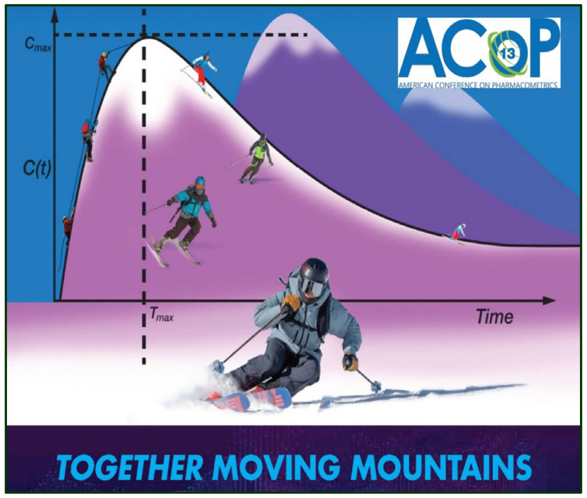


C_{max}
C(t)
T_{max}
Time
ACoP 13
AMERICAN CONFERENCE ON PHARMACOMETRICS
TOGETHER MOVING MOUNTAINS

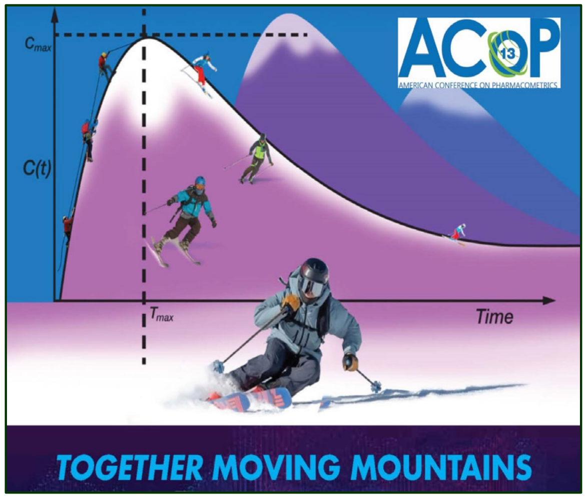
Session 1a: nlmixr2 Tutorial Session

Chairs: Matthew Fidler, William S. Denney

Matt Fidler	nlmixr2 overview
William Denney	Estimating initial PK model parameters for nlmix2 with PKNCA
	Q&A / Hands On

Aurora, CO

October 30 - November 2, 2022


ACoP
13
AMERICAN CONFERENCE ON PHARMACOMETRICS
C_{max}
C(t)
T_{max}
Time
TOGETHER MOVING MOUNTAINS

Session 1a: nlmixr2 Overview

nlmixr2: an open-source package for pharmacometric modeling in R

Matt Fidler

Aurora, CO

October 30 - November 2, 2022

nlmixr^2: an open-source package for pharmacometric modeling in R

ACoP Conference 2022 tutorial

Matthew Fidler

On behalf of the nlmixr2 development team:

Matt Fidler, Bill Denney, Richard Hooijmaijers, Rik Schoemaker, Mirjam Trame, Theodoros Papathanasiou, Justin Wilkins, Yuan Xiong, John Harrold, Huijuan Xu

Current nlmixr^2 team

nlmixr Team Lead

Matthew Fidler, PhD

Bill Denney, PhD

John Harrold, PhD

Richard Hooijmaijers, BSc

Theodoros Papathanasiou, PhD

Rik Schoemaker, PhD

Mirjam Trame, PhD

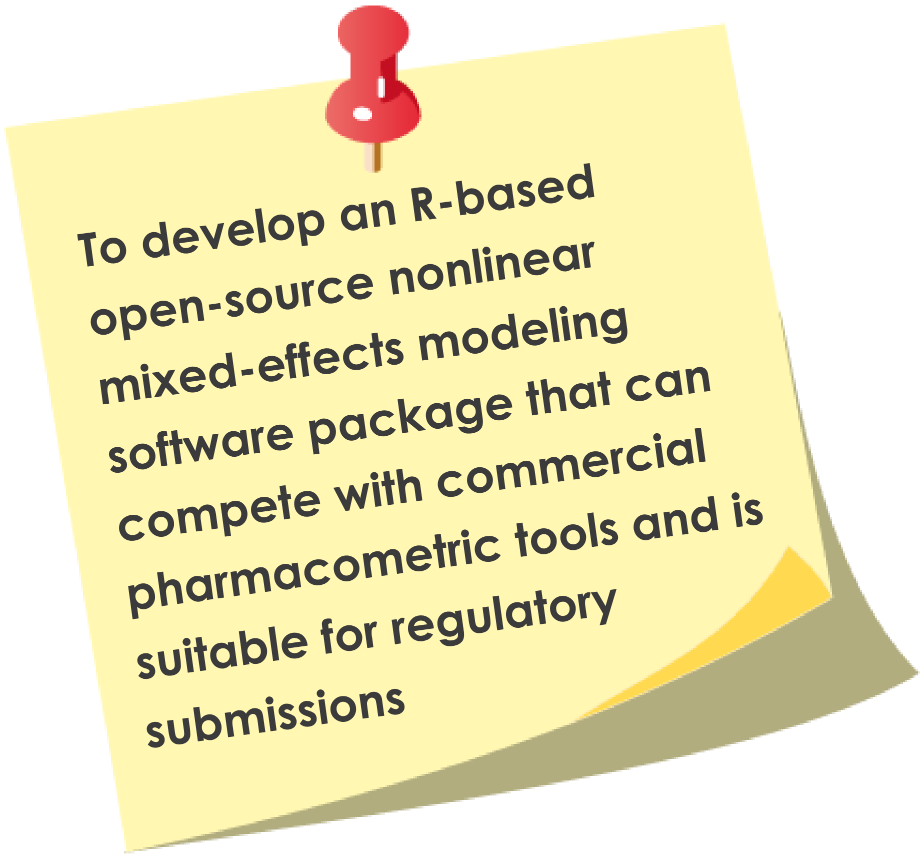
Justin Wilkins, PhD

Yuan Xiong , PhD

Huijuan Xu, PhD

nlmixr^2

Vision of nlmixr²



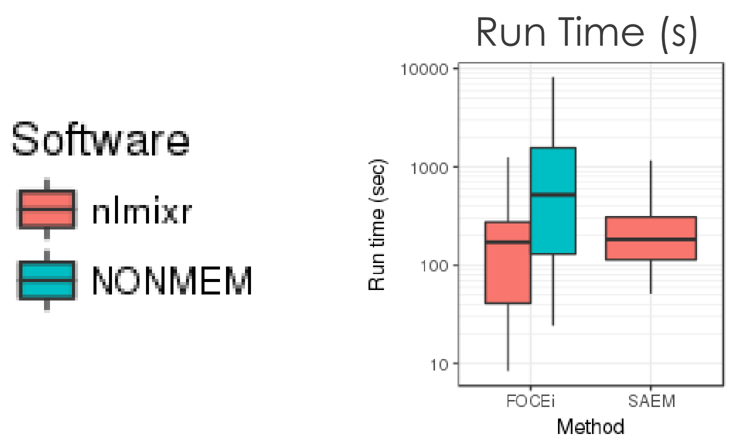
To develop an R-based open-source nonlinear mixed-effects modeling software package that can compete with commercial pharmacometric tools and is suitable for regulatory submissions

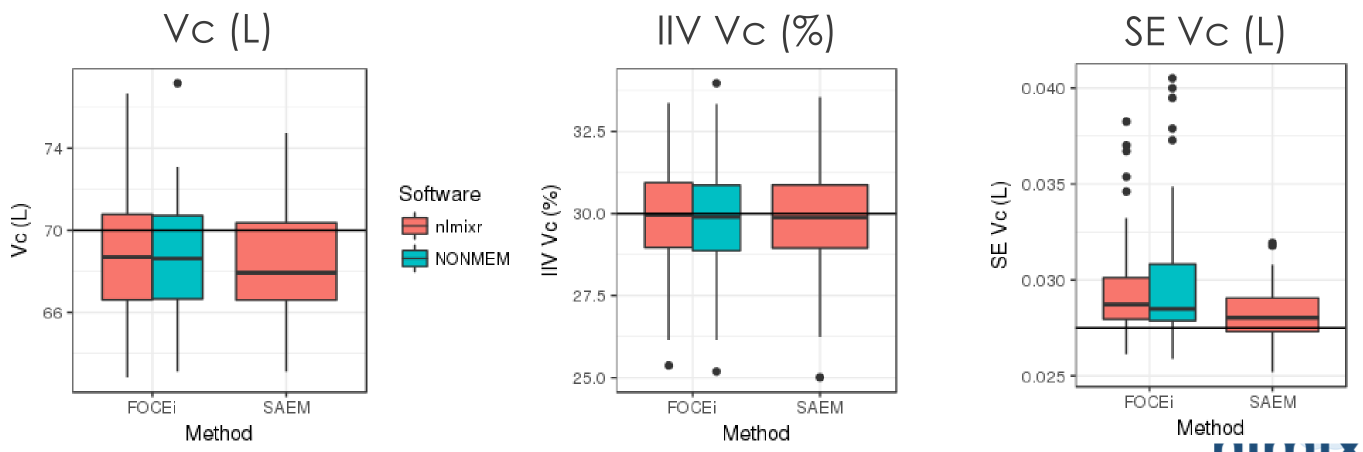
New Features in nlmixr^2/rxode^2

- Drop the requirement for saem to use mu-referencing
- Simpler, more consistent simulations with nlmixr2/rxode2
- Consistent interface between all the estimation control objects
- Simplified code to allow more extensions in the future (starting an API)
- Generalized likelihood
- Interaction with other tools like PKNCA, NONMEM and Monolix (babelmixr2)

nlmixr2 is a nonlinear mixed effects modeling R package with comparable performance to commercial software

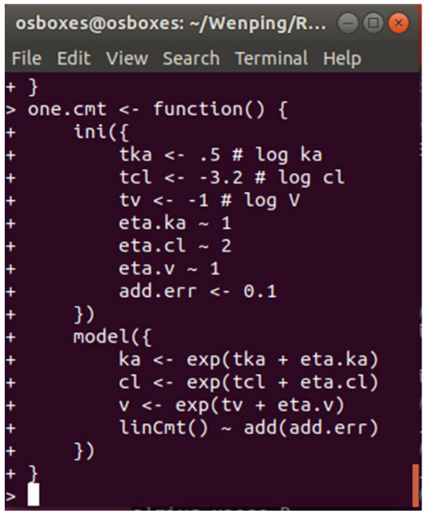
- Run time for ODE model:
  - FOCEi: nlmixr runs faster than NONMEM
  - SAEM: nlmixr runs as fast as Monolix and both are faster than NONMEM
- Parameter Estimates were similar for all three NLME tools.
- One known submission/approval to FDA with nlmixr



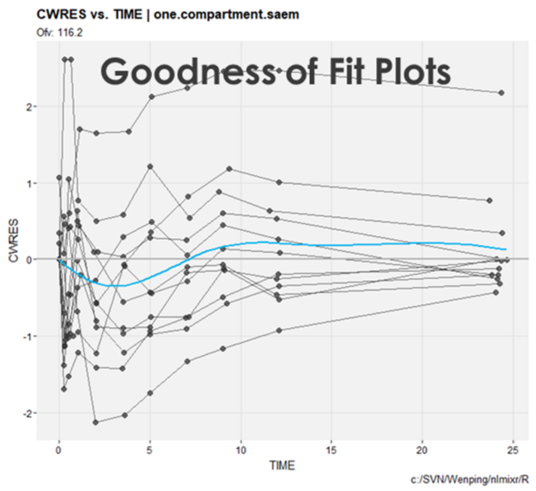


Outline

Model Syntax

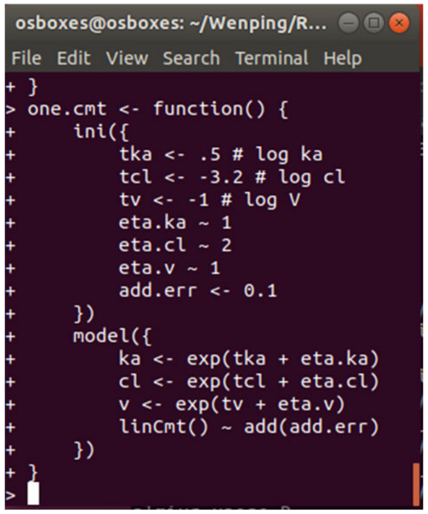

osboxes@osboxes: ~/Wenping/R...
File Edit View Search Terminal Help

```
+ }
> one.cmt <- function() {
+     ini({
+         tka <- .5 # log ka
+         tcl <- -3.2 # log cl
+         tv <- -1 # log V
+         eta.ka ~ 1
+         eta.cl ~ 2
+         eta.v ~ 1
+         add.err <- 0.1
+     })
+     model({
+         ka <- exp(tka + eta.ka)
+         cl <- exp(tcl + eta.cl)
+         v <- exp(tv + eta.v)
+         linCmt() ~ add(add.err)
+     })
+ }
>
```



nlmixr2

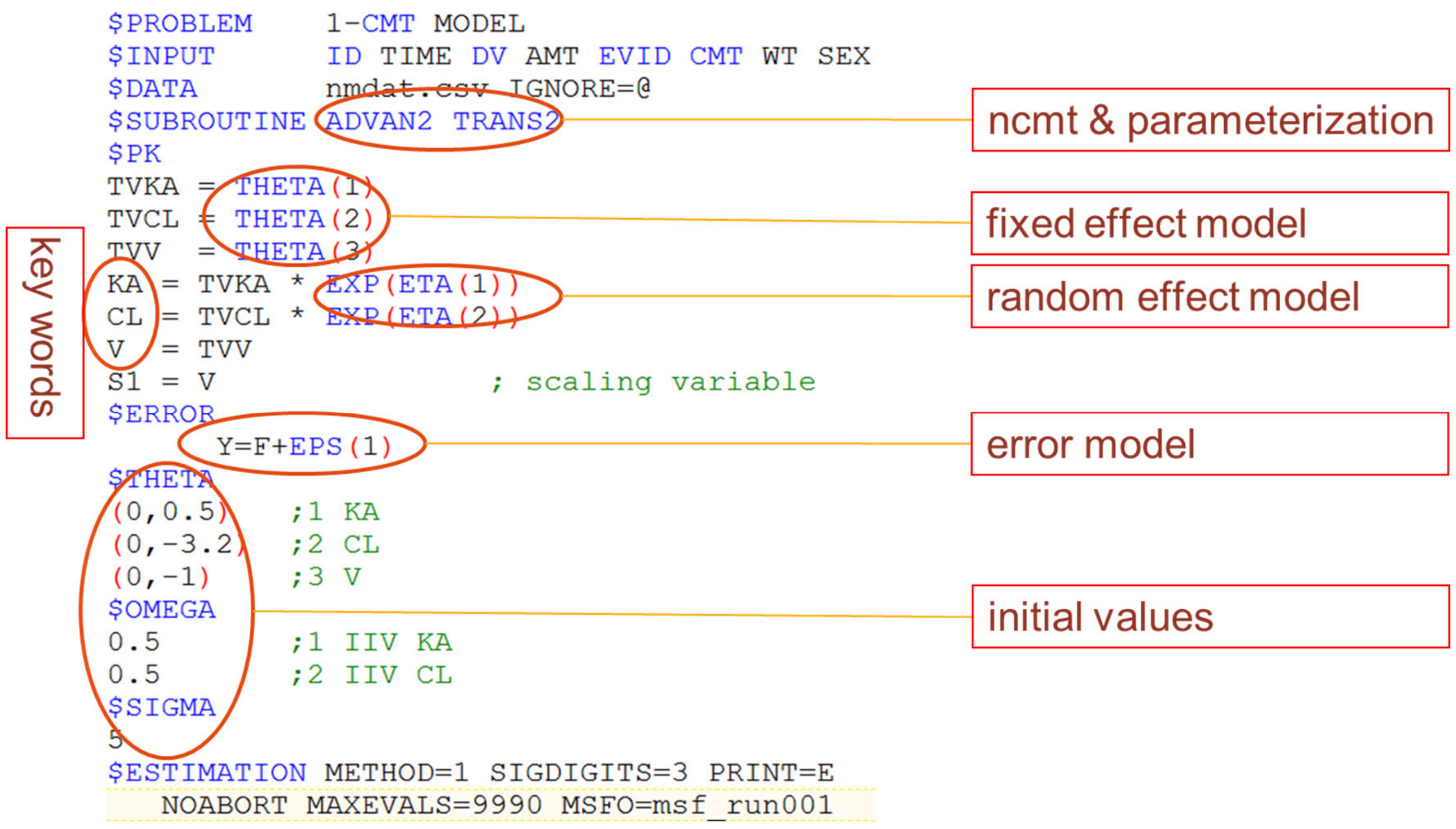
modeling syntax, running nlmixr2 models and nlmixr2 output


osboxes@osboxes: ~/Wenping/R...
File Edit View Search Terminal Help

```
+ }
> one.cmt <- function() {
+     ini({
+         tka <- .5 # log ka
+         tcl <- -3.2 # log cl
+         tv <- -1 # log V
+         eta.ka ~ 1
+         eta.cl ~ 2
+         eta.v ~ 1
+         add.err <- 0.1
+     })
+     model({
+         ka <- exp(tka + eta.ka)
+         cl <- exp(tcl + eta.cl)
+         v <- exp(tv + eta.v)
+         linCmt() ~ add(add.err)
+     })
+ }
>
```

nlmixr2

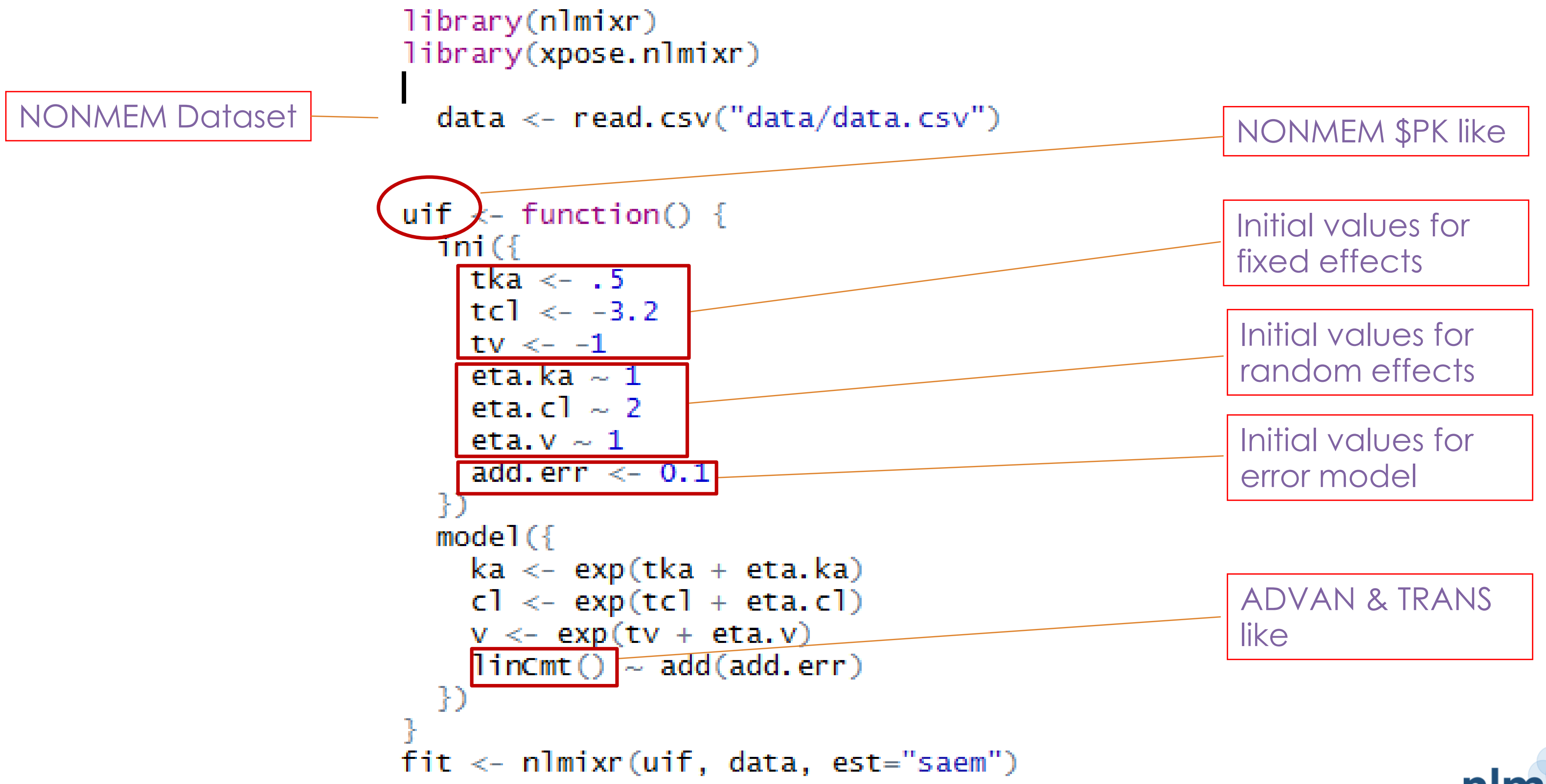
Anatomy of a NONMEM control stream for a popPK model



```
$PROBLEM     1-CMT MODEL
$INPUT       ID TIME DV AMT EVID CMT WT SEX
$DATA        nmdat.csv IGNORE=@
$SUBROUTINE ADVAN2 TRANS2
$PK
TVKA  = THETA(1)
TVCL  = THETA(2)
TVV   = THETA(3)
KA  = TVKA  * EXP(ETA(1))
CL  = TVCL  * EXP(ETA(2))
V   = TVV
S1  = V                     ; scaling variable
$ERROR
      Y=F+EPS(1)
$THETA
(0,0.5)      ;1  KA
(0,-3.2)     ;2  CL
(0,-1)       ;3  V
$OMEGA
0.5          ;1  IIV KA
0.5          ;2  IIV CL
$SIGMA
5
$ESTIMATION METHOD=1  SIGDIGITS=3  PRINT=E
   NOABORT MAXEVALS=9990 MSFO=msf_run001
```

nlmixr2

Anatomy of a nlmixr control stream for a popPK model compared to NONMEM


NONMEM Dataset
NONMEM $PK like
Initial values for fixed effects
Initial values for random effects
Initial values for error model
ADVAN & TRANS like

```
library(nlmixr)
library(xpose.nlmixr)

  data <- read.csv("data/data.csv")

uif <- function() {
  ini({
    tka <- .5
    tcl <- -3.2
    tv <- -1
    eta.ka ~ 1
    eta.cl ~ 2
    eta.v ~ 1
    add.err <- 0.1
  })
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl + eta.cl)
    v <- exp(tv + eta.v)
    linCmt() ~ add(add.err)
  })
}
fit <- nlmixr(uif, data, est="saem")
```

A nlmixr model has two main parts: initialization and model

Initialization ini({ })

label() or # (interactive only)

```
ini({
    lCl   <- 1.6; label("log Cl  (L/hr)")
    lVc    = log(90); label("log V  (L)")
    lKa    = fix(1)  #log Ka  (1/hr)
    add.sd = 0.2
    eta.Ka ~ 0.1 #IIV Ka
    eta.Cl + eta.Vc ~ c(0.1,
                        0.005, 0.1)
})
```

Lower triangular block matrix

- Population and Residual Estimates are defined using assign operators (**=**)
- Random Effects (ETAs) defined using a model formula (**~**; aka modelled by)

Model model({ })

model({ Relationship of Fixed/Random Pars First

```
model({
    Cl = exp(lCl + eta.Cl)
    Vc = exp(lVc + eta.Vc)
    KA = exp(lKa + eta.Ka)

    linCmt() ~ add(add.sd)

})
```

- Parameters defined based on ini block
- Fixed/Random relationships defined first
- Model (Solved/RxODE) defined next
- Unexplained error defined by formula (**~**)

nlmixr uses defined parameters to select 1, 2 or 3 solved compartment model with linCmt() → closed-form solutions

Solved System Parameterization Support

1 Compartment	2 Compartment	3 Compartment
Cl, V	Cl, V, Q, Vp	Cl, Vc, Q1, Vp1, Q2 Vp2
Kel, V	Kel, k12, k21, V	Kel, k12, k21, k13, k31, V
A, alpha	A, alpha, B, beta	A, alpha, B, beta, C, gamma

nlmixr also uses parameter aliases; Examples:
- V =Vc = V1 and Q = Cld.
- Parameter case does not matter

Parameter aliases are context dependent.
- The first can be Volume = Vc, (Can start with V2)
- Second numbered Volume = Vp
- All NONMEM style parameters are supported.

CMT #1 = depot (w/Ka) / central (without Ka) compartment

Model model({ })

```
model({
    Cl  = exp(lCl + eta.Cl)
    Vc  = exp(lVc + eta.Vc)
    KA  = exp(lKa + eta.Ka)
    Vp  = exp(lVp)
    Cld = exp(lCld)
    linCmt() ~ prop(prop.sd)
})
```

- 1 compartment solved model is specified by linCmt()
- 2 and 3 compartment model is also specified by linCmt()
- Type of model depends on provided parameters

https://nlmixr2.github.io/rxode2/articles/rxode2-model-types.html#solved-compartment-models

A nlmixr model block in case of no closed-form solution or PD model and ODE model block is required → linCmt cannot be used

Initialisation ini({ })

label() or # (interactive only)

```
ini({
    lCl    = 1.6; label("log Cl (L/hr)")
    lVc    = log(90); label("log V (L)")
    lKa    = 1     #log Ka (1/hr)
    prop.sd = 0.2
    eta.Ka ~ 0.1 #IIV Ka
    eta.Cl + eta.Vc ~ c(0.1,
                        0.005, 0.1)
})
```

Lower triangular block matrix

- Population and Residual Estimates are defined using assign operators (**=**)
- Random Effects (ETAs) defined using a model formula (**~**; aka modelled by)

Model model({ })

Relationship of Fixed/Random Pars First

```
model({
    Cl   = exp(lCl + eta.Cl)
    Vc   = exp(lVc + eta.Vc)
    KA   = exp(lKa + eta.Ka)
    kel = Cl / Vc
    d/dt(depot) = -KA*depot
    d/dt(centr) =  KA*depot-kel*centr
    cp = centr / Vc
    cp ~ prop(prop.sd)
})
```

➢ instead of linCMT

- Parameters defined based on ini block
- Fixed/Random relationships defined first
- Model (Solved/RxODE) defined next
- Unexplained error defined by formula (**~**)

Add Bioavailability (F) and lag time (alag) to the model

Initialisation ini({ })

label() or # (interactive only)

```
 ini({
    lCl    = 1.6; label("log Cl (L/hr)")
lVc    = log(90);   label("log V (L)")
    lKa    = 1          #log Ka (1/hr)
    lf     = log(1)
    lalag = log(0.5)
    prop.sd = 0.2
    eta.Ka ~ 0.1 #IIV Ka
    eta.Cl + eta.Vc ~ c(0.1,
                          0.005, 0.1)
 })
```

Lower triangular block matrix

- Population and Residual Estimates are defined using assign operators (**=**)
- Random Effects (ETAs) defined using a model formula (**~**; aka modeled by)

Model model({ })

Relationship of Fixed/Random Pars First

```
model({
    Cl   = exp(lCl + eta.Cl)
    Vc   = exp(lVc + eta.Vc)
    KA   = exp(lKa + eta.Ka)
    fD   = exp(lf)
    lagD = exp(lalag)
    kel = Cl / Vc
    d/dt(depot) = -KA*depot
    alag(depot) = lagD
    f(depot)    = fD
    d/dt(centr) = KA*depot-kel*centr
    cp = centr / Vc
    cp ~ prop(prop.sd)
})
```

Can also add rate/dur for modeled duration and rate

nlmixr2

Residual Error models and Multiple Endpoints

Error Model	Coding	Supported By
Additive/Normal	Y ~ add(add.sd)	nlme, fo, foi, foce, focei, saem
Proportional	Y ~ prop(prop.sd)	nlme, fo, foi, foce, focei, saem
Additive + Proportional	Y ~ add(add.sd) + prop(prop.sd)	nlme, fo, foi, foce, focei, saem
Lognormal/Exponential ***Note*: normal scale OBJF**	Y ~ lnorm(lnorm.sd)	fo, foi, foce, focei, saem
Power Model	Y ~ pow(pow.sd, pow)	fo, foi, foce, focei, saem
Additive + Power	Y ~ add(add.sd) + pow(pow.sd, d)	fo, foi, foce, focei, saem
Box-Cox transform both sides	Y ~ add(add.sd) +boxCox(lambda)	fo, foi, foce, focei, saem
Yeo-Johnson transform both sides	Y ~ add(add.sd) + yeoJohnson(lambda)	fo, foi, foce, focei, saem

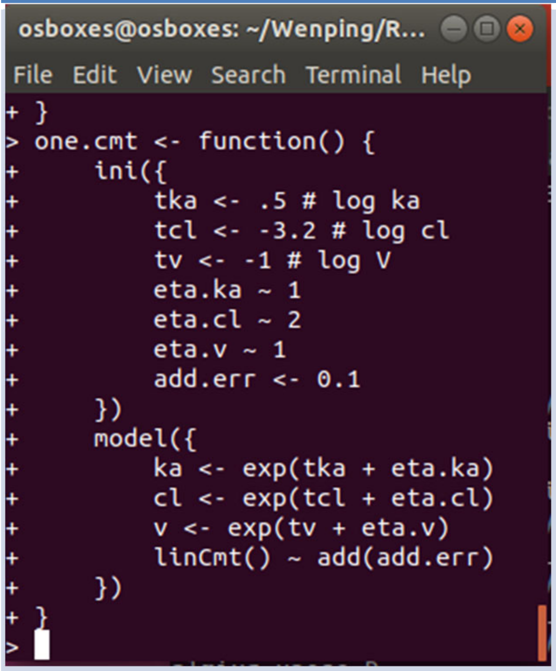
Multiple Endpoint:

```
PK ~ add(add.sd) + prop(prop.sd)  | depot
PD ~ add(pd.sd)                   | err
```

Now generalized llik for focei

Finalizing and checking a nlmixr model verifies nlmixr detects the correct solved model (or RxODE model), as well as showing the parsed initial estimates

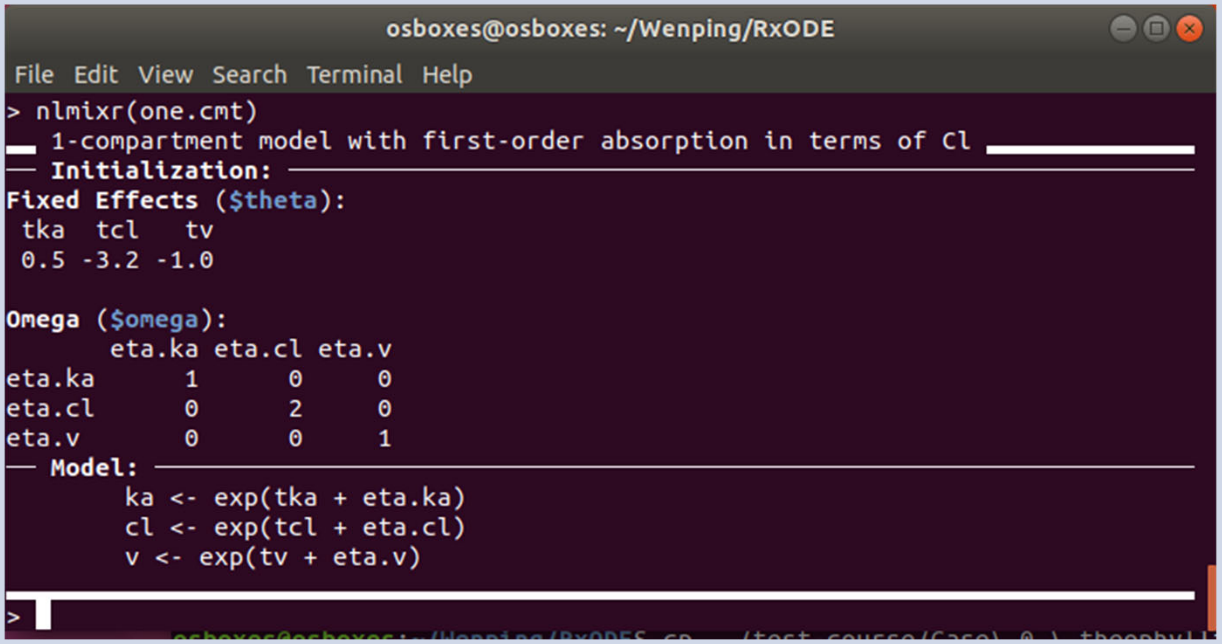
Finalising models


osboxes@osboxes: ~/Wenping/R...
File Edit View Search Terminal Help

```
+ }
> one.cmt <- function() {
+     ini({
+         tka <- .5 # log ka
+         tcl <- -3.2 # log cl
+         tv <- -1 # log V
+         eta.ka ~ 1
+         eta.cl ~ 2
+         eta.v ~ 1
+         add.err <- 0.1
+     })
+     model({
+         ka <- exp(tka + eta.ka)
+         cl <- exp(tcl + eta.cl)
+         v <- exp(tv + eta.v)
+         linCmt() ~ add(add.err)
+     })
+ }
>
```

To finalize a model, put the **`ini`** and **`model`** in a named function

Checking how the model is parsed


osboxes@osboxes: ~/Wenping/RxODE
File Edit View Search Terminal Help

```
> nlmixr(one.cmt)
── 1-compartment model with first-order absorption in terms of Cl ──────────
── Initialization: ─────────────────────────────────────────────────────────
Fixed Effects ($theta):
 tka   tcl    tv
 0.5 -3.2 -1.0

Omega ($omega):
       eta.ka eta.cl eta.v
eta.ka      1      0     0
eta.cl      0      2     0
eta.v       0      0     1
── Model: ─────────────────────────────────────────────────────────────────
        ka <- exp(tka + eta.ka)
        cl <- exp(tcl + eta.cl)
        v <- exp(tv + eta.v)

>
```

By calling **`nlmixr`** on the named R function, it will tell you how **nlmixr** parsed the model; This is especially useful in checking what solved system **nlmixr** detected before running the entire model

Fitting nlmixr models takes the estimation method (with its options) and produces a nlmixr combined dataset/fit object

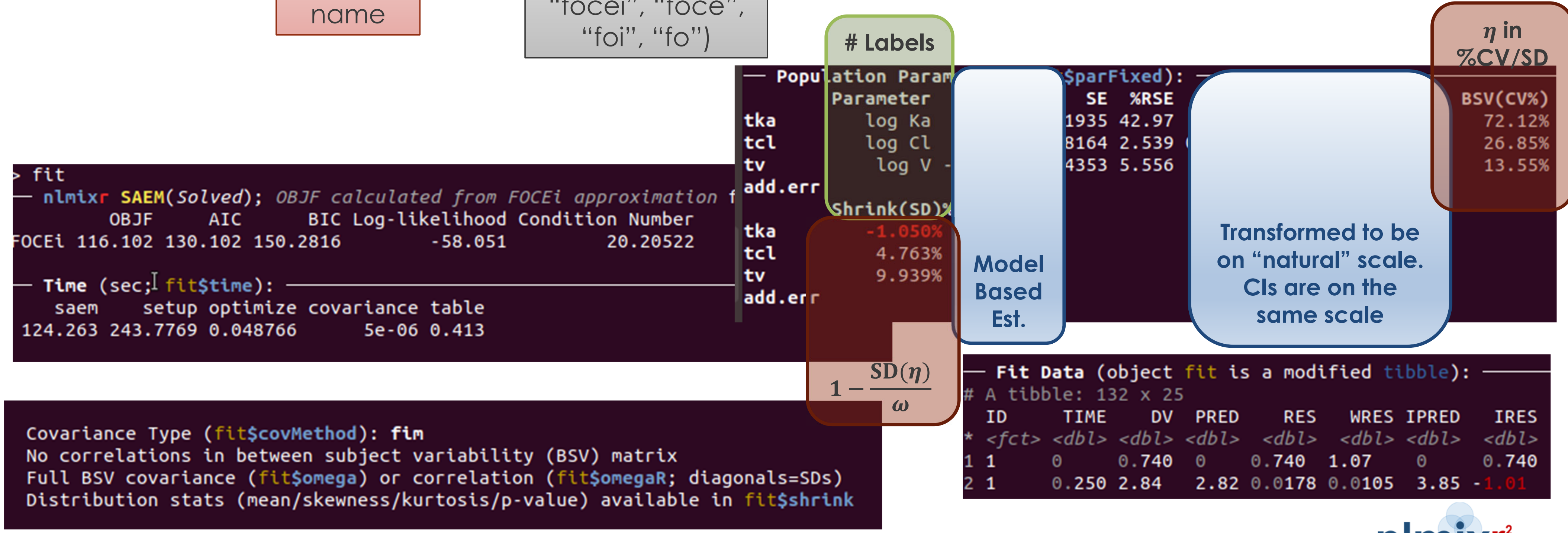
```
fit <- nlmixr(one.cmt, data, est = "saem", table=tableControl(cwres=TRUE, npde=TRUE))
```

Assigned R object

Function Name is run name

Estimation methods = ("nlme", "saem", "focei", "foce", "foi", "fo")

Optional if cwres and or npde calculations wanted



```
> fit
── nlmixr SAEM(Solved); OBJF calculated from FOCEi approximation f
         OBJF     AIC      BIC Log-likelihood Condition Number
FOCEi 116.102 130.102 150.2816       -58.051         20.20522

── Time (sec; fit$time): ──────────────────────────────
    saem    setup optimize covariance table
 124.263 243.7769 0.048766      5e-06 0.413
```

```
── Population Param           $parFixed): ──
            Parameter            SE   %RSE                          BSV(CV%)
tka            log Ka          1935  42.97                            72.12%
tcl            log Cl          8164  2.539 (                          26.85%
tv             log V -         4353  5.556                            13.55%
add.err

            Shrink(SD)%
tka           -1.050%
tcl            4.763%
tv             9.939%
add.err
```

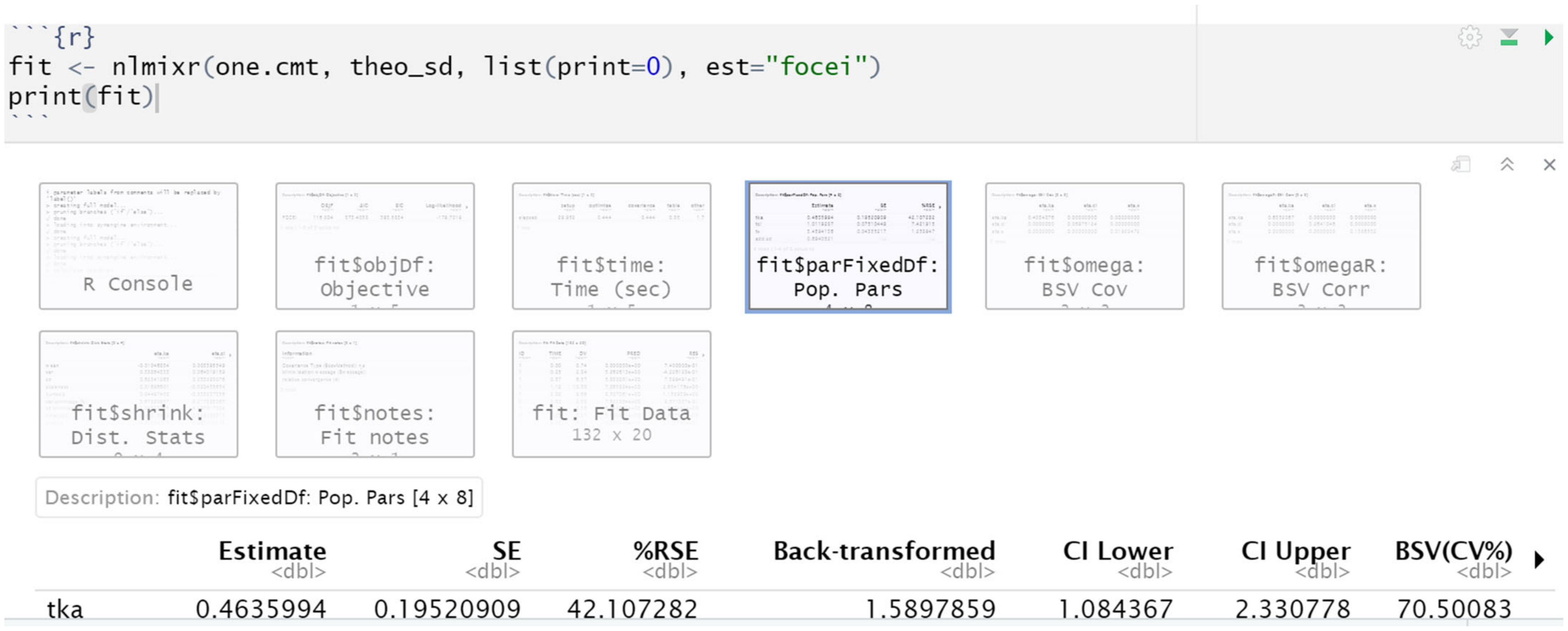
```
Covariance Type (fit$covMethod): fim
No correlations in between subject variability (BSV) matrix
Full BSV covariance (fit$omega) or correlation (fit$omegaR; diagonals=SDs)
Distribution stats (mean/skewness/kurtosis/p-value) available in fit$shrink
```

```
── Fit Data (object fit is a modified tibble): ──────────
# A tibble: 132 x 25
  ID     TIME    DV  PRED    RES  WRES IPRED   IRES
* <fct> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>
1 1     0     0.740 0      0.740 1.07  0      0.740
2 1     0.250 2.84  2.82  0.0178 0.0105 3.85 -1.01
```

# Labels
Model Based Est.
Transformed to be on "natural" scale. CIs are on the same scale
η in %CV/SD
$1 - \frac{\mathrm{SD}(\eta)}{\omega}$

nlmixr2

In Rstudio's Rmarkdown or notebook, the output is similar but in tabular form that is easier to click through



~~~
```{r}
fit <- nlmixr(one.cmt, theo_sd, list(print=0), est="focei")
print(fit)
```
~~~

Description: fit$parFixedDf: Pop. Pars [4 x 8]

	Estimate <dbl>	SE <dbl>	%RSE <dbl>	Back-transformed <dbl>	CI Lower <dbl>	CI Upper <dbl>	BSV(CV%) <dbl>
tka	0.4635994	0.19520909	42.107282	1.5897859	1.084367	2.330778	70.50083

Inclusion of Covariates into a SAEM nlmixr model

Initialization ini({ })

```
 ini({
    lCl     = 1.6      #log Cl  (L/hr)
    lVc     = log(90)  #log V  (L)
    lKa     = fix(1)   #log Ka  (1/hr)
    beta.wt = 0.75     #estimate of covariate effect
    prop.sd = c(0,0.2,1)
    eta.Ka   ~ 0.1 #IIV Ka
    eta.Cl + eta.Vc ~ c(0.1,
                        0.005, 0.1)
})
```

SCM covariate building:

covarSearchAuto()

```
Cl = exp(   tCl   +   eta.Cl   +   beta.wt * lnWt70 )
```

Fixed or Population Parameter

Random or Individual Parameter

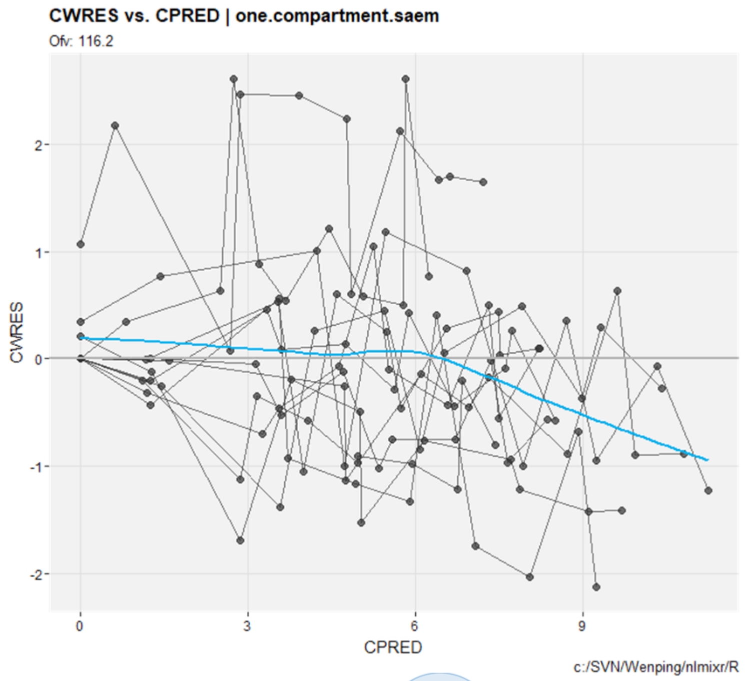
Covariate Estimate times transformed covariate

$$\exp(t_{Cl} + e_{Cl})\left(\frac{\mathrm{WT}}{70}\right)^{\mathrm{WT}_{CL}}$$

$$\exp(t_{Cl} + e_{Cl} + \mathrm{WT}_{CL} \cdot \mathrm{logWt70})$$

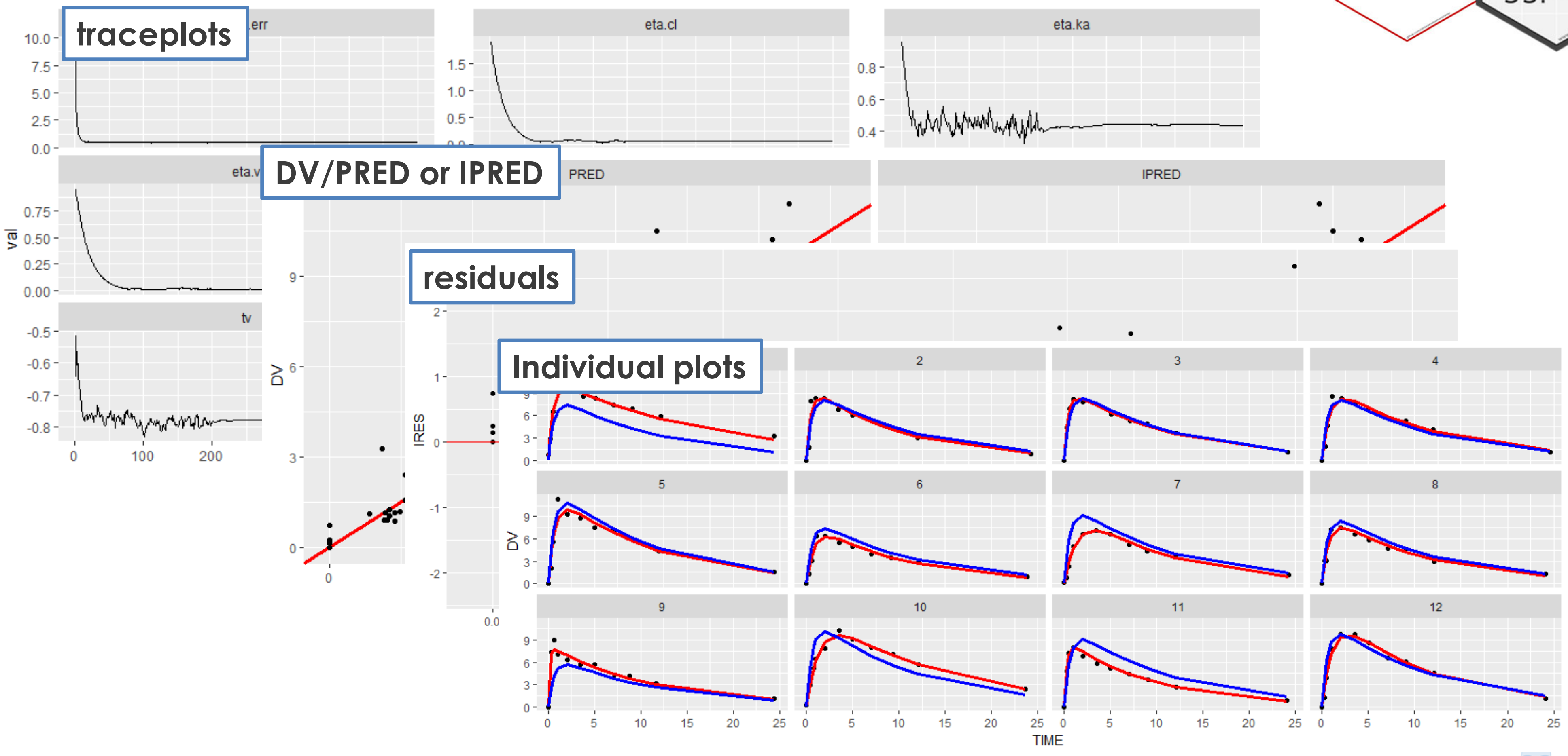
nlmixr2

diagnostic plots from a nlmixr model



nlmixr2

Simple goodness of fit plots can be produced by a simple plot(fit)



Resources, documentation and further reading

- Home of nlmixr2, rxode2, xpose.nlmixr2, support packages (most recent versions)
  - https://github.com/nlmixr2 New version of nlmixr
- Documentation: continually evolving
  - https://nlmixr.org/
- Open course material:
- Twitter: @nlmixr
- LinkedIn: https://www.linkedin.com/groups/8621368/

Estimating initial compartmental PK model parameters with PKNCA

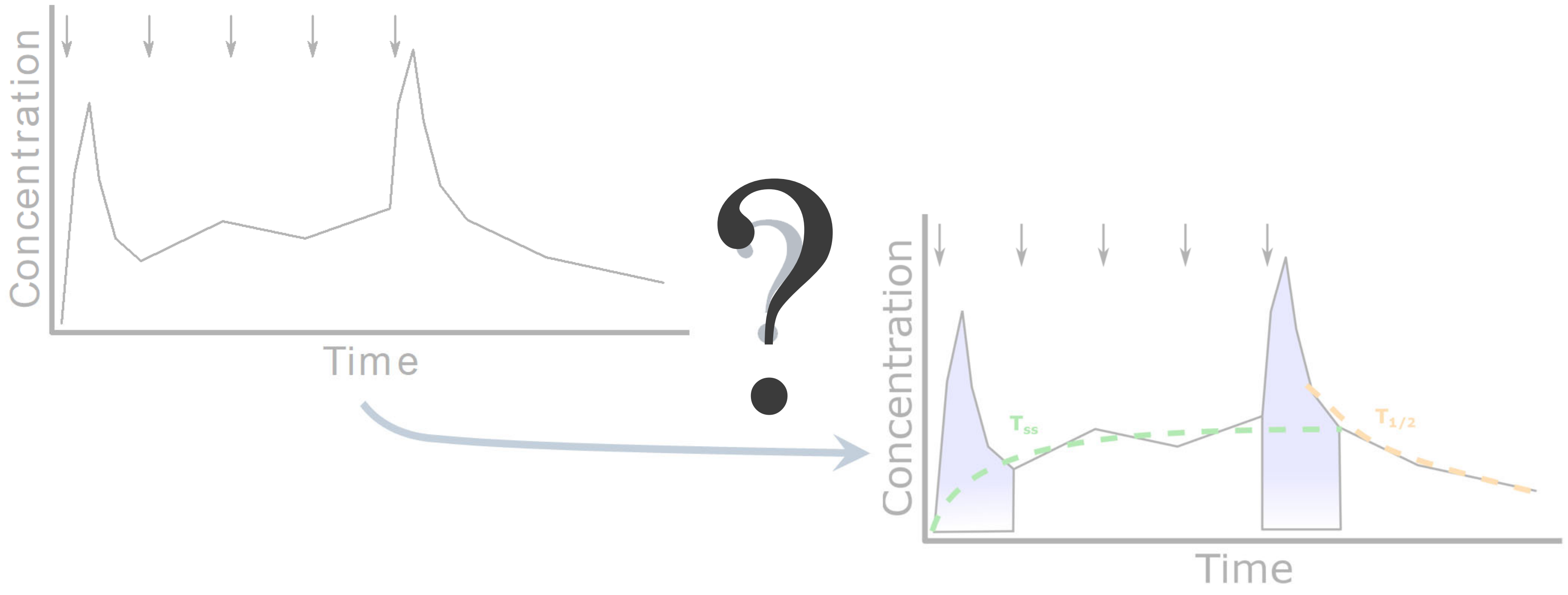
Motivation

Photo from https://www.pexels.com/photo/father-surrounded-by-his-children-while-working-8188698/

NCA in 5(+4) Lines of R

```r
library(PKNCA)
# Load the PK concentration data
d_conc <- datasets::Theoph
d_conc$treatment <- ifelse(d_conc$Dose < median(d_conc$Dose), yes = "Low dose", no =
"High dose")
# Load the dosing data
d_dose <- d_conc[d_conc$Time == 0, ]
# Create a concentration object. (Note that any number of grouping
# levels is supporting; you are not restricted to this list.)
obj_conc <- PKNCAconc(d_conc, concentration~time|treatment+subject)
# Create a dosing object.
obj_dose <- PKNCAdose(d_dose, dose~time|treatment+subject)
# Combine the concentration and dosing information; automatically
# define the intervals for NCA calculation; optionally give unit information here.
obj_data <- PKNCAdata(obj_conc, obj_dose)
# Calculate the NCA parameters
obj_results <- pk.nca(obj_data)
# Summarize the results
summary(obj_results)
```

Any Questions?



nlmixr2

What does PKNCA do (and not do)?

- **Organizes** concentration/time and dose/time data
- **Predicts** what you most likely need from NCA parameters from the concentration and dosing data.
- **Allows user control** of all NCA parameter and summary calculations
  - If you want a type of control that it doesn't allow, please make a feature request.
  - Does everything according to your business rules
- **Calculates** all (standard) NCA parameters
  - Optionally includes units and unit conversions in calculations
  - Targets SDTM-like PK parameter terms
- **Summarizes** the parameters

With babelmixr2 and PKNCA, nlmixr2 PK models can automatically have good initial estimates

- Using the new **babelmixr2** library, **nlmixr2** and PKNCA team up to:
  - Provide initial estimates for 1-, 2-, or 3-compartment population PK models
  - Automate unit conversion (did you remember to put the correct scaling factor in your model?)

How are the estimates made?

- `ka`: T_{max} is 4 absorption half-lives: $ka = \frac{\log(2)}{\frac{tmax}{4}}$
- `vc`: the inverse of dose-normalized C_{max} estimates central volume: $vc = \frac{1}{cmax_{dn}}$
- `cl`: clearance is clearance… of course
- `vp`, `vp2`, `q`, and `q2` are multiples of vc and cl (multiples are controlled by `pkncaControl()` arguments).

- Bounds and estimates are set based on percentiles of NCA parameters:
  - Lower bound: 0.1× 1st percentile,
  - Estimate: 1× 50th percentile, and
  - Upper bound: 10× the 99th percentile
- Additionally, `ka` is the minimum of the above or a lower bound of 0.03 and an estimate of 3.
- Unit conversion converts dose units/ volume units to concentration units.

Requirements for automatic initial parameter estimates with babelmixr2

- **nmixr2**-formatted data
  - No need to change it for PKNCA, that will be done automatically
- A model with parameters named `ka`, `vc`, `vp`, `vp2`, `cl`, `q`, and `q2`.
  - Only the needed parameters need to be included (e.g. no ka for an IV dosing model)
  - Parameters can be in either in the `ini()` or `model()` block.
  - **babelmixr2** will autodetect any parameter transformations (e.g. log-scale estimated as `lcl` in `ini()` for `cl` in `model()`).

How do I use babelmixr2 to get initial estimates by PKNCA?

```
library(babelmixr2)
fit <-
 nlmixr2(
  one.cmt,
  data = data,
  est = "pknca",
  control =
   pkncaControl(
    concu = "mg/L",
    doseu = "mg/kg",
    timeu = "hr",
    volumeu = "L/kg"
   )
  )
```

Compartmental PK model with `ka`, `vc`, `vp`, `vp2`, `cl`, `q`, and `q2`

Normal nlmixr2-formatted data

est = "pknca" will use PKNCA to create new initial estimates

pkncaControl() allows setting options to control PKNCA estimation for the new initial estimates

What can pkncaControl() do?

```
pkncaControl(
  concu = "mg/L",
  doseu = "mg/kg",
  timeu = "hr",
  volumeu = "L/kg",
  dvParam = "cp",
  groups = c(),
  ncaData = NULL,
  ncaResults = NULL
)
```

Set concentration units (for the DV column)

Set dosing units (for the AMT column)

Set time units (for the TIME column)

Set volume units (for vc, vp, and vp2 parameter estimates)

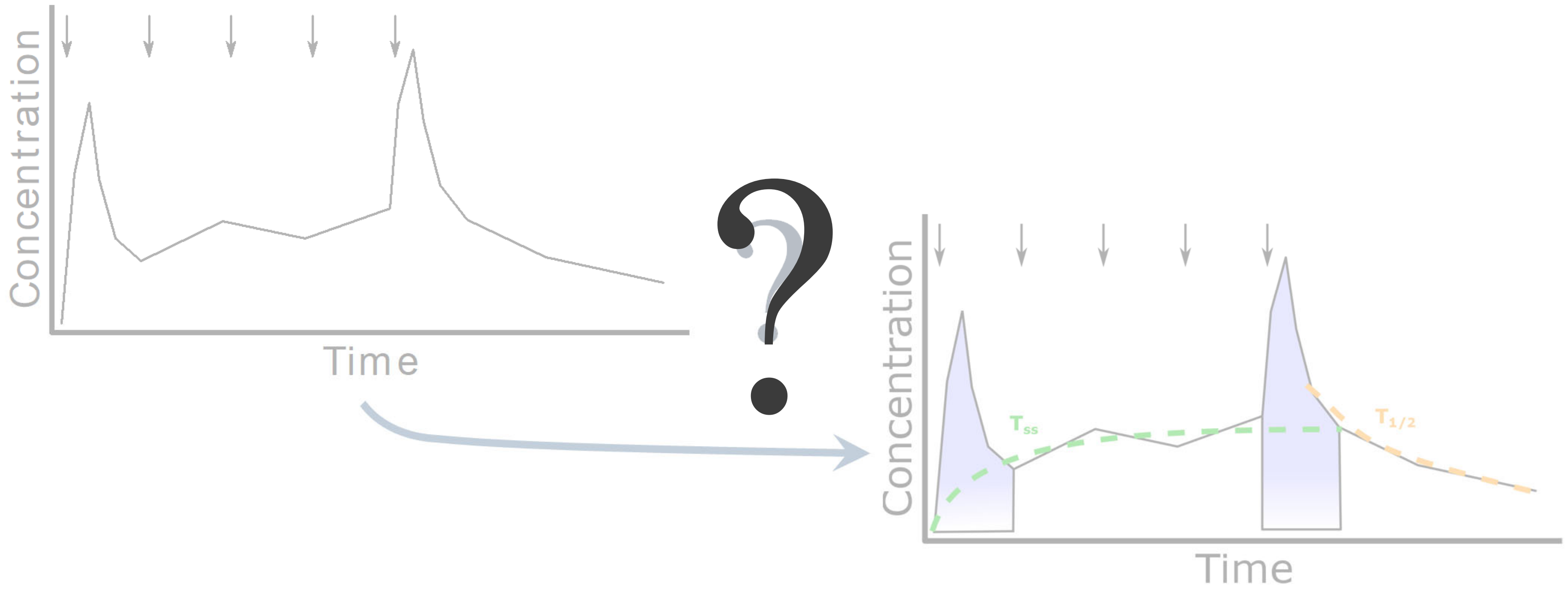
The parameter name for the dependent variable in the model (used for automatic unit conversion)

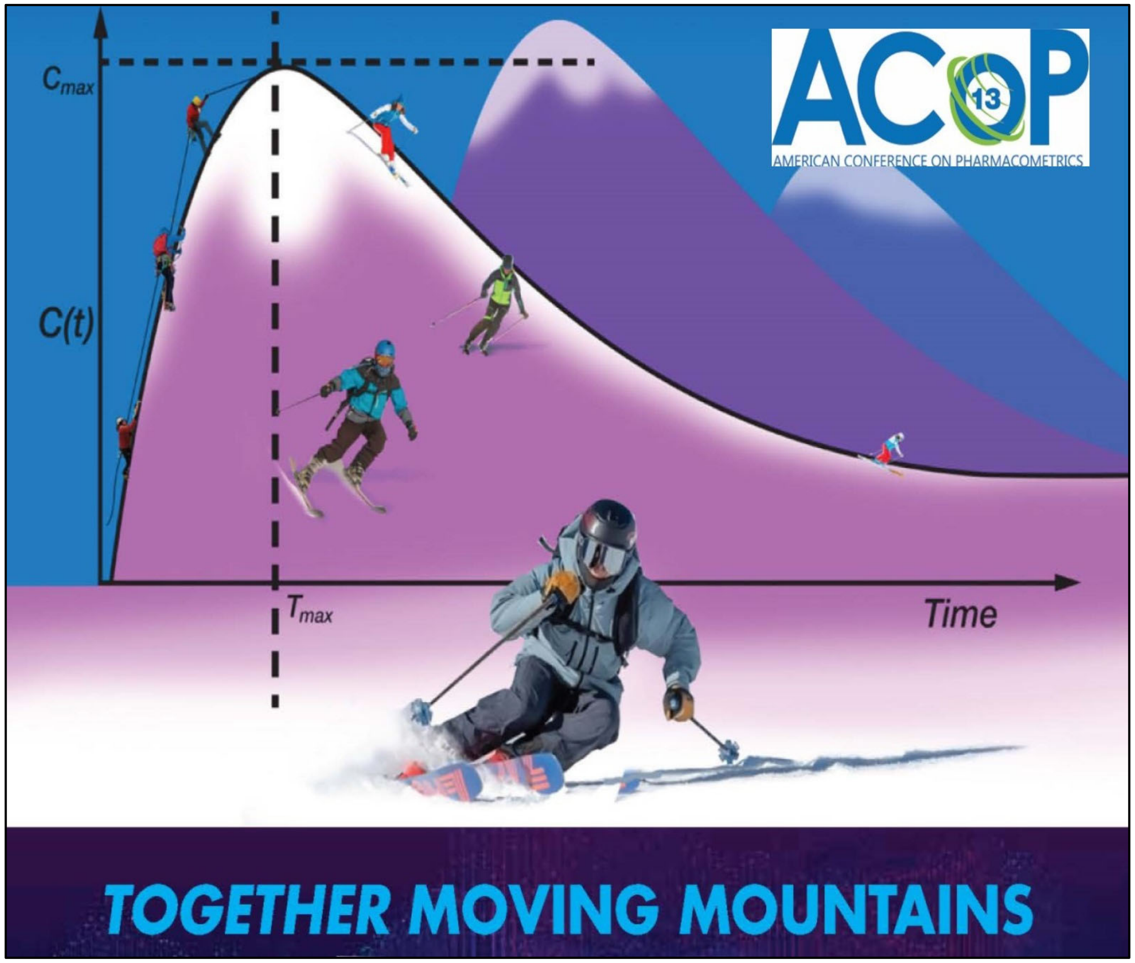
Column names for grouping in the PKNCA calculation

Data to use for NCA calculation (e.g. in case some studies have dense vs sparse data)

Provide precalculated PKNCA results (in case the automation does not calculate for you). Calculated parameters must include `tmax`, `cmax.dn`, and `cl.last`.

Any Questions?




C_{max}
C(t)
T_{max}
Time
ACoP 13
AMERICAN CONFERENCE ON PHARMACOMETRICS
TOGETHER MOVING MOUNTAINS

Aurora, CO

October 30 - November 2, 2022