nlmixr Cheat Sheet

October 2018

github.com/nlmixrdevelopment

Getting nlmixr

What you need first

- R 3.3 or better
  - RxODE
  - PreciseSums
  - SnakeCharmR
  - nlmixr
- Rtools (if you use Windows)
- Python with SymPy

See our GitHub homepage for a detailed installation guide and links to nlmixr installers!

Optional extras

xpose.nlmixr: Graphical diagnostics using xpose

shinyMixR: A GUI for building nlmixr models in shiny

Solved systems

Linear compartmental PK models with oral and IV dosing all have closed-form solutions similar to NONMEM ADVANs.

model

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

The `linCmt()` term replaces the ODEs. nlmixr will guess the model form from the parameters specified.

Residual error

Additive, proportional and combined additive and proportional error models are available.

model

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

Writing models

```
model <- function(){
  ini({
    tka <- log(1.5)
    tcl <- log(4)
    tv  <- log(20)
    eta.ka ~ 0.5
    eta.cl ~ 0.5
    eta.v  ~ 0.5
    add.err <- 0.1
  })
  model({
    ka <- exp(tka + eta.ka)
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv + eta.v)

    d/dt(depot) = -ka * depot
    d/dt(cent) =  ka * depot -
          cl / v * cent
    cp = cent / v
    cp ~ add(add.err)
  })
}
```

Fixed effects (<- or =)

Random effects (~)

Residual error (<-)

Initial estimates

Model parameters

ODEs

Concentration

Residual error

Model

Models are defined as functions, with `ini` (initial estimates) and `model` (model) blocks. Parameters are best defined on the log scale. Assignments can use `<-` or `=`. Random effects are expressed as variances using the tilde (~). Bounds are supported for FOCEi (but not NLME or SAEM currently), e.g.

ini

```
tcl <- c(-3, 0.1, 5) # log scale (FOCEi only)
```

Error blocks

Parameter correlations are expressed as triangular blocks (zeroes should not be used):

ini

```
eta.cl + eta.v ~ c(0.1,
                   0.005, 0.1)
```

SAEM Covariates

SAEM Covariates must pre-transformed in the dataset to be linear. Initial estimates are necessary. Often this requires covariates to be on the log-scale. This is only required with SAEM.

ini

```
coveff1 <- c(-5, 0.1, 5)
coveff2 <- c(-0.5, -0.1, 0)
Data: logWtCov = log(wt/70)
```

model

```
cl <- exp(tcl + coveff1*logWt + eta.cl)
v2 <- exp(tv2 + coveff2*CatCov + eta.v2)
```

Running models

```
nlmixr(
  object,
  data,
  est = "nlme",
  control = list(),
  calc.resid = TRUE,
  ...)
```

Model

Estimation method

Calculate conditional weighted residuals

NONMEM/RxODE data

Control parameters

Estimation methods

NLME, SAEM, FOCEi, FOCE, FOi, FO & posthoc methods

est = "nlme"

Based on the Pinheiro-Bates `nlme::nlme` method in base R. Fast, but known to produce suboptimal results under some circumstances.

control = nlmeControl()

maxIter	Max number of iterations (50)
pnlsMaxIter	Max iterations for PNLS optimization (7)
msMaxIter	Max iterations for NLM optimization (50)
minScale	Minimal scaling factor for PNLS (0.001)
tolerance	Convergence tolerance ($1e^{-6}$)
niterEm	Iterations for EM step (25)
pnlsTol	Tolerance for PNLS step ($1e^{-3}$)
msTol	Tolerance for MS step ($1e^{-7}$)
returnObject	Return object after unsuccessful convergence (F)
msVerbose	Show trace details for NLM (F)
apVar	Calculate approximate covariance matrix (T)
.relStep	Relative step size for numerical derivatives
minAbsParApVar	Minimum absolute parameter value in the approximate variance calculation (0.05)
opt	Optimizer (`"nlminb"` [default] or `"nlm"`)
natural	Use natural parametrization for general positive-definite matrices (`pdSymm`) in `reStruct` (T)
sigma	Fixed residual error. Calculate if NULL (default) or 0
...	Additional arguments to `nlminb`

est = "saem"

An implementation of the stochastic approximation expectation-maximization algorithm. No termination criteria, can be slow when using ODEs.

control = saemControl()

seed	Random seed (99)
n.burn	Number of iterations in the SA (burn-in) step (200)
n.em	Number of iterations in the EM step (300)
nmc	Number of Markov chains (3)
nu	Numbers of transitions of kernels used in the Hasting-Metropolis algorithm. Default is c(2,2,2) representing 40 for each transition initially (each multiplied by 20)
atol	Absolute convergence tolerance ($1e^{-6}$)
rtol	Relative convergence tolerance ($1e^{-4}$)
stiff	Flag for stiff ODE systems (T, uses LSODA)
transit_abs	Flag for transit absorption model (F)
print	Iterations to complete before printing to console (1)
...	Additional arguments

Running Models

Estimation methods

est = "focei", "foce", "foi", "fo"

These methods are based on our interpretation of the NONMEM routines

control = foceiControl()

outerOpt	Outer Optimization Routine (bobyqa)
sigdig	Controls tolerances of estimation and ODE solving routines. Not the same as NONMEM sigidig parameter but with similar meaning (4)
scaleC	Custom scaling for focei optimization
covMethod	Covariance method "r,s" uses sandwich matrix, "r" uses hessian matrix and "s" uses cross-product matrix. "" does not calculate covariance/standard errors
maxOuterIterations	Maximum number of outer iterations before stopping estimation
maxInnerIterations	Maximum number of inner iterations before stopping individual eta estimation.
...	

est = "posthoc"

Uses posthoc step of FOCEi algorithm, it similar to using foceiControl(maxOuterIterations=0)

Table Options

table = tableControl()

This controls what additional table output are included in the final nlmixr model

table = tableControl()

cwres	Boolean indicating if you need to calculate conditional weighted residuals (CWRES). On by default for FOCE(i) routines. This will also generate WRES, CPRED and CRES. Additionally this will add the FOCEi objective function value
npde	Calculate npde residuals (NPDE). This will also generate EPRED and ERES
nsim	Number of simulations used for NPDE (default 300)
ties	Boolean indicating if noise will be added to avoid ties in NPDE calculation (TRUE)
seed	Random seed to use for npde calculation (1009)
...	

Adding Table items after fit

You can add conditional weighted residuals, weighted residuals and NPDE to any fit by the following functions

Adding Tables

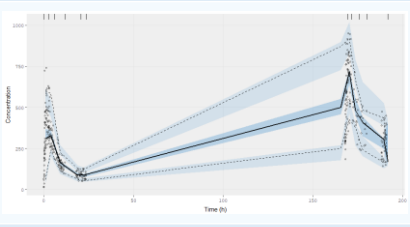
```
fit <- fit %>% addCwres()
fit <- fit %>% addNpde()
```

VPCs: vpc

nlmixr uses the simulation capabilities of RxODE and the vpc package to generate VPCs directly from the fitted model object:

nlmixr

```
vpc_ui(myfit, n=600,show=list(obs_dv=T),
   log_y=T, xlab="Time (h)",
   ylab="Concentration (mg/L)")
```



Most useful VPC options

`fit`	nlmixr fit object
`n`	Number of simulation iterations
`bins`	Either "density", "time", or "data", "none", or one of the approaches available in `classInterval()` such as "jenks" (default) or "pretty", or a numeric vector specifying the bin separators
`n_bins`	When using the "auto" binning method, what number of bins to use
`bin_mid`	Either "mean" for the mean of all timepoints (default) or "middle" to use the average of the bin boundaries
`show`	What to show in VPC (obs_dv, obs_ci, pi, pi_as_area, pi_ci, obs_median, sim_median, sim_median_ci). See example
`stratify`	Character vector of stratification variables (max 2)
`smooth`	"Smooth" the VPC (connect bin midpoints) or show as rectangular boxes (default T)
`pred_corr`	Perform prediction-correction (default F)
`pi`	Simulated prediction interval to plot. Default is c(0.05, 0.95)
`ci`	Confidence interval to plot. Default is (0.05, 0.95)
`facet`	"wrap", "columns", or "rows"
`log_y`	Logarithmic y-axis? (default F)
`xlab`	Label for x-axis
`ylab`	Label for y-axis
`title`	Title
`uloq`	Upper limit of quantification (default NULL)
`lloq`	Lower limit of quantification (default NULL)
`vpc_theme`	Theme. Expects list of class `vpc_theme` created with function `vpc_theme()`

Graphical diagnostics: xpose

Loading a model into xpose

In order to use the functionality of xpose, we first need to convert our nlmixr model object into an xpose database using the xpose.nlmixr package.

xpose.nlmixr

```
xpdb <- xpose_data_nlmixr(myfit,
          xp_theme = theme_xp_nlmixr())
```

The `xp_theme` option allows a theme object (defining how plots will be drawn) to be specified.

Plot layers and aesthetics

Besides being able to manipulate xpose graphs in the same ways as ggplot2 graphs using layers, plot aesthetics can be directly specified using `layer_argument`, where `layer` is the layer, and `argument` is the argument applying to it.

xpose

```
dv_vs_pred(xpdb,
   point_color="blue")
```

Layers for scatterplots	
`point`	Options for `geom_point`
`line`	Options for `geom_line`
`guide`	Options for `geom_abline`
`smooth`	Options for `geom_smooth`
`text`	Options for `geom_text`
`xscale`	Options for `scale_x_continuous` or `scale_x_log10`
`yscale`	Options for `scale_y_continuous` or `scale_y_log10`

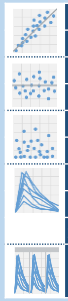
Layers for distributions	
`histogram`	Options for `geom_histogram`
`density`	Options for `geom_density`
`rug`	Options for `geom_rug`
`xscale`	Options for `scale_x_continuous` or `scale_x_log10`
`yscale`	Options for `scale_y_continuous` or `scale_y_log10`

Access functions

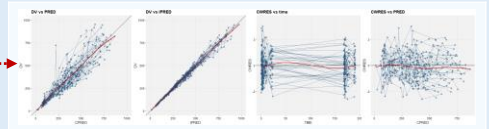
`get_code(xpdb)`	Display model
`get_data(xpdb)`	Extract data
`print(xpdb)`	Display summary of xpose data object

Icons and content for xpose courtesy of Ben Guiastrennec and the xpose team! Xpose can do much more than this – get the official cheat sheet at uupharmacometrics.github.io/xpose/articles/cheatsheet.pdf

Basic goodness-of-fit

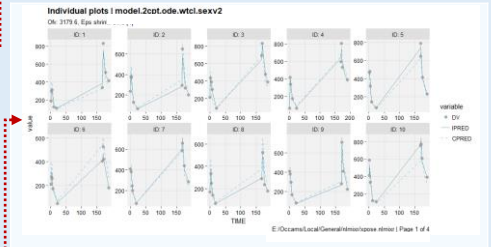


```
dv_vs_pred(xpdb)
dv_vs_ipred(xpdb)
res_vs_idv(xpdb, res="CWRES")
res_vs_pred(xpdb, res="CWRES")
absval_res_vs_idv(xpdb, res="CWRES")
absval_res_vs_pred(xpdb, res="CWRES")
dv_vs_idv(xpdb, group="ID")
ipred_vs_idv(xpdb, group="ID")
pred_vs_idv(xpdb, group="ID")
dv_preds_vs_idv(xpdb)
```

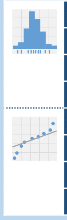


Individual plots

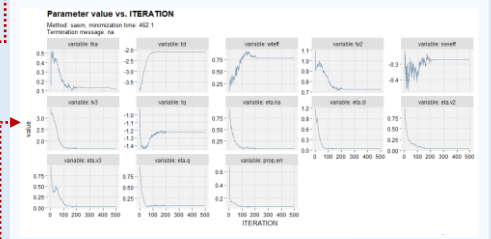
```
ind_plots(xpdb)
```



Distributions



```
prm_distrib(xpdb)
eta_distrib(xpdb)
cov_distrib(xpdb)
res_distrib(xpdb, res="CWRES")
prm_qq(xpdb)
eta_qq(xpdb)
cov_qq(xpdb)
res_qq(xpdb, res="CWRES")
```



Iteration trace plots

```
prm_vs_iteration(xpdb)
```

Editing and subsetting data

Editing/filtering data in xpose is performed by dplyr.

`filter`	Subset data based on logical condition(s)
`mutate`	Add, modify or remove variables

xpose

```
xpdb %>%
  filter(WT>70) %>%
  dv_vs_pred()
```

Plot types

The xpose package supports different plot types, according to the type of data being plotted.

xpose

```
dv_vs_pred(xpdb, type="pls")
eta_distrib(xpdb, type="hdr")
```

Scatterplots	
`p`	Point
`l`	Line
`s`	Smooth
`t`	Text

Distributions	
`h`	Histogram
`d`	Density line
`r`	Rug

Editing data types

xpose.nlmixr tries to assign variables to types automatically, and often this works well. Sometimes manual adjustments are needed, though.

`list_vars(xpdb)`	Display variable assignments
`set_var_types(xpdb, ...)`	Modify variable assignments

xpose

```
list_vars(xpdb1)
xpdb2 <- set_var_types(xpdb1, .problem = 1,
   catcov='sex')
```