# Case study 1: developing a two-compartment PK model through RsNLME

## Introduction

- R is one of the most widely used softwares among pharmacometricians to perform data manipulation/visualization and statistical analysis.
- RsNLME provides a R interface to the Phoenix NLME engine to enable users to
  - Define PK/PD models via R objects (package **RsNlme**).
  - Use the "Initial Estimates" shiny app to visually determine a set of reasonable initial values for fixed effects (package **RsNlme**).
  - Perform estimation and simulation in a R environment with the capability of parallelizing the runs using Multicore, MPI and Grids (SGE/Torque/LSF) in-house or hosted on AWS (package **Certara.NLME8**).
  - Access the xpose graphics library for PK/PD models by creating compatible database from NLME results (package **Xpose.Nlme**).

## Objectives

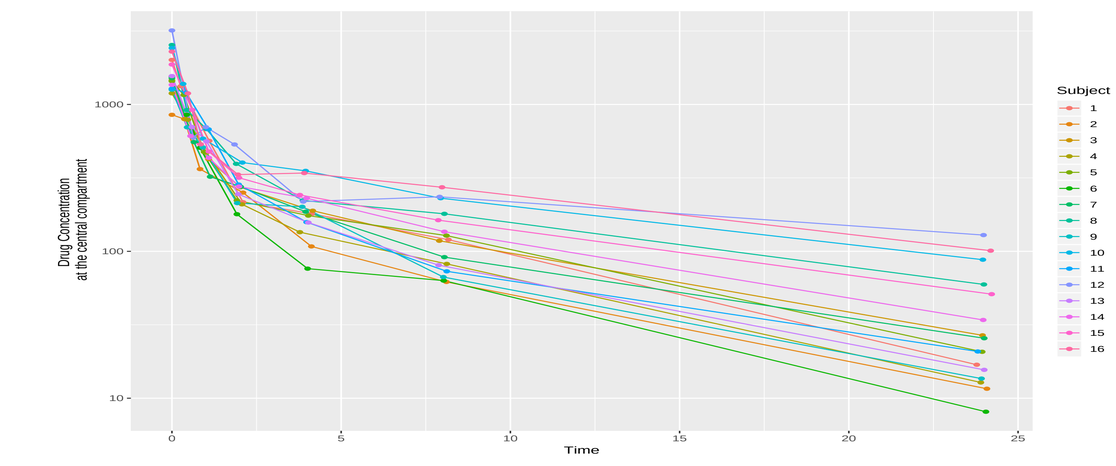
- Visually inspect the data and create the base model.
- Identify covariates through the stepwise covariate search.
- Bootstrapping analysis for the model selected by the covariate search procedure.

Note: R script and input dataset for this example can be found in C:\Program Files\R\R-n.n.n\library\RsNlme\

## Construct the base model

### Load the Input Dataset and Visually Inspect the Data

```
# load the input data set
dt_InputDataSet = fread("16subjects.csv")
```




### Define the Base Model

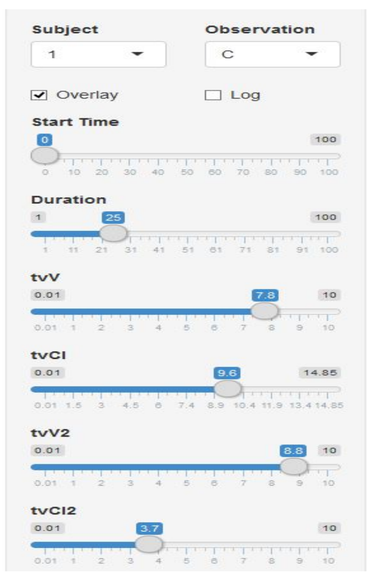
```
# define the basic PK model (a two-compartment model with IV bolus)
model = pkmodel(numComp = 2, modelName = "TwCpt_IVBolus_FOCE-ELS")

# reset residual error model (default: additive model with SD = 1)
residualEffect(model,"C") = c(errorType = Multiplicative, SD = "0.16")
```

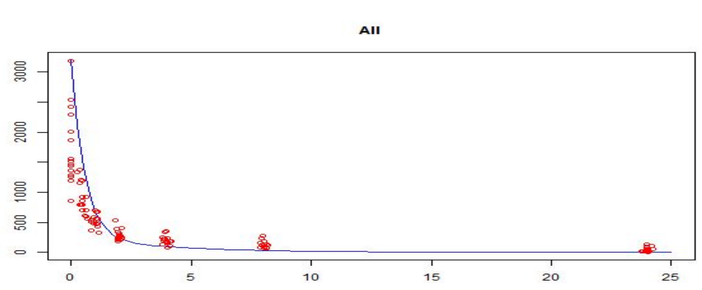
### Map Model Variables to Input Dataset Columns

```
# initialize model mapping and automatically mapping some of the model
  variables to the data columns
initColMapping(model) = dt_InputDataSet

# manually set up the mapping for the rest of variables
modelColumnMapping(model) = c(id = "Subject", CObs = "Conc", A1 = "Amount")
```

### Use the Initial Estimates Shiny App, estimatesUI, to Visually Determine a Set of Reasonable Initial Values for Fixed Effects

```
# host setup: run locally with MPI enabled
host = NlmeParallelHost(sharedDirectory = Sys.getenv("NLME_ROOT_DIRECTORY")
                        , parallelMethod = NlmeParallelMethod("LOCAL_MPI")
                        , hostName = "MPI"
                        , numCores = 4)

# invoke the Initial Estimates shiny app
estimatesUI(model, unique(dt_InputDataSet$Subject), host)
```

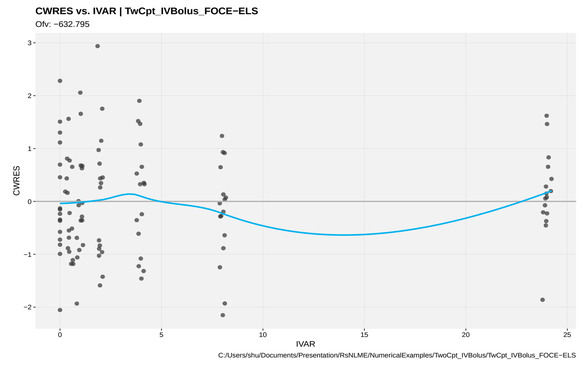


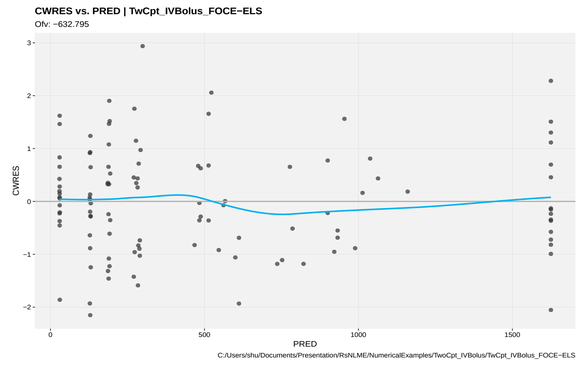




## ....

### Fit the Base Model with Initial Estimates Picked from Shiny App

```
 # accept initial estimates picked from the shiny app
 initFixedEffects(model) = getInitialEstimates()

# engine setup
engineParams = NlmeEngineExtraParams(PARAMS_METHOD = METHOD_FOCE_ELS,
                    PARAMS_NUM_ITERATIONS = 1000, PARAMS_SAND = "TRUE")
# fit the model
job = fitmodel(host, engineParams, model)

# imports results of an NLME run into xpose database to create commonly
# used diagnostic plots
 xp = xposeNlme(dir = model@modelInfo@workingDir
                , modelName = "TwCpt_IVBolus_FOCE-ELS")
```




## Identify covariates through the covariate search

### Add Covariates to the Base Model

```
# define covariates, sex, weight, and age
sex = categoricalCovariate("sex",c(1,2), c("female","male"))
weight = NlmeCovariateParameter("weight", centerValue = "70",
                        continuousType = CovarNumber, direction = Forward)
age = NlmeCovariateParameter("age")

# automatically incorporate the covariates into the basic model
covarModel = addCovariates(covarModel, c(sex,weight,age),
                                c("V" = "weight,age", "Cl" = "sex,weight"))
```

### Run the Stepwise Covariate Search

```
# set up for the stepwise covariate search
stepwiseSearchSetup = NlmeStepwiseParams(0.01, 0.001, "-2LL")

# run stepwise covariate search
job = stepwiseSearch(host, engineParams, covariateModel(covarModel),
              stepwiseSearchSetup, covarModel)
```

### Load and View Results

```
# load and view the model selected by the stepwise covariate search
stepwiseLines = readLines("Stepwise.txt")
```

## BootStrapping analysis for the selected model

### Reset the Covariates to the List Suggested by the Covariate Search

```
 # return a new model with all covariate effects cleared
 selectedCovarModel = resetCovariateEffects(covarModel)

 # enable the covariates selected by the covariate search
 covariateEffect(selectedCovarModel, "sex", "Cl") =  EnableEffect
 covariateEffect(selectedCovarModel, "age", "V") =  EnableEffect

 # update the PML statements
 selectedCovarModel = generatePMLModel(selectedCovarModel)
```

### Run the Bootstrap

```
 # Copy the selected model into a new object, and then create a new working
 # directory and copy model, input dataset, and column mapping files to it
 bootModel = copyModel(selectedCovarModel
       , modelName = "TwoCpt_IVBolus_SelectedCovarModel_Bootrapping"))

 # Bootstrap setup
 bootSetup = NlmeBootstrapParams(numReplicates = 10, randomNumSeed = 1234
                                 , stratifyColumns = "sex")

 # Run the bootstrapping for the model selected during the covariate search
 job = bootstrap(host, engineParams, bootSetup, bootModel)
```

### Load and View Results

```
# load and view the estimation results for all boostrap runs
dt_out = fread("out.csv")
```

## Conclusions

- RsNLME provides R command line access to the Phoenix NLME engine allowing pharmacometricians with little or no knowledge of Phoenix NLME to format and visualize data, build and analyze models, and post-process results.
- RsNLME also provides greater flexibility for advanced Phoenix NLME users to work seamlessly with other R packages within the R environment.