Case example 1

Teun Post, Richard Hooijmaijers

Introduction

This document shows the various ways of running and analysing the case model. The first section desribes the data then submitting of models is described and finally the analysis.

Data

For this case study the Theophylline dataset is used. It is well known, publicly available and included in the `nlmixr` package. The base Theophylline dataset consists of 12 subjects with each 12 concentration measurements between 0 and 24 hours.

Modeling

Directly in `nlmixr`

In case a model is submitted directly in nlmixr the following can be done:

1. Define your model (using the unified user interface)
2. Import, create and/or adapt the required data
3. Run the `nlmixr` function and if applicable specify model settings

An example for the case study is given below:

```
library(nlmixr)

# define the model
basemod <- 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)
  })
}

# run nlmixr with previously defined model (data is present in package!)
fit <- nlmixr(basemod,theo_sd,est="saem")
```

Using shinyMixR in interactive session

In case a model is submitted using shinyMixR, the model should be defined in a separate file. This file should contain the model in the same form as needed for running directly in nlmixr. However, additional meta data should be provided for settings, the model should be named `runx` (where x is an increasing number) and data used should be indicated. Models are submitted by default in a separate R session to prevent the current R session from "freezing".

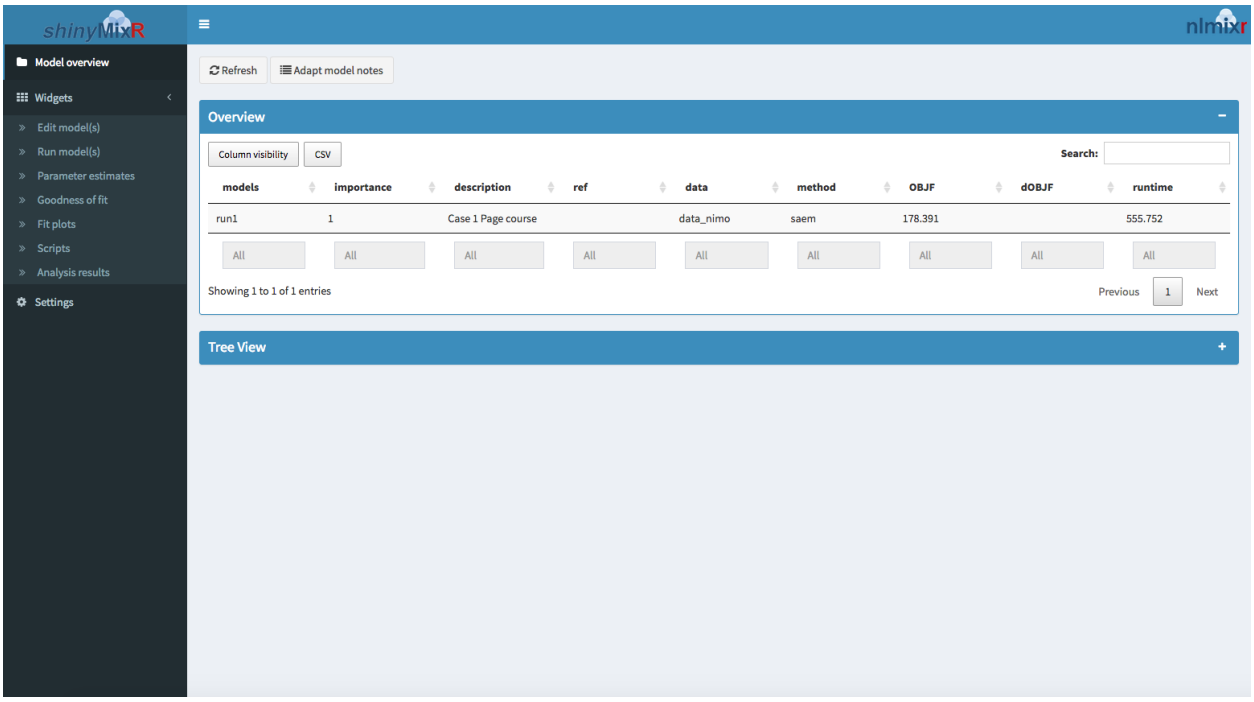
```
library(shinyMixR)

# Run a model
run_nmx("run1",proj)

# Progress can be read from external file
readLines("shinyMixR/temp/run1.prog.txt")
```

```
## -- Example of model with meta data --
## run1 <- function() {
##   data = "theo_sd"
##   desc = "base model"
##   ref  = ""
##   imp  = 1
##   est  = "nlme"
##   control<-list()
##   ini({
##     tka <- .5
##     tcl <- -3.2
## ...
```

Using shinyMixR interface

The interface can be opened and model(s) can be submitted in the run model widget (see example of app below). Within this widget also the progress can be assessed.

```
# -- Be aware that the same model is not already running interactively --
run_shinymixr(launch.browser = TRUE)
```



Analysing

Directly in `nlmixr`

High level results from an nlmixr run can be obtained by printing the fit object (within the latest R/Rstudio version important parts are colored in the concole):

```
library(nlmixr)
```

```
## Loading required package: nlme
```

```
## Loading required package: RxODE
```

```
fit
```

```
## -- nlmixr SAEM fit (Solved); OBJF calculated from FOCEi approximation ------------------------------
```

```
##      OBJF     AIC      BIC Log-likelihood Condition Number
##  116.102 130.102 150.2816        -58.051         120.3932
```

```
##
## -- Time (sec; x$time): -----------------------------------------------------------------------------
```

```
##           saem  setup Likelihood Calculation covariance table
## elapsed 15.716 11.308                  0.642          0 0.157
```

```
##
## -- Parameters (x$par.fixed): -----------------------------------------------------------------------
```

```
##         Estimate     SE %RSE Back-transformed(95%CI) BSV(CV%) Shrink(SD)%
## tka        0.450  0.194 43.0       1.57 (1.07, 2.29)    72.1%     -1.05%
## tcl        -3.22 0.0816 2.54 0.0401 (0.0342, 0.0471)    26.9%      4.76%
## tv        -0.784 0.0435 5.56    0.457 (0.419, 0.497)    13.6%      9.94%
## add.err    0.692                              0.692                11.9%
```

```
##
##   No correlations in between subject variability (BSV) matrix

##   Full BSV covariance (x$omega) or correlation (x$omega.R; diagonals=SDs)

##   Distribution stats (mean/skewness/kurtosis/p-value) available in x$shrink

##
## -- Fit Data (object x is a modified data.frame): ----------------------------------------------------
```

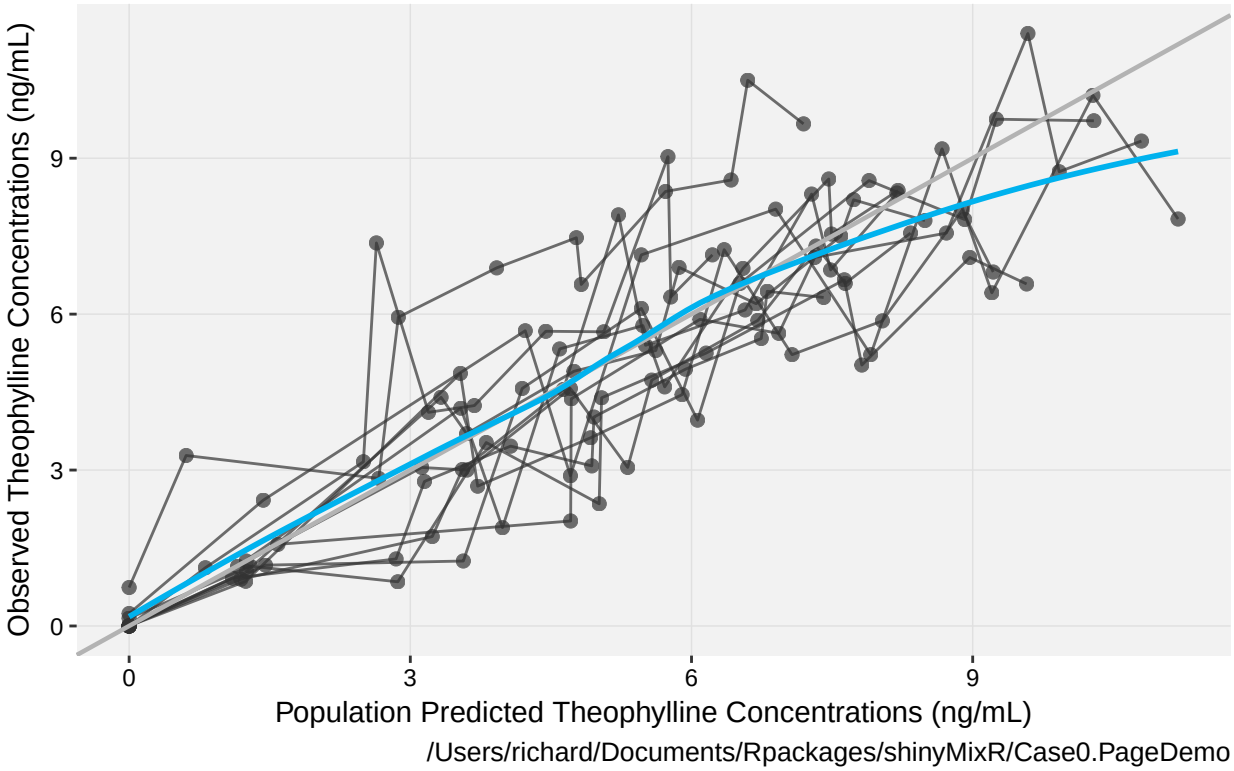
```
## # A tibble: 132 x 22
##   ID     TIME    DV  PRED    RES   WRES IPRED   IRES  IWRES CPRED  CRES
## * <fct> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 1     0.    0.740  0.   0.740  1.07    0.    0.740  1.07   0.   0.740
## 2 1     0.250 2.84   2.82 0.0178 0.0105  3.85 -1.01  -1.46   2.66 0.177
## 3 1     0.570 6.57   5.06 1.51   0.680   6.76 -0.191 -0.276  4.82 1.75
## # ... with 129 more rows, and 11 more variables: CWRES <dbl>,
## #   eta.ka <dbl>, eta.cl <dbl>, eta.v <dbl>, rx0 <dbl>, rx1 <dbl>,
## #   ka <dbl>, cl <dbl>, v <dbl>, rx1c <dbl>, Central <dbl>
```

For plotting, it is advised to use the `xpose.nlmixr` package. Results can be easily exported using, e.g. using the `pdf()` or `png()` functions. Examples of some default plots are given below:

```
library(xpose.nlmixr)
xpdb <- xpose_data_nlmixr(fit)

dv_vs_pred(xpdb) +
  ylab("Observed Theophylline Concentrations (ng/mL)") +
  xlab("Population Predicted Theophylline Concentrations (ng/mL)")
dv_vs_ipred(xpdb) +
  ylab("Observed Theophylline Concentrations (ug/mL)") +
  xlab("Individual Predicted Theophylline Concentrations (ng/mL)")
res_vs_pred(xpdb) +
  ylab("Conditional Weighted Residuals") +
  xlab("Population Predicted Theophylline Concentrations (ng/mL)")
res_vs_idv(xpdb) +
  ylab("Conditional Weighted Residuals") +
  xlab("Time (h)")
prm_vs_iteration(xpdb)
absval_res_vs_idv(xpdb, res = 'IWRES') +
  ylab("Individual Weighted Residuals") +
  xlab("Time (h)")
absval_res_vs_pred(xpdb, res = 'IWRES')  +
  ylab("Individual Weighted Residuals") +
  xlab("Population Predicted Theophylline Concentrations (ng/mL)")
ind_plots(xpdb, nrow=3, ncol=4) +
  ylab("Predicted and Observed Theophylline concentrations (ng/mL)") +
  xlab("Time (h)")
res_distrib(xpdb) +
  ylab("Density") +
  xlab("Conditional Weighted Residuals")
nlmixr::vpc(fit,nsim=500, show=list(obs_dv=T),
            ylab = "Theophylline Concentrations (ng/mL)", xlab = "Time (h)")
```

DV vs. CPRED | run2

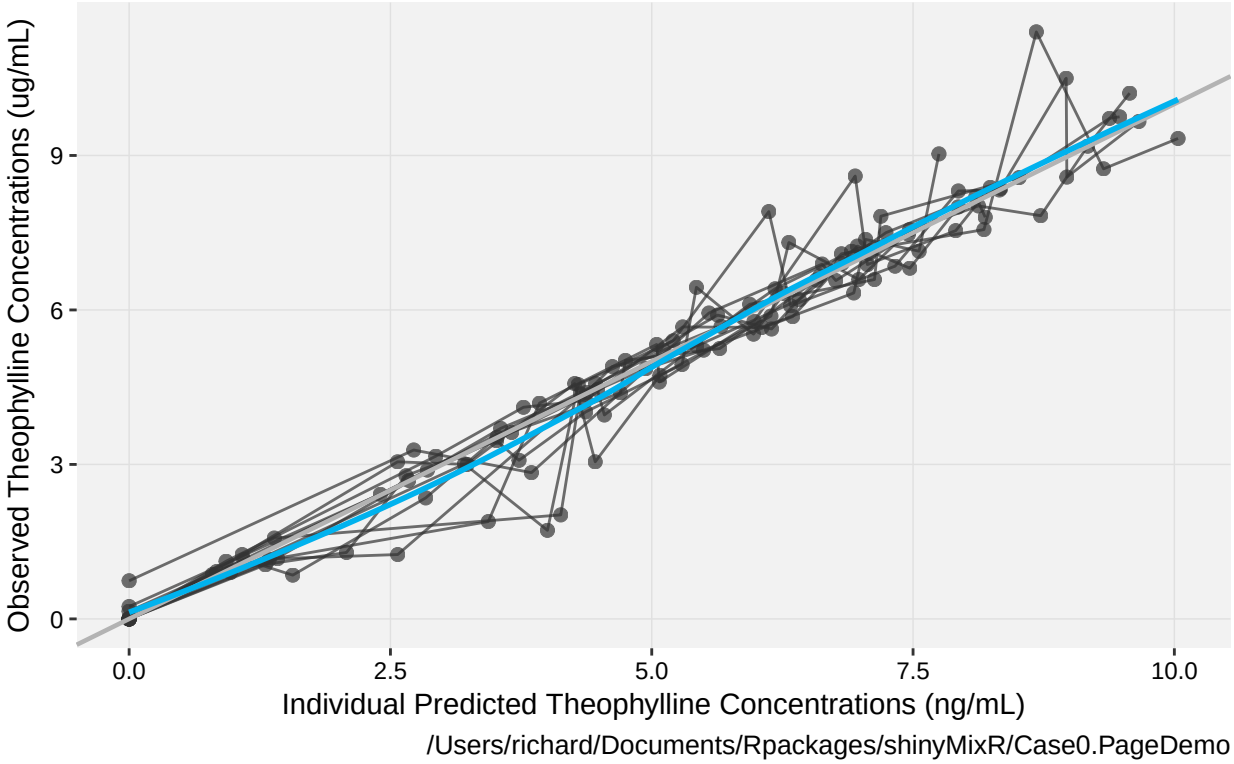
Ofv: 116.1



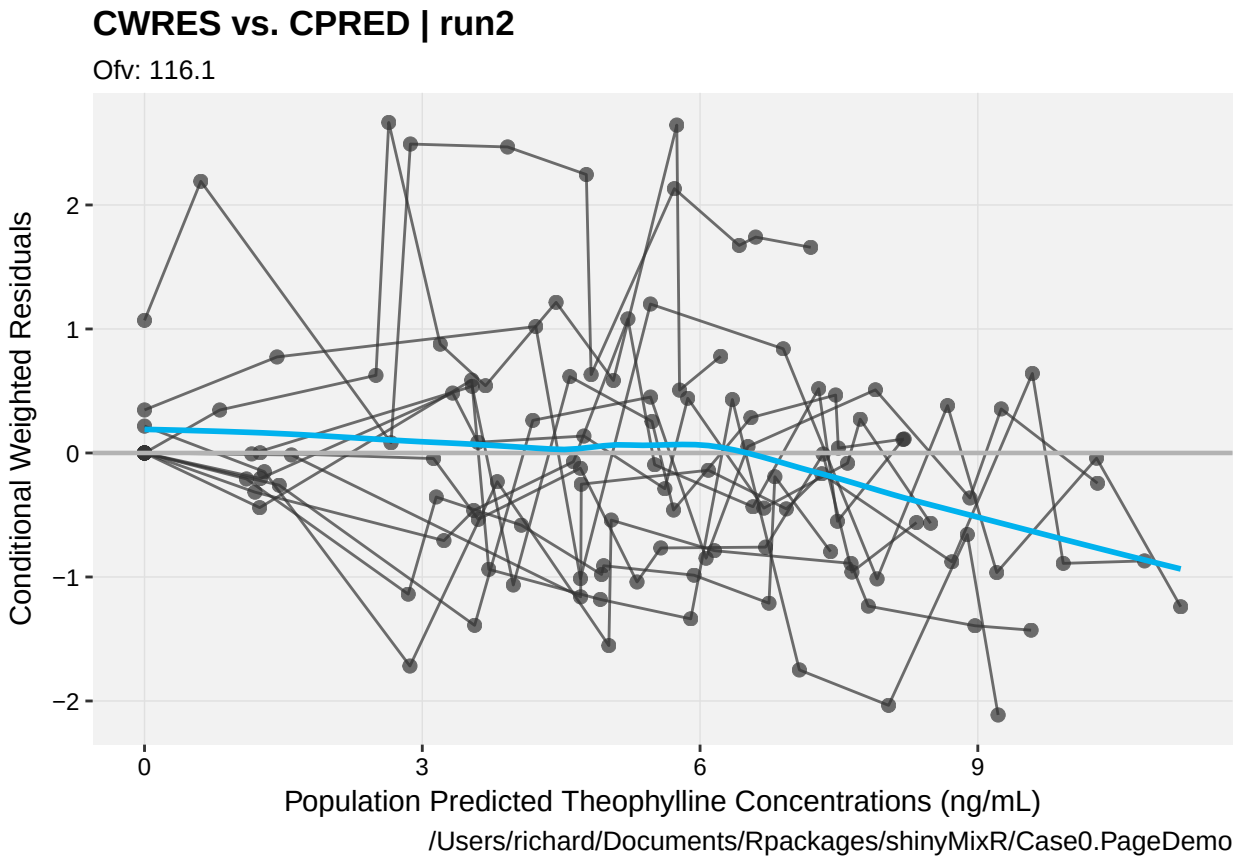
/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo

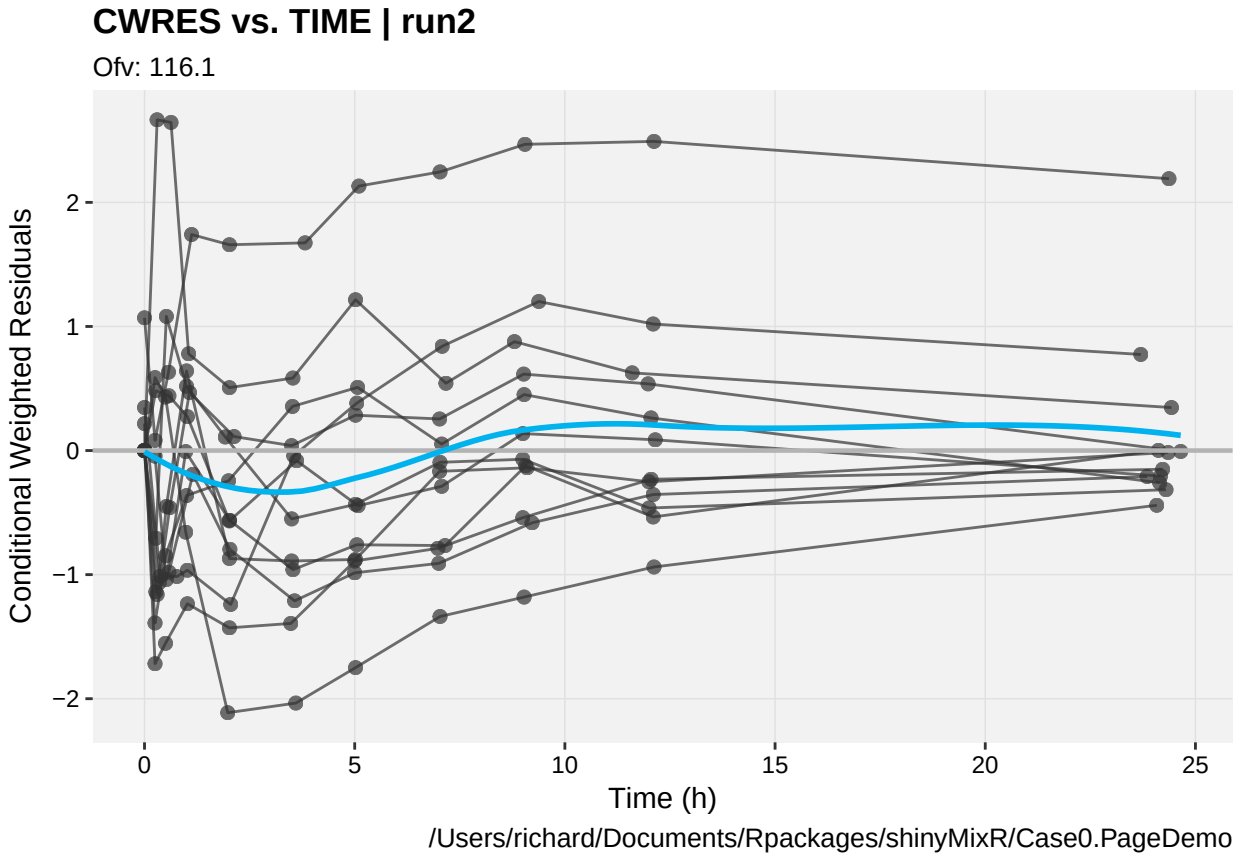
DV vs. IPRED | run2

Ofv: 116.1, Eps shrink: 11.9 [1]



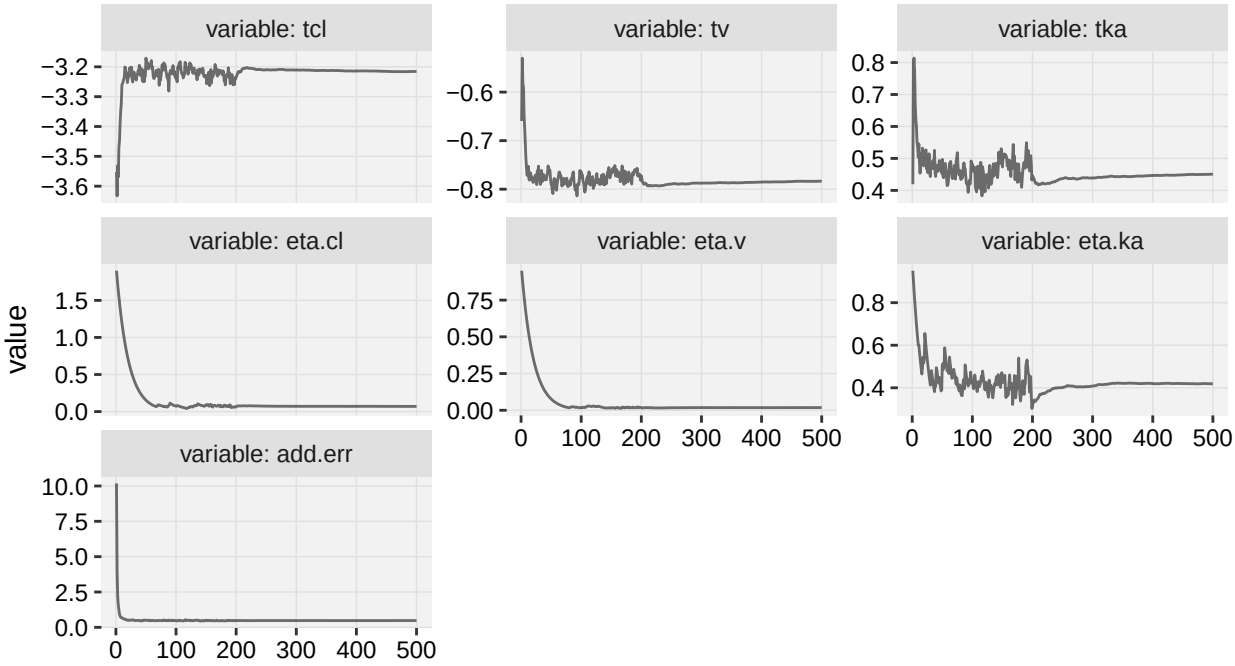
/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo


CWRES vs. CPRED | run2
Ofv: 116.1
Conditional Weighted Residuals
2
1
0
−1
−2
0
3
6
9
Population Predicted Theophylline Concentrations (ng/mL)
/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo


CWRES vs. TIME | run2
Ofv: 116.1
Conditional Weighted Residuals
2
1
0
−1
−2
0
5
10
15
20
25
Time (h)
/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo

Parameter value vs. ITERATION | run2

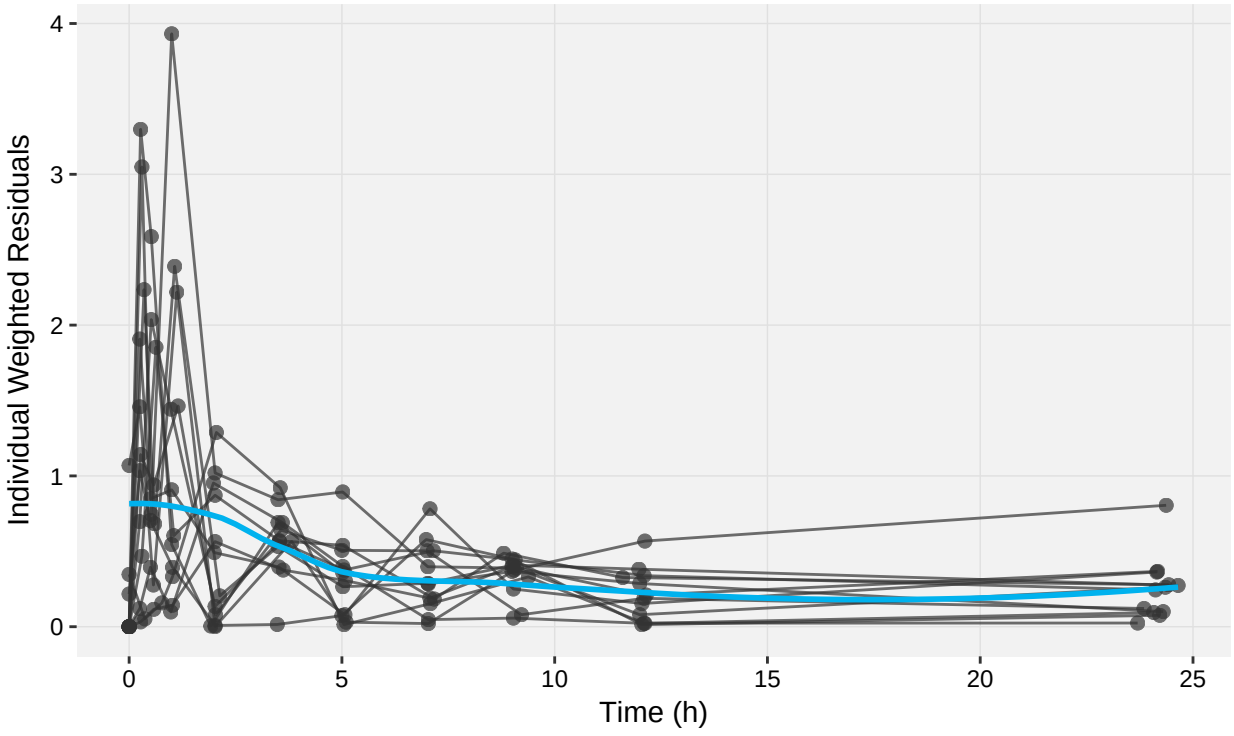
Method: saem, minimization time: 15.7
Termination message: na



/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo

abs(IWRES) vs. TIME | run2

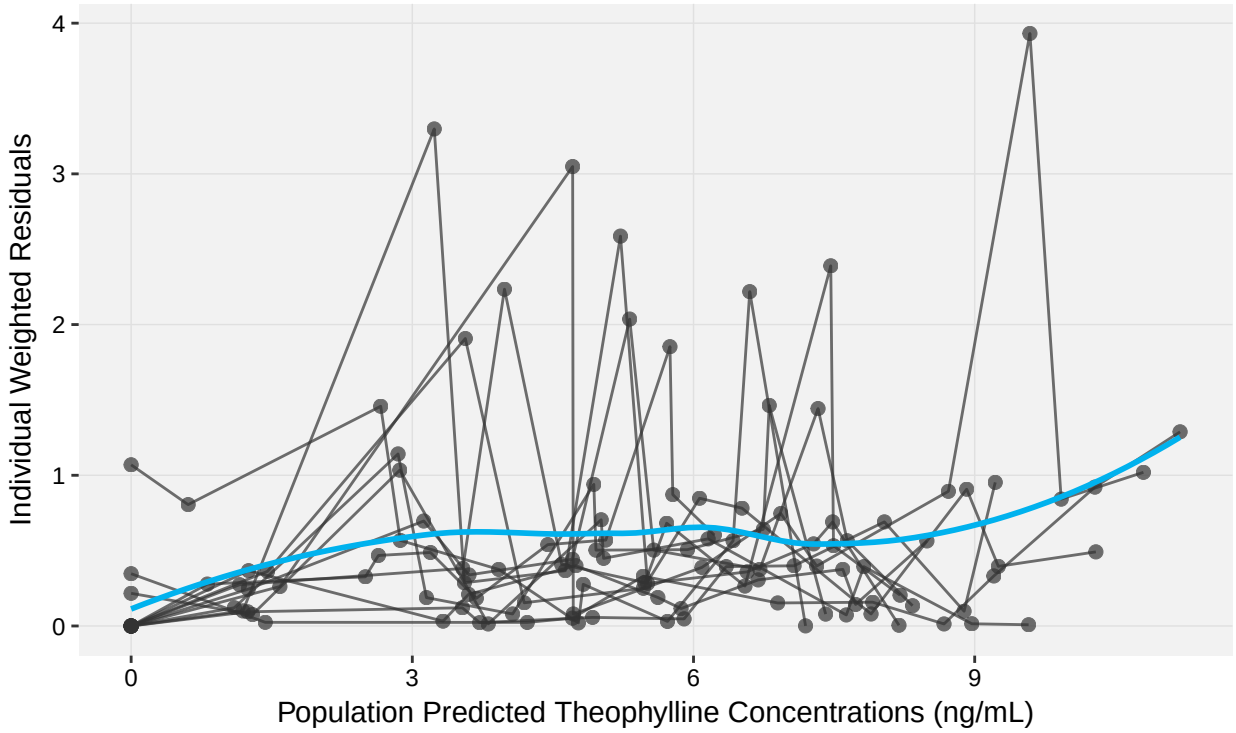
Ofv: 116.1



/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo

abs(IWRES) vs. CPRED | run2

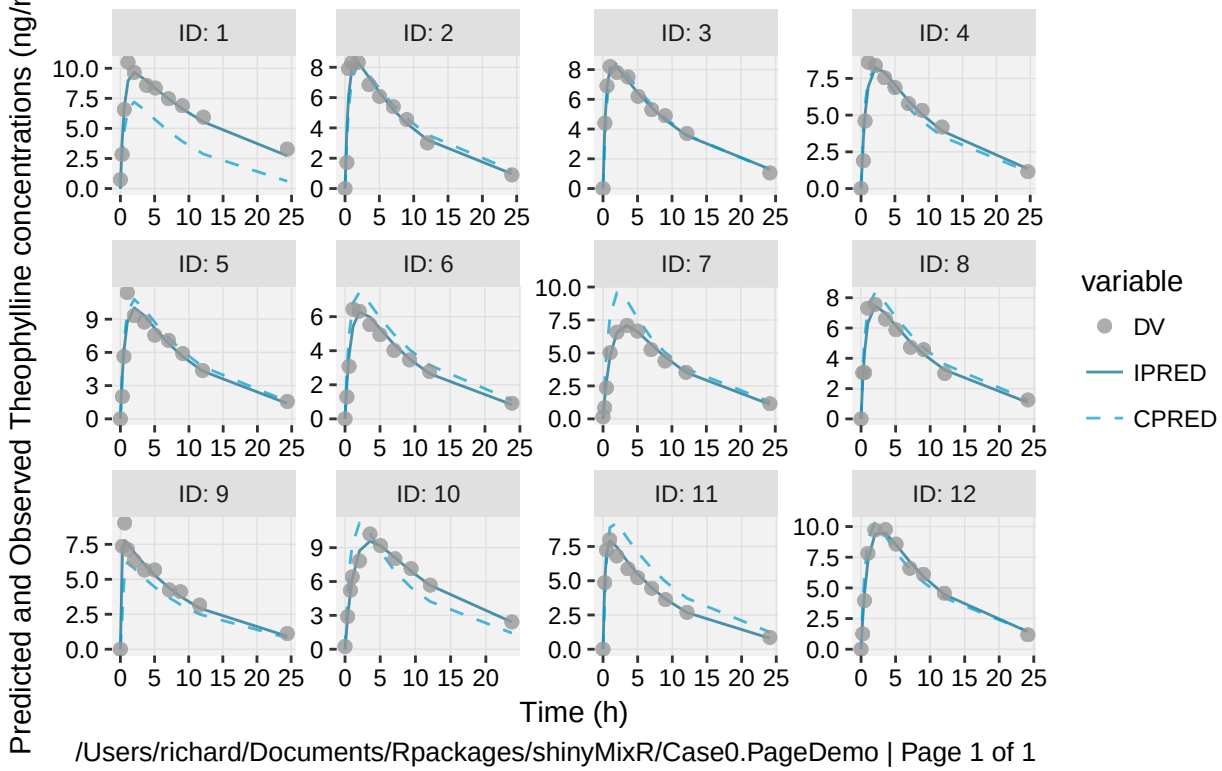
Ofv: 116.1



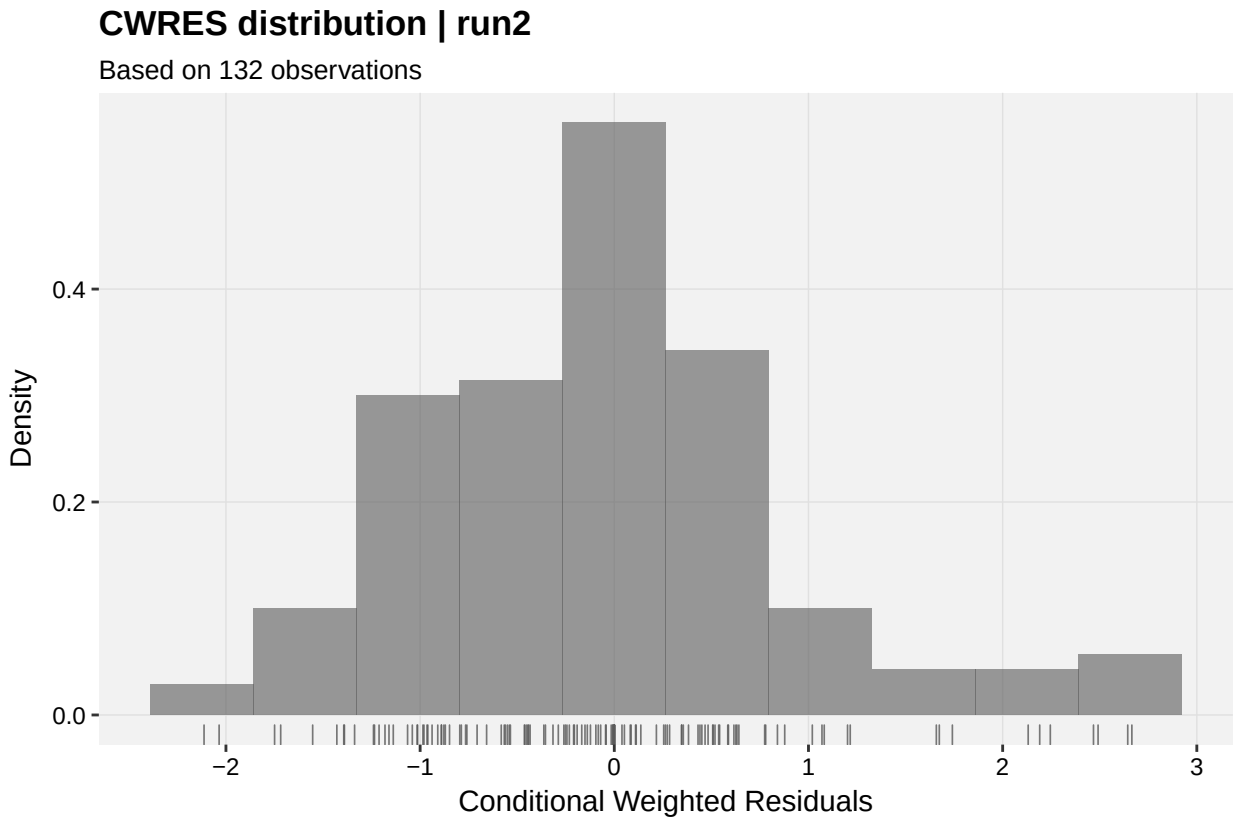
/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo

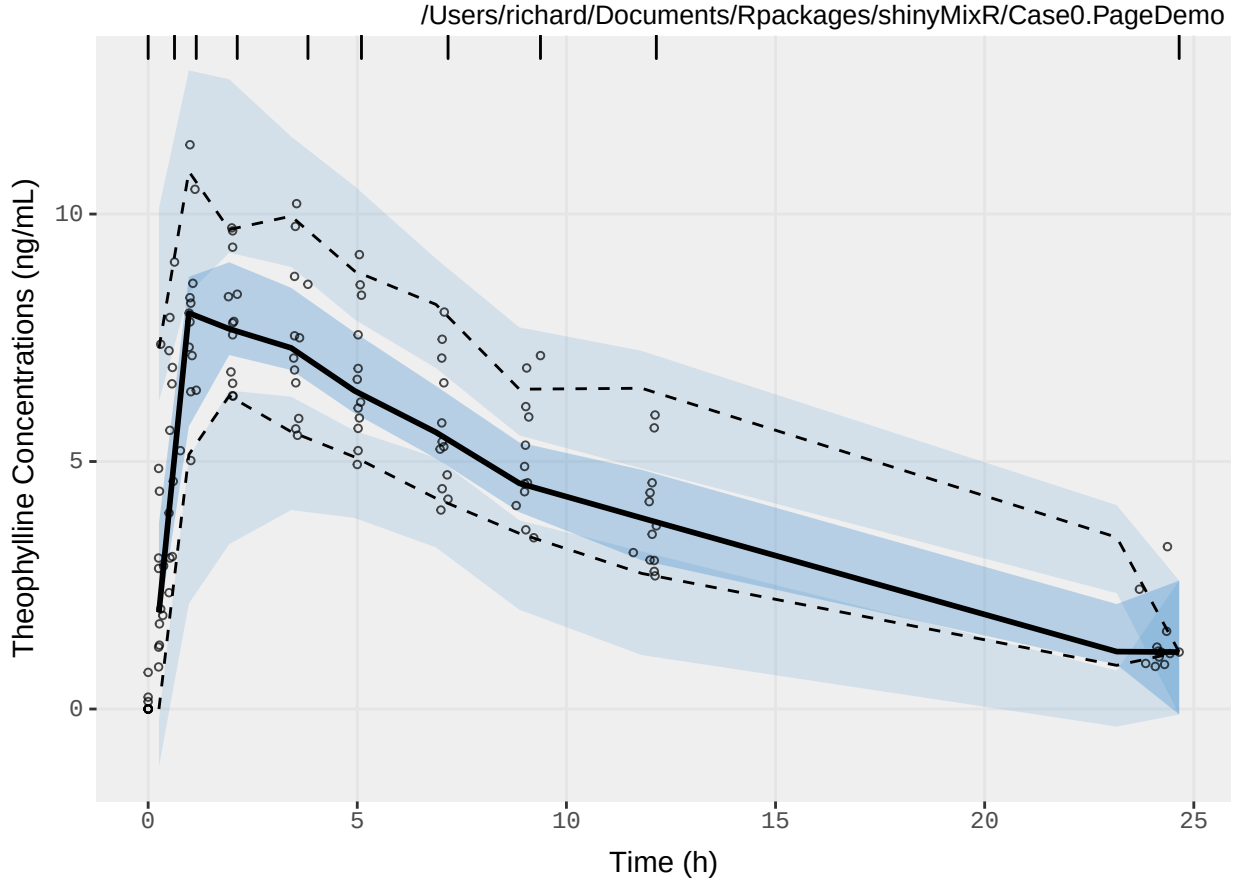
Individual plots | run2

Ofv: 116.1, Eps shrink: 11.9 [1]



/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo | Page 1 of 1


CWRES distribution | run2
Based on 132 observations
Density
0.4
0.2
0.0
−2
−1
0
1
2
3
Conditional Weighted Residuals


/Users/richard/Documents/Rpackages/shinyMixR/Case0.PageDemo
Theophylline Concentrations (ng/mL)
10
5
0
0
5
10
15
20
25
Time (h)

Using shinyMixR in interactive session

For shinyMixR there are a couple of functions present in the package to create plots and the xpose.nlmixr functions can also be used

```
gof_plot(fit)              # combine multiple GOF at once
prm_vs_iteration(xpdb)     # xpose.nlmixr functions can be used directly
fit_plot(fit,type="user") # "default ggplot" output can be created
```

Some results can be exported by the function directly. Other output can be xported using the R3port package or with the pdf() function e.g.

```
gof_plot(res,outnm="gofplot.tex",mdlnm="run1")
pl <- prm_vs_iteration(xpdb2); R3port::ltx_plot(pl,out="plot.tex")
pdf("results.pdf"); fit_plot(res); dev.off()
```

Using shinyMixR interface

There are widgets for a few default plots, user scripts are possible in the script widget plots are by default saved in the analyis folder and made available in the interface The post processing scripts can also be used in an interactive session. In these cases the model should be provided, e.g.

```
models <- "run1"
source("./scripts/postprocessing script.r")
```

When creating the plots in the interface, reports can be generated in pdf or html format: