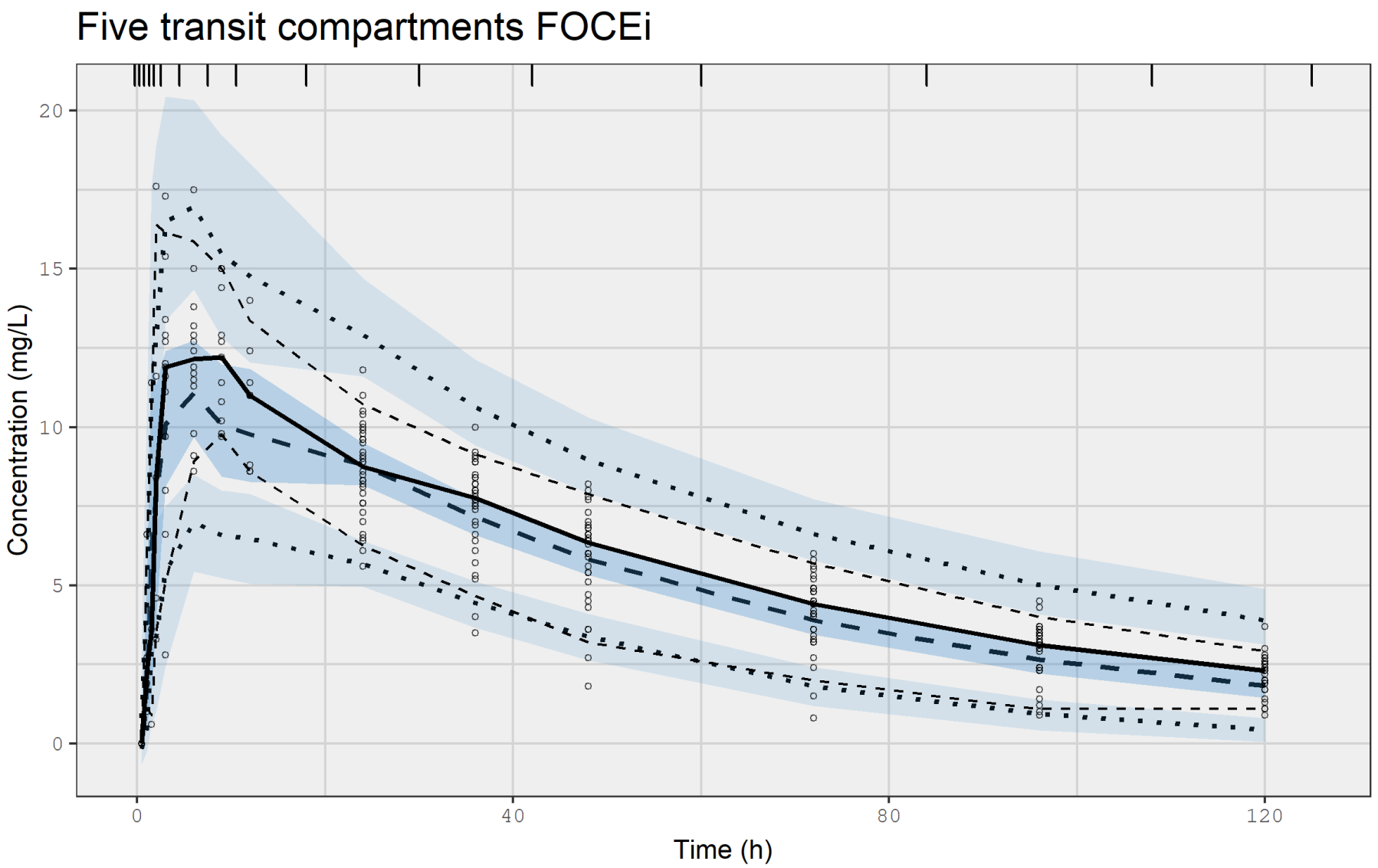
Running nlmixr²: 5 transit compartments

```
## 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)
  })}
```

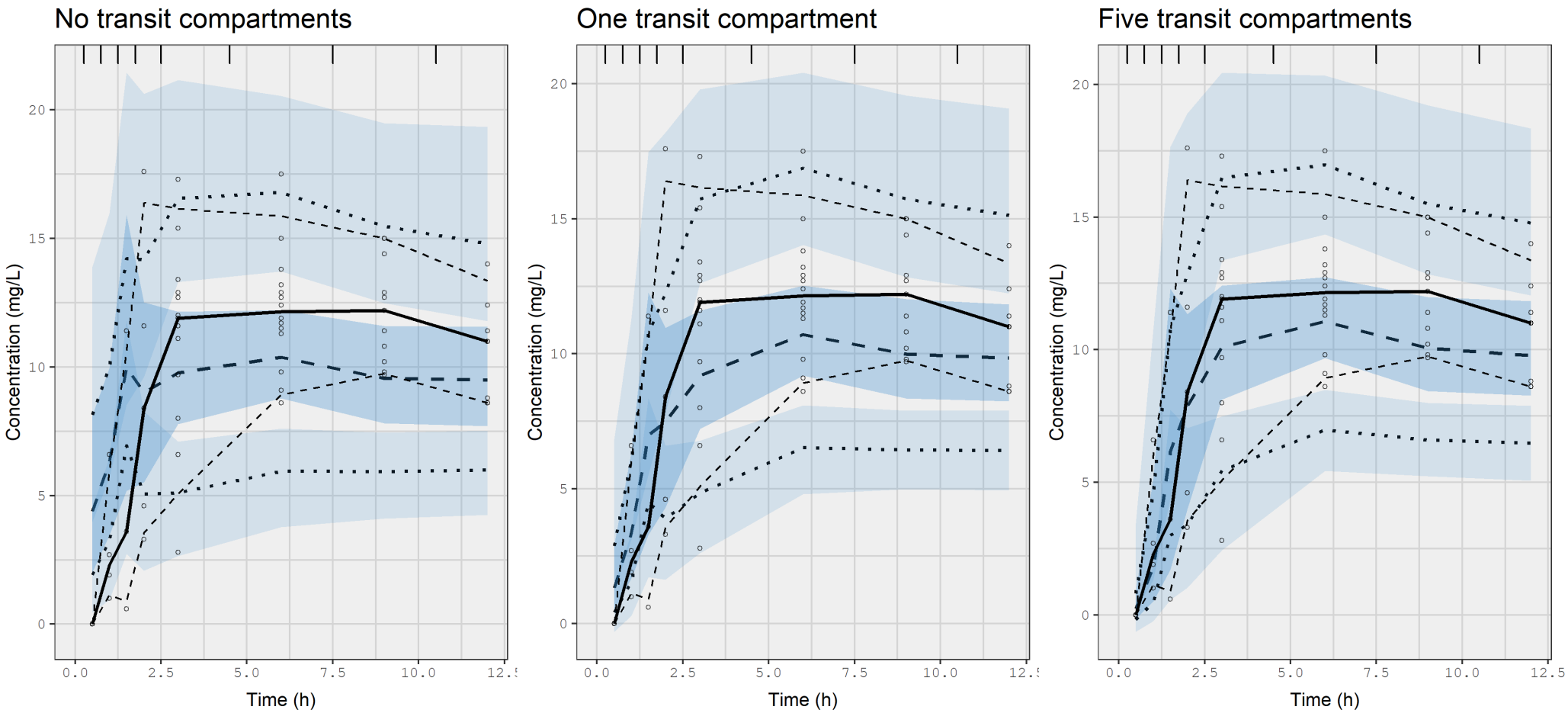
nlmixr2 output: 5 transit compartments with ODEs using FOCEi

```
> run004F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 230.5818 707.8889 736.0925      -345.9445        54.22588        3.848696

── Time (sec run004F$time): ──

        setup optimize covariance table compress  other
elapsed 0.001    4.453      4.453  0.05     0.03 16.243

── Population Parameters (run004F$parFixed or run004F$parFixedDf): ──

                             Parameter    Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr     log transit rate constant (/h)   1.37  0.139 10.2          3.94 (3, 5.17)     53.2      46.0%
lcl                        log Cl (L/h)  -2.04 0.0285  1.4     0.13 (0.123, 0.138)     30.2      1.69%
lv                           log V (L)   2.08 0.0302 1.45          8 (7.54, 8.49)     22.9      6.38%
prop.err   proportional error (SD/mean) 0.0759                              0.0759
add.err             additive error (mg)  0.361                               0.361

  Covariance Type (run004F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run004F$omega) or correlation (run004F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run004F$shrink
  Information about run found (run004F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run004F$censInformation): No censoring
```

VPC for first order absorption with 5 transit compartments using FOCEi



Comparing the VPCs for the first 12 hours to see if absorption is described better using zero, one, or five transit compartments

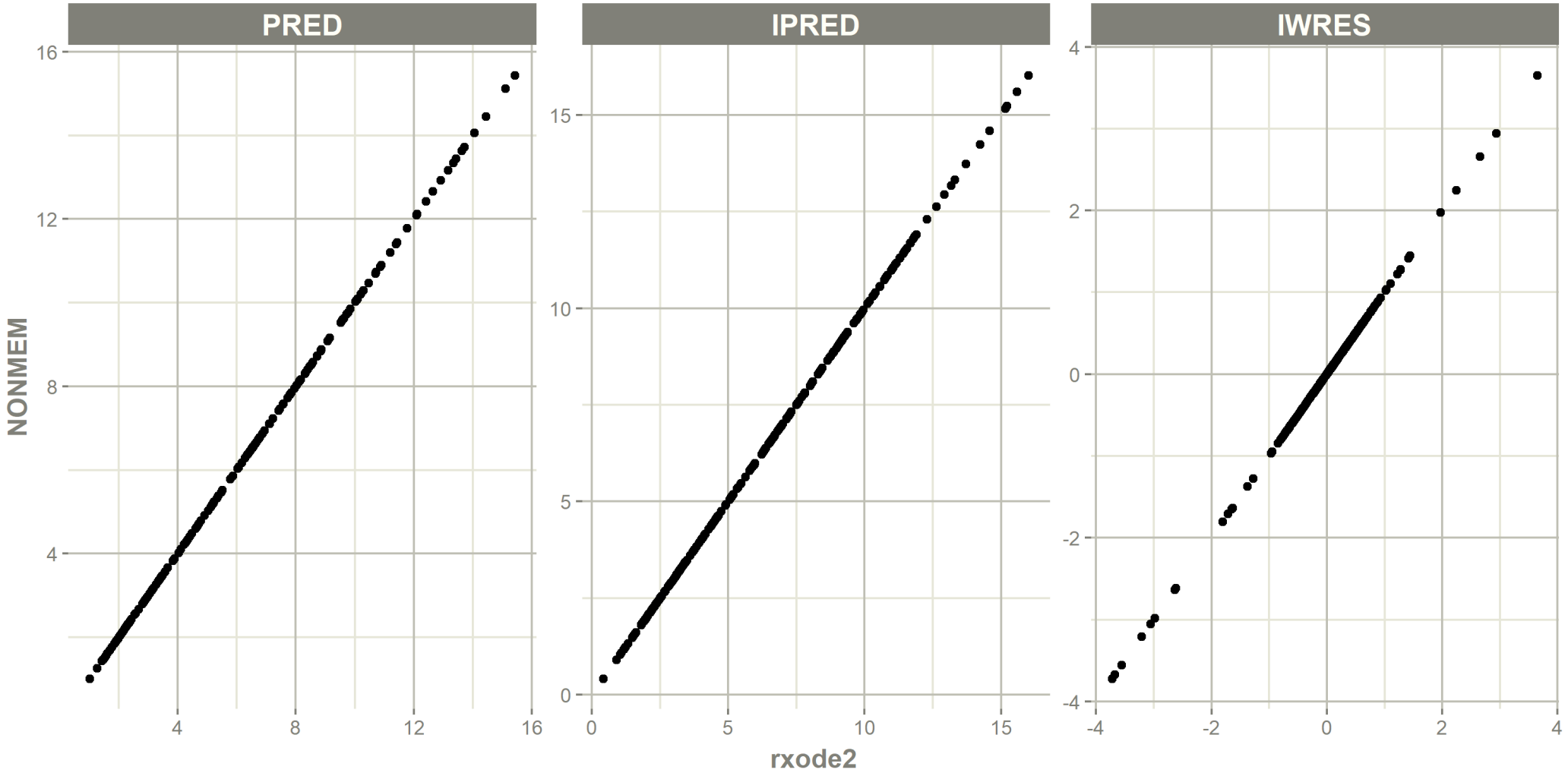


- For future reference, the code is provided in HandsOn_3.R

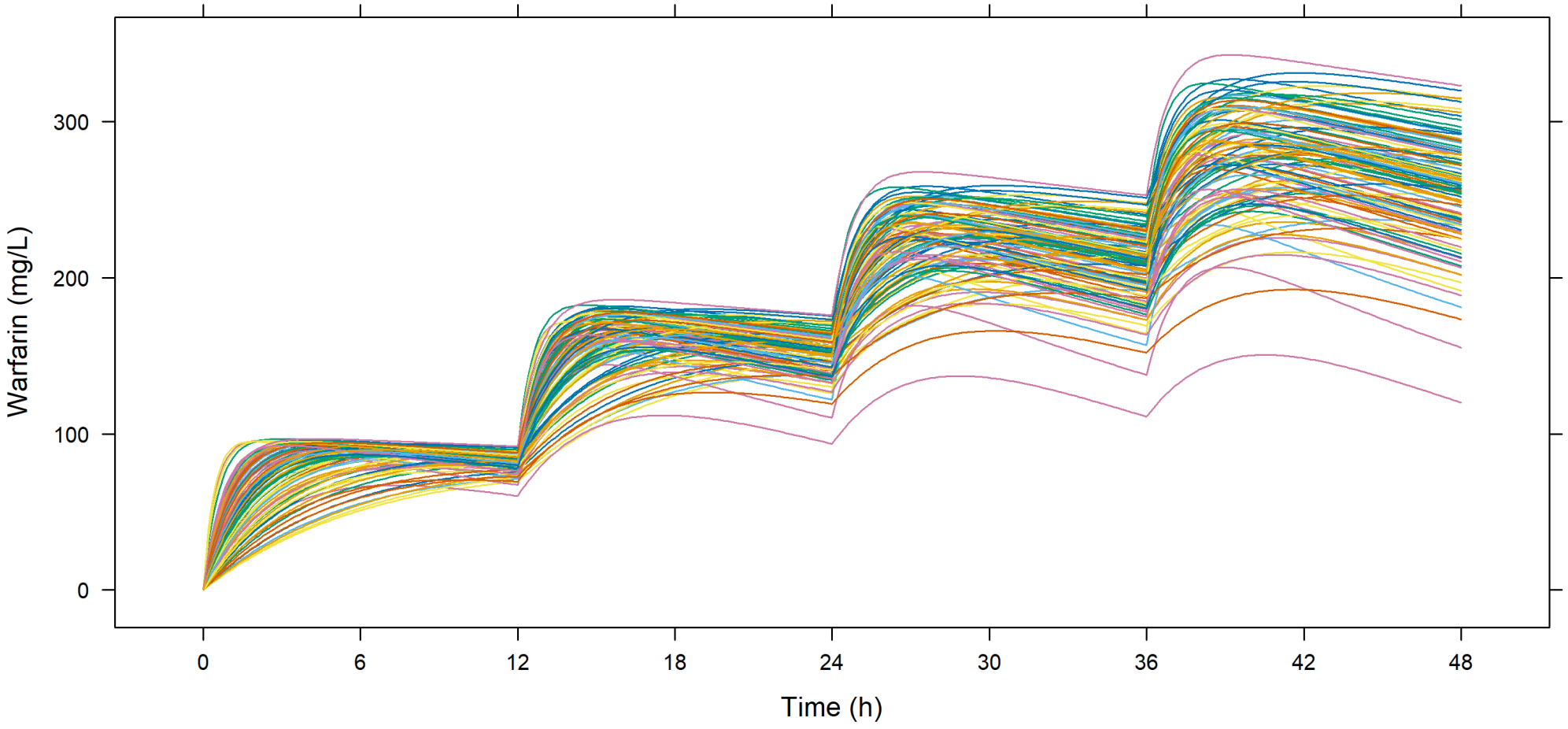
nlmixr^2 can read in NONMEM output and create nlmixr^2 models to simulate and re-estimate using nonmem2rx and babelmixr2

```
library(nlmixr2)
library(babelmixr2)
library(nonmem2rx)

## Read in NONMEM results -in this case from run005- into an rxode2 object
## Models read in correctly if Uppsala-style residual errors are used
## with SIGMA fixed to 1 and with only a single SIGMA
## Example with combined additive and proportional error:

mod5 <- nonmem2rx("run005.ctl",lst=".res", save=FALSE)

## to check if the model has been implemented correctly, NONMEM PRED, IPRED, and IWRES are
## compared with the rxode2 generated values after conversion

plot(mod5)
```

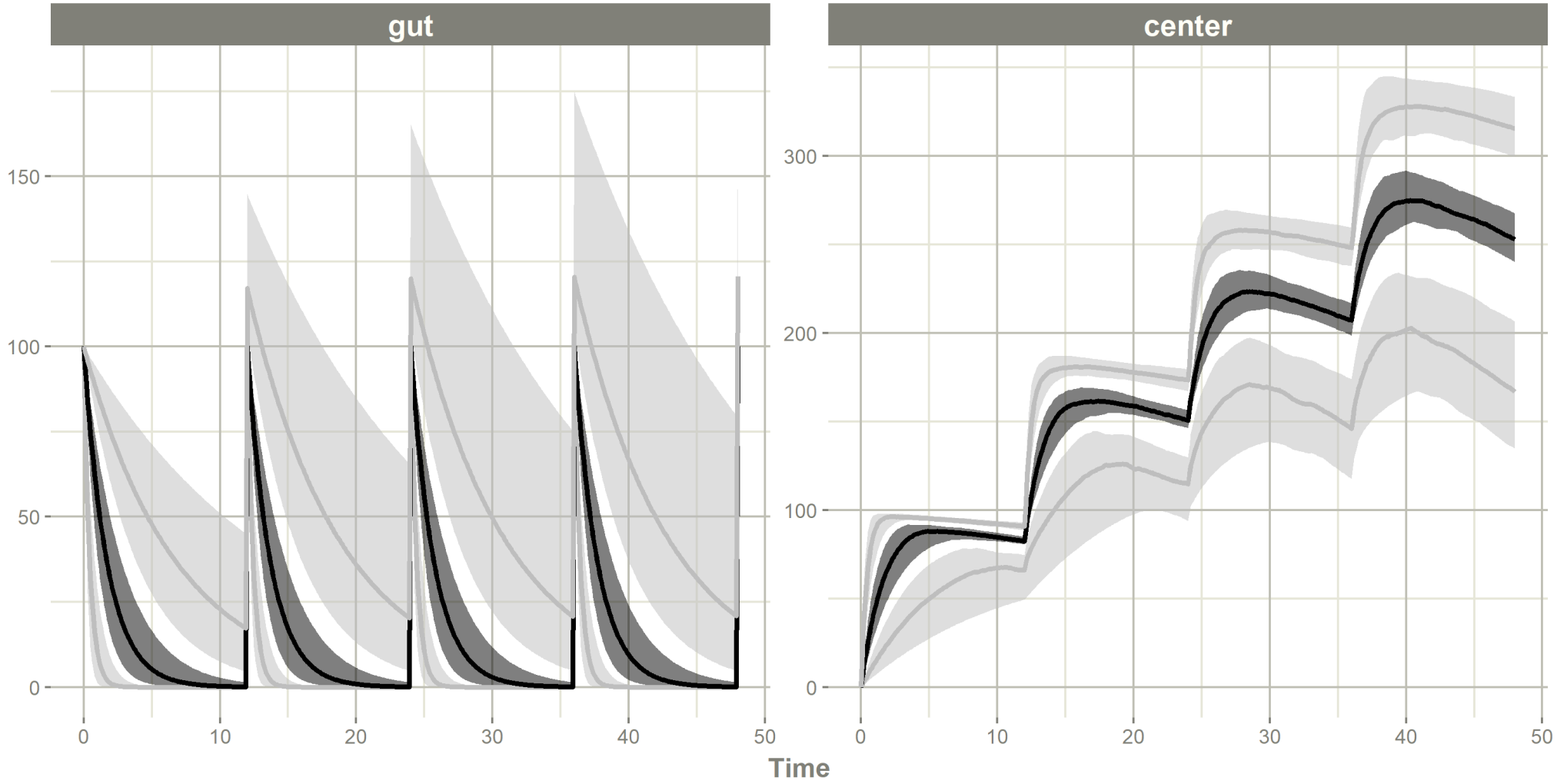
Check to see if the NONMEM-generated PRED, IPRED and IWRES values match the values when the model is used to simulate using rxode2



The rxode2 object can be used to simulate new regimens: e.g. four doses of 100 mg at 12 hours apart using 100 subjects

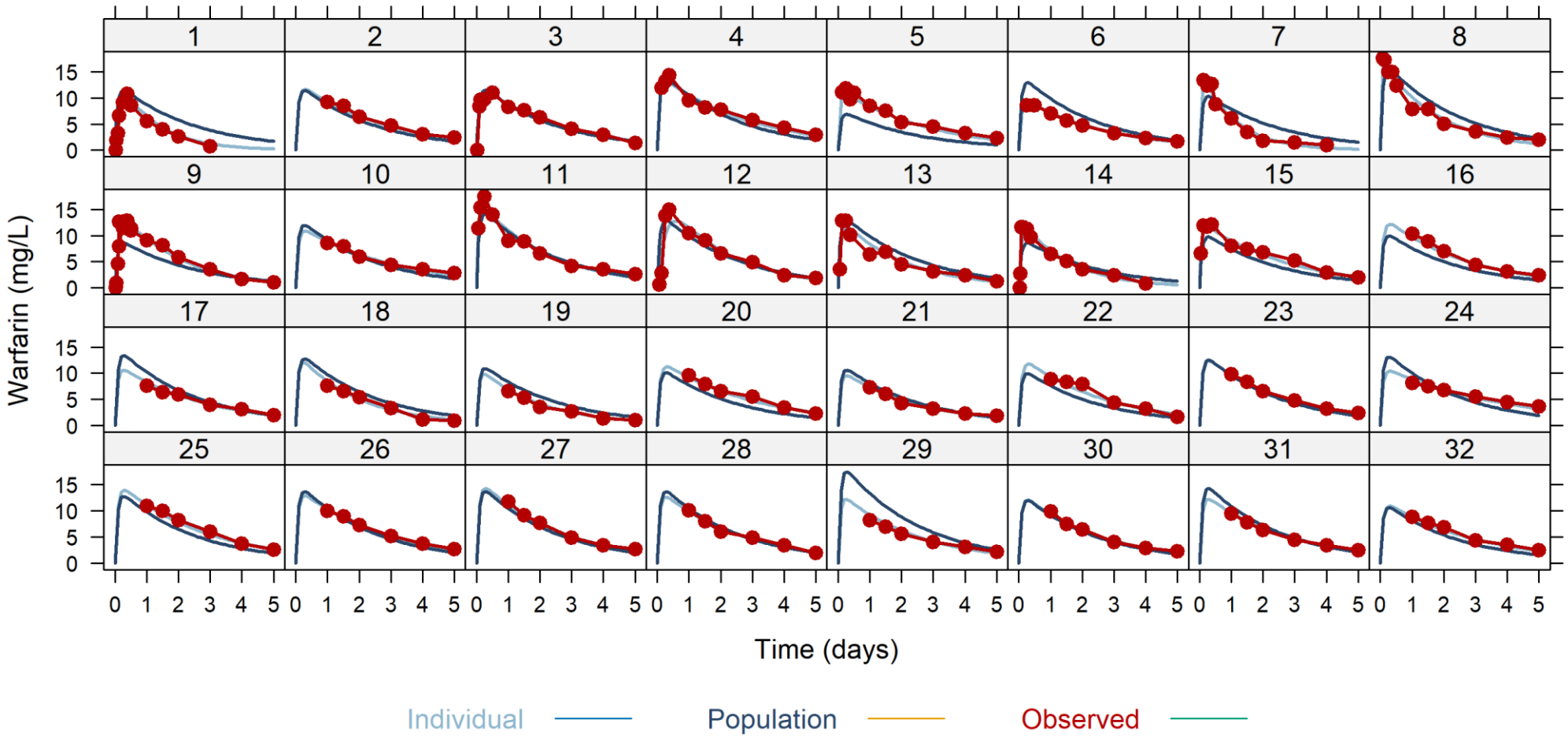


Or simulate 100 studies with these NONMEM parameters to capture uncertainty in theta estimates

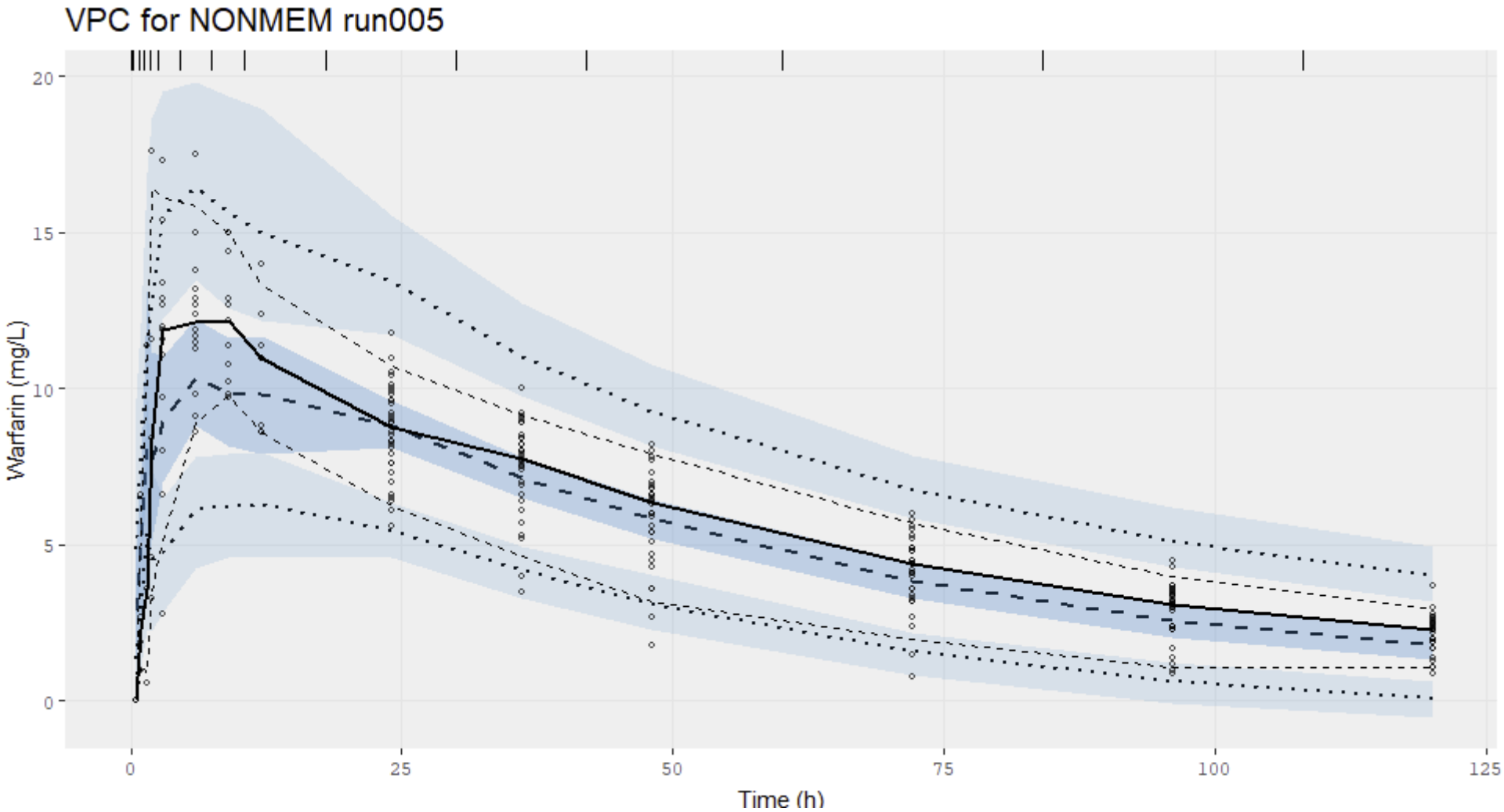


The rxode2 object can be translated to an nlmixr2 object to create individual plots

```
run005NM <- as.nlmixr2(mod5)
indivs5 <- augPred (run005NM)
```



The rxode2 object can be translated to an nlmixr2 object to create VPCs



The results of run005 as generated by NONMEM and read in using nonmem2rx

```
> #results for the NONMEM-translated model
> run005NM
── nlmixr² nonmem2rx reading NONMEM ver 7.5.0 ──

               OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
nonmem2rx -37.97987 439.3273 467.5309      -211.6636        143.2888        7.453375

── Time (sec run005NM$time): ──

        setup table compress NONMEM as.nlmixr2
elapsed  0.03  0.06     0.03   9.65       5.04

── Population Parameters (run005NM$parFixed or run005NM$parFixedDf): ──

          Parameter    Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
theta1       log ka -0.569  0.246 43.3 -0.569 (-1.05, -0.0866)
theta2       log cl  -2.01 0.0538 2.68     -2.01 (-2.11, -1.9)
theta3        log v   2.06  0.046 2.23       2.06 (1.97, 2.15)
EPS_prop   EPS_prop  0.117                              0.117
EPS_pkadd EPS_pkadd  0.722                              0.722
ETA_ka                                                    78.0      45.8%
ETA_cl                                                    28.7      3.74%
ETA_v                                                     22.2      13.4%
```

NONMEM-reported OFV:

```
********************************************************************************************************************************
**********                                                                                                    **********
**********                      FIRST ORDER CONDITIONAL ESTIMATION WITH INTERACTION                           **********
#OBJT:**********                        MINIMUM VALUE OF OBJECTIVE FUNCTION                                 **********
**********                                                                                                    **********
********************************************************************************************************************************

#OBJV:*********************************************      423.327       ************************************************************
```

The results of run005 as generated by nlmixr2 after rerunning the NONMEM model imported using nonmem2rx

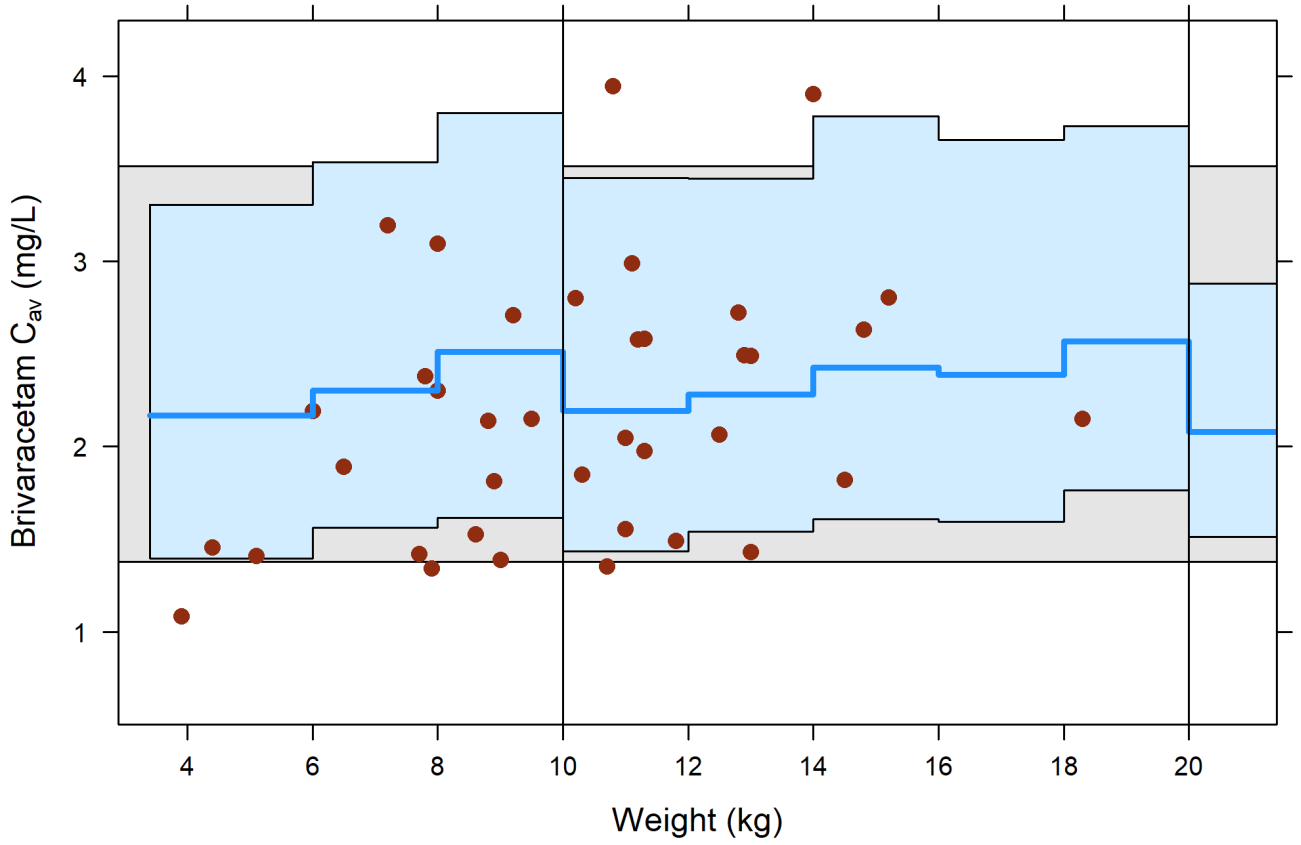
```
> #nlmixr2 results after running the NONMEM-translated model
> run005F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 423.3296 900.6368 928.8404      -442.3184        38.37877         2.75426

── Time (sec run005F$time): ──

        setup optimize covariance table compress other
elapsed 0.001    1.596      1.596  0.05     0.02 8.427

── Population Parameters (run005F$parFixed or run005F$parFixedDf): ──

          Parameter    Est.     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
theta1       log ka -0.569  0.344 60.4    -0.569 (-1.24, 0.105)
theta2       log cl  -2.01 0.0776 3.87     -2.01 (-2.16, -1.86)
theta3        log v   2.06 0.0669 3.25        2.06 (1.93, 2.19)
EPS_prop   EPS_prop  0.117                               0.117
EPS_pkadd EPS_pkadd  0.722                               0.722
ETA_ka                                                            78.0      45.8%
ETA_cl                                                            28.7      3.74%
ETA_v                                                             22.2      13.4%
```

Hands-on session IV: read in NONMEM results using nonmem2rx and run cool stuff

- Examine and run the code in HandsOn_4.R to explore the amazing functionality of nonmem2rx

For examination at home: simulations of paediatric posology

- The course material has a subdirectory NhanesPosologyGraphs that provides a full worked example of reading in a NONMEM model and using it to simulate pediatric exposures to create the following graph:

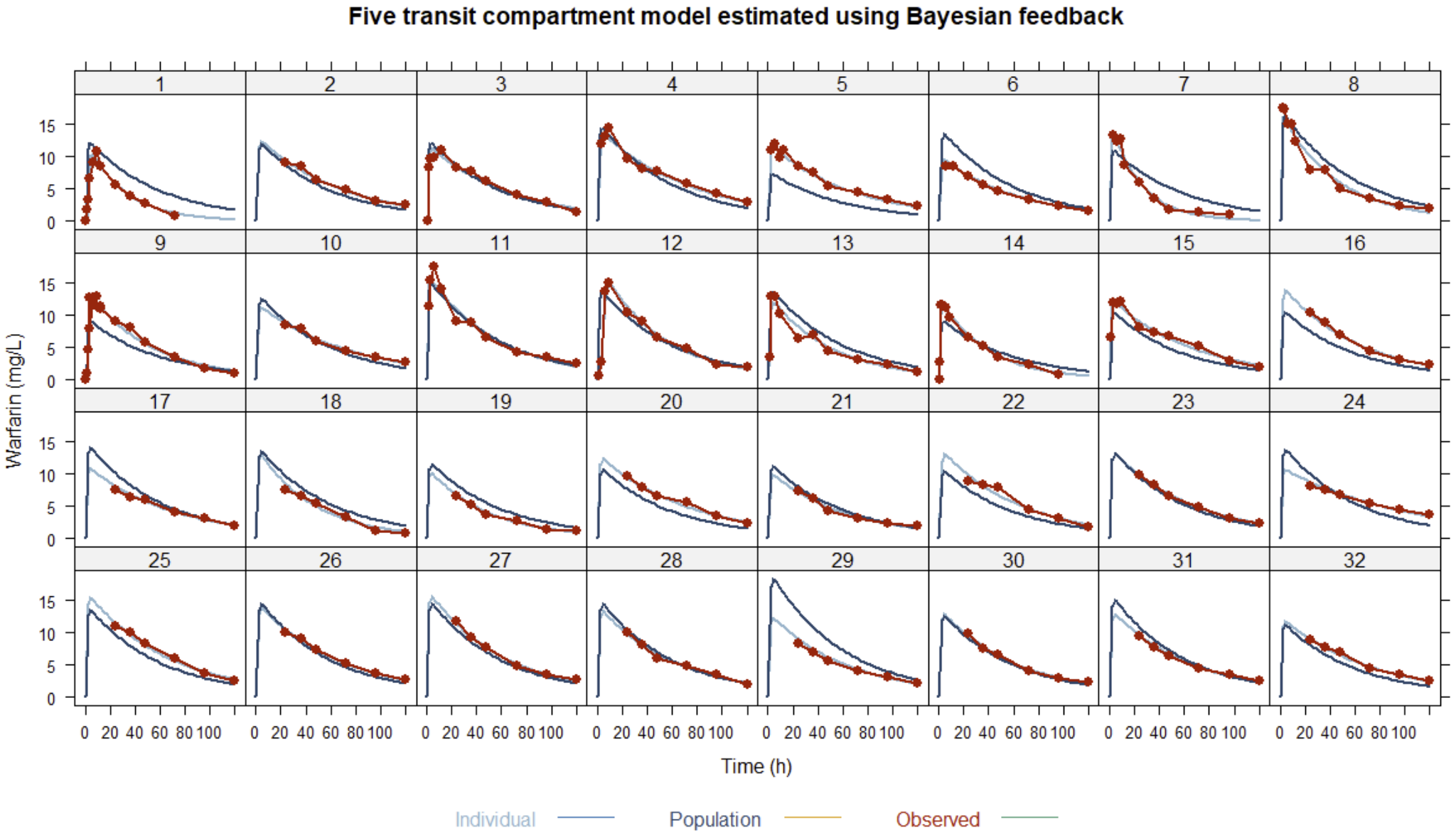


nlmixr^2 can generate empirical Bayes estimates for Bayesian feedback: individual EBEs for a new data set using existing population parameters

```
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
    eta.cl ~ 0.08073339
    eta.v ~ 0.04490733
  })
  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)
  })
}

run005ph <- nlmixr2(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 population parameter estimation



Parameterisation and mu-referencing

- For SAEM, parameters must be defined using ‘mu-referencing’
- This means that inter-individual variability (IIV) 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 nlmixr² code

```
## One compartment transit model with Sex on V
KAtr1_sexV <- function() {
  ini({
    lktr <- log(1.15)  #log k transit (/h)
    lcl  <- 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
    eta.cl ~ 0.1
    eta.v ~ 0.1
  })
  model({
    #Sex on volume
    cl <- exp(lcl + 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)
  })
```

nlmixr² output: mu-referenced sex on V (log-scale)

```
> run006F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 303.6673 782.9744 814.7035      -382.4872        49.19971         48.1342

── Time (sec run006F$time): ──

        setup optimize covariance table compress  other
elapsed 0.001    3.844      3.844  0.06     0.02 28.441

── Population Parameters (run006F$parFixed or run006F$parFixedDf): ──

                           Parameter  Est.    SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr                log k transit (/h) 0.202 0.324  160      1.22 (0.649, 2.31)     64.8      46.0%
lcl                       log CL (L/h) -2.02 0.117 5.82    0.133 (0.106, 0.167)     30.0      3.29%
lv                           log V (L)  1.73 0.229 13.2       5.65 (3.61, 8.86)     15.8      14.9%
Sex_V                     log Sex on V 0.404  0.24 59.3  0.404 (-0.0653, 0.874)
prop.err proportional error (SD/mean) 0.102                                0.102
add.err         additive error (mg/L) 0.451                                0.451

  Covariance Type (run006F$covMethod): s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run006F$omega) or correlation (run006F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run006F$shrink
  Information about run found (run006F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • using S matrix to calculate covariance, can check sandwich or R matrix with $covRS and $covR
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run006F$censInformation): No censoring
```

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 0.404 (95%CI: -0.0653/0.874) translates using

```
exp(c(0.404, -0.0653, 0.874))
```

to a fold-change estimate of

1.498 (95%CI: 0.937/2.397)

which corresponds to an increase of

49.9% (95%CI: -6.3%/140%)

for males compared to females

mu-referencing and allometric scaling

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

Corresponding nlmixr² code: allometric scaling

```
One.comp.transit.allo <- function() {
  ini({
    lktr <- log(1.15)  #log k transit (/h)
    lcl  <- log(0.15)  #log Cl (L/hr)
    lv   <- log(7)     #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
    eta.cl ~ 0.1
    eta.v ~ 0.1
  })
  model({
    #Allometric scaling on weight
    cl <- exp(lcl + eta.cl + ALLC * log(WT/70))
    v  <- exp(lv + eta.v + ALLV * log(WT/70))
    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)
  })
}
```

nlmixr² output: allometric scaling ODEs using FOCEi
Change in OFV compared to model without allometric scaling: -25.67

```
> run007F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 295.5111 772.8182 801.0219      -378.4091        6.813814        2.154404

── Time (sec run007F$time): ──

        setup optimize covariance table compress other
elapsed 0.013    7.981      7.981  0.05     0.02 7.475

── Population Parameters (run007F$parFixed or run007F$parFixedDf): ──

                           Parameter   Est.     SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr                log k transit (/h)  0.111 0.0115  10.3       1.12 (1.09, 1.14)     80.3       54.0%
lcl                      log Cl (L/hr)  -2.05 0.0271  1.32    0.129 (0.123, 0.136)     31.9       15.3%
lv                          log V (L)   2.08 0.0151 0.726        8.03 (7.8, 8.27)     18.0       28.5%
ALLC            allometric exponent cl   0.75  FIXED FIXED                    0.75
ALLV             allometric exponent v      1  FIXED FIXED                       1
prop.err proportional error (SD/mean) 0.0986                               0.0986
add.err          additive error (mg/L)  0.475                                0.475

  Covariance Type (run007F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run007F$omega) or correlation (run007F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run007F$shrink
  Information about run found (run007F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run007F$censInformation): No censoring
```

Hands-on session V: running an nlmixr^2 posthoc analysis and implementing covariates using mu-referencing

- Examine the code in HandsOn_5.R to run a posthoc analysis
- Examine the code in HandsOn _5.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

nlmixr² output: allometric scaling ODEs using FOCEi
Freely estimated exponents: drop of 5.30 points

```
> run007Free
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 290.2097 771.5168 806.7714      -375.7584        90.12464        3.870697

── Time (sec run007Free$time): ──

        optimize covariance table compress other
elapsed    7.835      7.835  0.03     0.01  7.43

── Population Parameters (run007Free$parFixed or run007Free$parFixedDf): ──

                           Parameter  Est.    SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
lktr                log k transit (/h)   0.3 0.312  104      1.35 (0.733, 2.49)     63.2      42.4%
lcl                      log Cl (L/hr) -2.01 0.103 5.15     0.134 (0.11, 0.164)     26.6      1.63%
lv                          log V (L)  2.08 0.071 3.42       7.99 (6.95, 9.18)     13.8      20.3%
ALLC           allometric exponent cl 0.696 0.562 80.7     0.696 (-0.405, 1.8)
ALLV            allometric exponent v 0.944 0.432 45.7    0.944 (0.0985, 1.79)
prop.err proportional error (SD/mean)   0.1                                0.1
add.err          additive error (mg/L) 0.479                              0.479

  Covariance Type (run007Free$covMethod): s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run007Free$omega) or correlation (run007Free$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run007Free$shrink
  Information about run found (run007Free$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • using S matrix to calculate covariance, can check sandwich or R matrix with $covRS and $covR
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run007Free$censInformation): No censoring
```

PKPD analysis with nlmixr²: sequential estimation

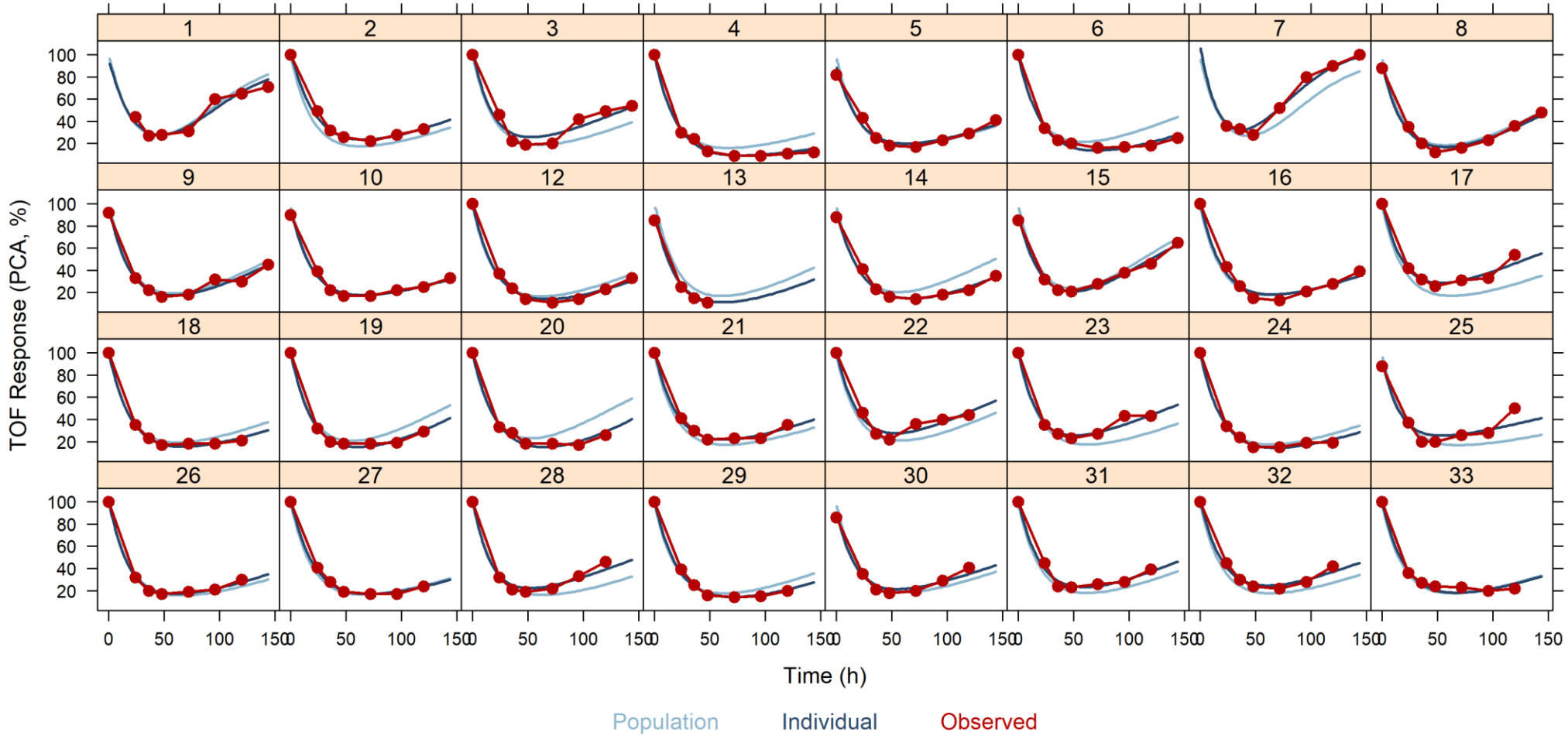
- First approach: use EBEs from a previous PK model to define PK profiles and estimate PKPD relationship
- Extract EBEs from nlmixr2 object and merge to PKPD data file

```
load(file = “run007F.Rdata")
#Use the one compartment transit model to start the PD analysis
EBEs <- as.data.table(run007F)
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)

# remove PK measurements from file so as to not confuse the vpcPlot later on
PDdata[, DEL := ifelse(DVID == 1 & AMT == 0, 1, 0)]
Pddata <- Pddata[DEL == 0]
```

PKPD analysis with nlmixr²: 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)
  })
}
```

Individual graphs for turnover Emax PD model with PK using EBEs



PKPD analysis with nlmixr^2: simultaneous estimation

- Second approach: estimate PK and PD simultaneously
- The source of observations is identified using a `dvid` data item
- `dvid` is coded as `central` and `effect` in the data object to identify the two types of observation as presented in the model code
- `dvid` can also coded as `1` and `2` in the data object to identify the two types of observation as presented in the model code

Immediate effect simultaneous PKPD analysis with nlmixr^2: `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^2: `model` block

```
#Immediate effect
  model({
    ktr <- exp(tktr + eta.ktr)
    cl  <- exp(tcl + 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 nlmixr²: examine defined model

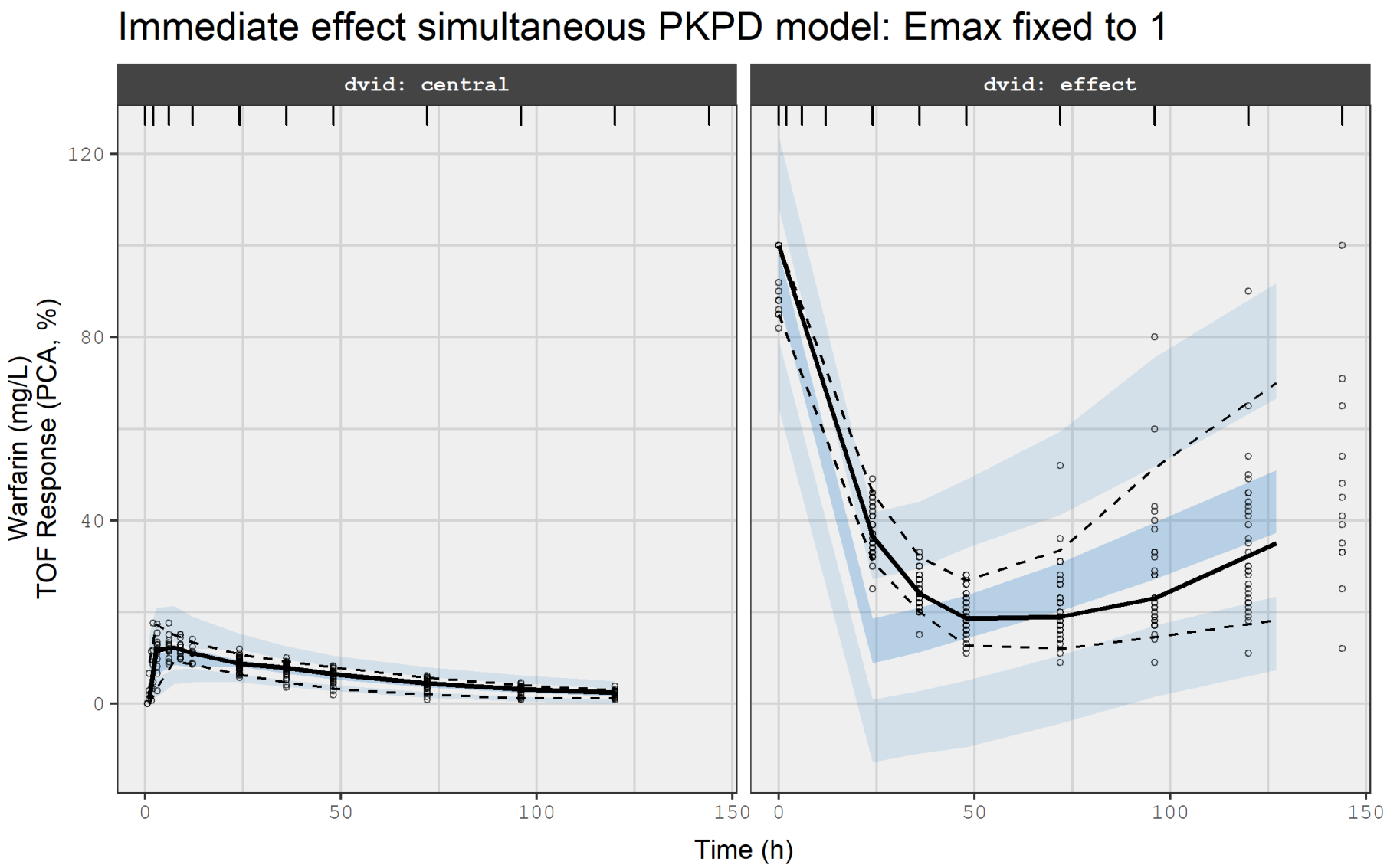
```
nlmixr2(KA1tr1_PDimmemax1) # Show initial estimates and model information:
```

```
> nlmixr2(KA1tr1_PDimmemax1) # Show initial estimates and model
i parameter labels from comments will be replaced by 'label()'
 -- rxode2-based free-form 3-cmt ODE model ------------------------------------------------------------
 -- Initalization: --
Fixed Effects ($theta):
      tktr        tcl         tv eps.pkprop  eps.pkadd       tc50        te0  eps.pdadd
 0.2623643 -1.8971200  2.0794415  0.1000000  0.4000000  0.4054651  4.6051702 20.0000000

Omega ($omega):
        eta.ktr eta.cl eta.v eta.c50 eta.e0
eta.ktr       1    0.0   0.0     0.0    0.0
eta.cl        0    0.3   0.0     0.0    0.0
eta.v         0    0.0   0.3     0.0    0.0
eta.c50       0    0.0   0.0     0.5    0.0
eta.e0        0    0.0   0.0     0.0    0.1

States ($state or $stateDf):
  Compartment Number Compartment Name
1                  1            depot
2                  2          central
3                  3            trans
 -- Multiple Endpoint Model ($multipleEndpoint): --
    variable                    cmt                    dvid*
1     cp ~ … cmt='central' or cmt=2 dvid='central' or dvid=1
2 effect ~ …  cmt='effect' or cmt=4  dvid='effect' or dvid=2
  * If dvids are outside this range, all dvids are re-numered sequentially, ie 1,7, 10 becomes 1,2,3 etc

 -- μ-referencing ($muRefTable): --
  theta     eta level
1  tktr eta.ktr    id
2   tcl  eta.cl    id
3    tv   eta.v    id
4  tc50 eta.c50    id
5   te0  eta.e0    id
```

nlmixr² output: immediate effect simultaneous PKPD model

```
> run009F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 1762.228 2675.923 2730.263      -1324.961        195.1339        11.52368

── Time (sec run009F$time): ──

        setup optimize covariance table compress  other
elapsed 0.001   11.671     11.671  0.06     0.03 24.507

── Population Parameters (run009F$parFixed or run009F$parFixedDf): ──

                Parameter  Est.      SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr          log ktr (/h) 0.374  0.167  44.7       1.45 (1.05, 2.02)     69.6      44.7%
tcl           log CL (L/h) -2.03 0.0436  2.15    0.132 (0.121, 0.144)     29.1      3.82%
tv              log Vc (L)   2.1 0.0529  2.52       8.16 (7.35, 9.05)     24.1      9.32%
eps.pkprop                  0.104                                0.104
eps.pkadd                     0.5                                  0.5
tc50       log ec50 (mg/L) 0.382 0.0742  19.4       1.46 (1.27, 1.69)     28.0      40.3%
te0              log e0  4.52 0.0214 0.474         92.1 (88.3, 96)     14.3      60.9%
eps.pdadd                   12.3                                 12.3

  Covariance Type (run009F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run009F$omega) or correlation (run009F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run009F$shrink
  Information about run found (run009F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run009F$censInformation): No censoring
```

VPC for immediate effect simultaneous PKPD model



Effect compartment simultaneous PKPD analysis with nlmixr2: ini block

```
#Effect compartment model
KA1tr1_PDceemax <- 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)
    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({

})
}
```

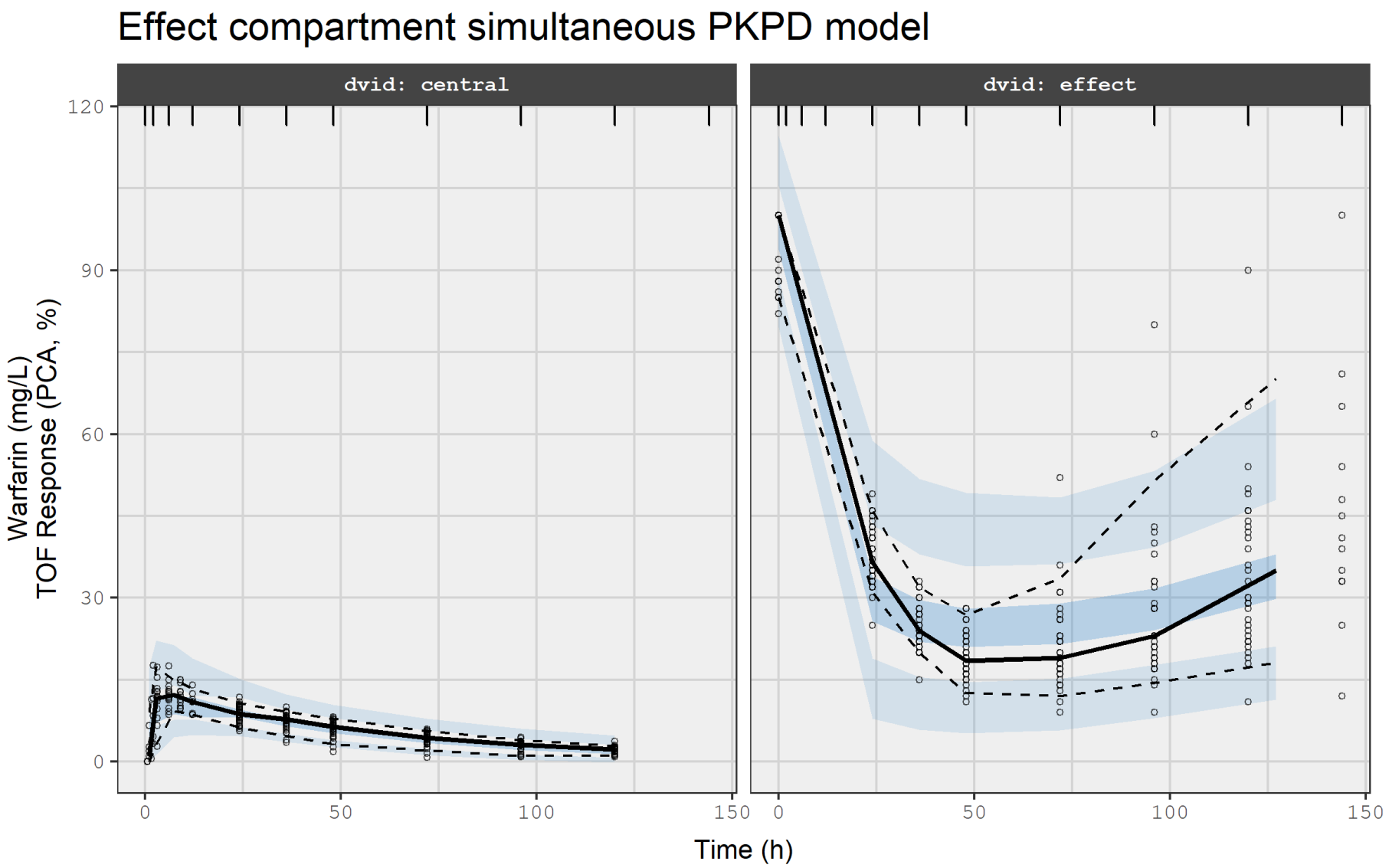
Effect compartment simultaneous PKPD analysis with nlmixr²: `model` block

```
#Effect compartment model
 model({
    ktr <- exp(tktr + eta.ktr)
    cl  <- exp(tcl + 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
  })
}
```

nlmixr² output: effect compartment simultaneous PKPD model

```
> run010F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF  AIC    BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 1523.306 2441 2503.7        -1205.5         108.638        32.47682

── Time (sec run010F$time): ──

        setup optimize covariance table compress   other
elapsed 0.001   23.595     23.595  0.06     0.02 234.389

── Population Parameters (run010F$parFixed or run010F$parFixedDf): ──

                Parameter  Est.     SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr          log ktr (/h) 0.277 0.0887  32.1       1.32 (1.11, 1.57)     57.9      40.0%
tcl           log CL (L/h) -2.04 0.0542  2.66     0.13 (0.117, 0.144)     29.4      1.99%
tv              log Vc (L)  2.06 0.0797  3.86       7.86 (6.72, 9.19)     23.3      9.85%
eps.pkprop                 0.104                                 0.104
eps.pkadd                  0.495                                 0.495
tc50       log ec50 (mg/L) 0.531 0.0795    15        1.7 (1.46, 1.99)     31.8      18.0%
tkout       log tkout (/h) -3.94  0.151  3.82 0.0195 (0.0145, 0.0262)     34.5      34.3%
te0               log e0  4.57 0.0273 0.597        96.8 (91.7, 102)     8.68      47.7%
eps.pdadd                  6.44                                  6.44

  Covariance Type (run010F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run010F$omega) or correlation (run010F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run010F$shrink
  Information about run found (run010F$runInfo):
    • gradient problems with initial estimate and covariance; see $scaleInfo
    • last objective function was not at minimum, possible problems in optimization
    • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
    • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run010F$censInformation): No censoring
```

VPC for effect compartment simultaneous PKPD model



Turnover model simultaneous PKPD analysis with nlmixr²: `ini` block

```
#Turnover simultaneous PKPD model
KA1tr1IPP_PDtoemax <- 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)
    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({

})
}
```

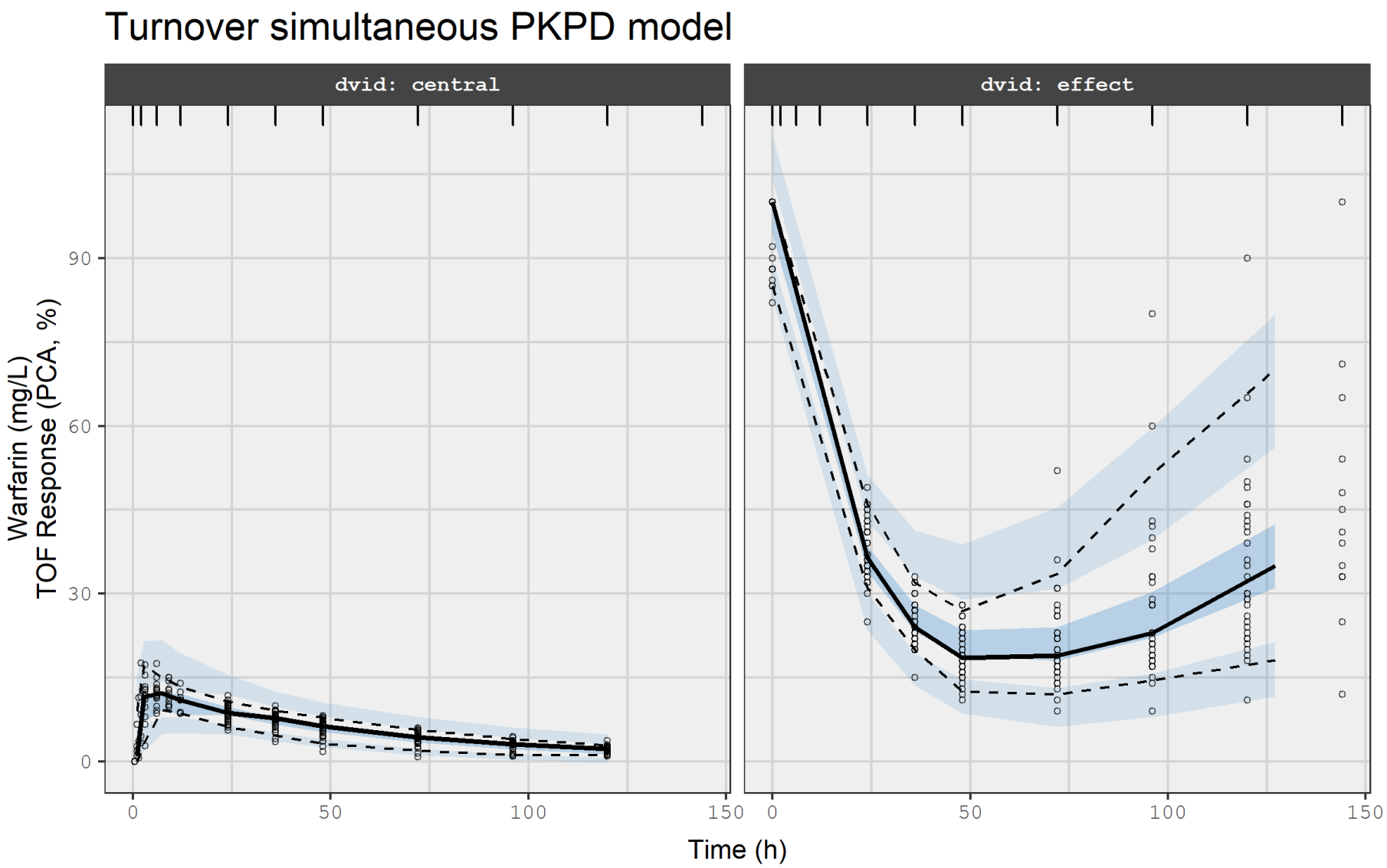
Turnover model simultaneous PKPD analysis with nlmixr²: `model` block

```
#Turnover simultaneous PKPD model
 model({
ktr <- exp(tktr + eta.ktr)
    cl  <- exp(tcl + 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
  })
}
```

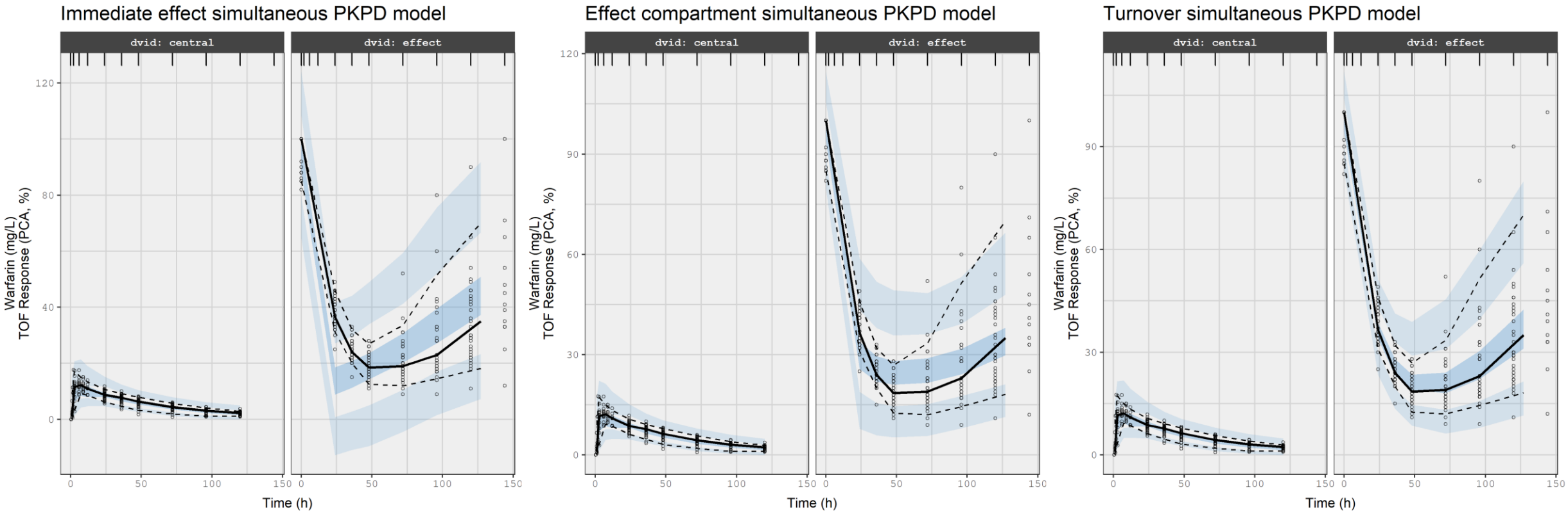
nlmixr² output: turnover simultaneous PKPD model

```
> run011F
── nlmixr² FOCEi (outer: nlminb) ──

          OBJF      AIC      BIC Log-likelihood Condition#(Cov) Condition#(Cor)
FOCEi 1330.127 2247.821 2310.521      -1108.911        338.3274        3.064891

── Time (sec run011F$time): ──

        setup optimize covariance table compress   other
elapsed 0.001   28.334     28.334  0.06     0.03 352.441

── Population Parameters (run011F$parFixed or run011F$parFixedDf): ──

                Parameter  Est.     SE  %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
tktr          log ktr (/h) 0.154  0.181   118      1.17 (0.818, 1.66)     66.7       48.0%
tcl           log CL (L/h)    -2 0.0511  2.55    0.135 (0.122, 0.149)     32.3       10.7%
tv              log Vc (L)  2.06 0.0238  1.16       7.81 (7.45, 8.18)     27.8       19.4%
eps.pkprop                 0.104                                0.104
eps.pkadd                  0.458                                0.458
tc50       log ec50 (mg/L) 0.136 0.0727  53.3      1.15 (0.994, 1.32)     49.0       11.0%
tkout       log tkout (/h) -2.96 0.0208 0.703  0.052 (0.0499, 0.0541)     8.89       32.5%
te0               log e0  4.57 0.0104 0.227       96.6 (94.6, 98.5)     5.28       18.0%
eps.pdadd                  3.72                                3.72

  Covariance Type (run011F$covMethod): r,s
  No correlations in between subject variability (BSV) matrix
  Full BSV covariance (run011F$omega) or correlation (run011F$omegaR; diagonals=SDs)
  Distribution stats (mean/skewness/kurtosis/p-value) available in run011F$shrink
  Information about run found (run011F$runInfo):
   • gradient problems with initial estimate and covariance; see $scaleInfo
   • last objective function was not at minimum, possible problems in optimization
   • ETAs were reset to zero during optimization; (Can control by foceiControl(resetEtaP=.))
   • initial ETAs were nudged; (can control by foceiControl(etaNudge=., etaNudge2=))
  Censoring (run011F$censInformation): No censoring
```

VPC for turnover simultaneous PKPD model



VPCs to compare the immediate effect model, the effect compartment model, and the turnover simultaneous PKPD model



Hands-on session VI: running nlmixr2 PKPD analysis

- Examine the code in HandsOn_6.R to run the sequential or one of the simultaneous PKPD analyses