shinyMixR: A project-centric R/Shiny run management tool for nlmixr

Richard Hooijmaijers[1,5], Matt Fidler[3,5], Rik Schoemaker[2,5], Mirjam Trame[3,5], Wenping Wang[3,5], Justin Wilkins[2,5], Yuan Xiong[4,5], Teun Post[1,5]

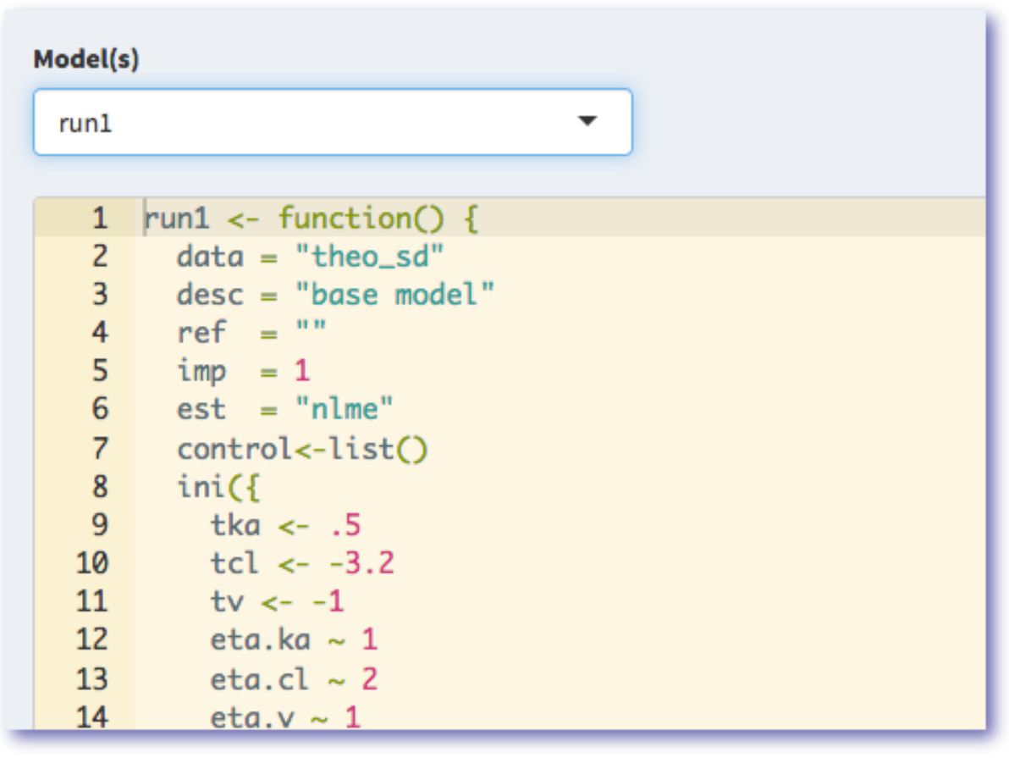
[1] LAP&P Consultants, The Netherlands, [2] Occams, The Netherlands, [3] Novartis Pharmaceuticals, USA, [4] Certara Strategic Consulting, USA, [5] The nlmixr team

Introduction

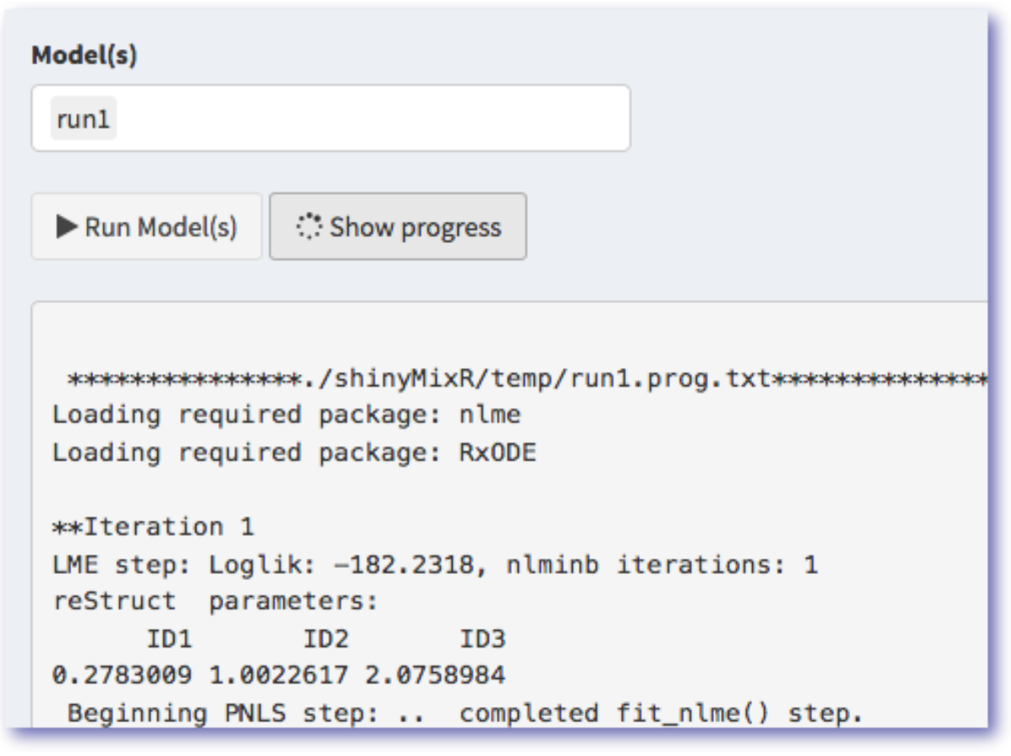
The combination of open-source packages nlmixr and RxODE, available on CRAN[1,2] and actively developed on GitHub[1,2], provides a non-linear mixed effects system to perform population-type pharmacokinetic and pharmacodynamic analyses and simulations[3] in R[4]. The ability to perform population modeling in R provides an opportunity to work via a single unified workflow. The aim of this current work was to develop a user-friendly tool for nlmixr based on shiny, which would facilitate a workflow around an nlmixr project. Ultimately, this should allow for:

1) dynamic and interactive model development
2) quick and efficient communication of population PK-PD models
3) rapid demonstration of simulation results (also see RxODE Shiny)
4) reporting of modelling results[5].

ShinyMixR[6] is set up as an open source nlmixr project management tool written completely in R, and deployed as an R package. The shinyMixR system is built around a project-centric structure and provides an interface to nlmixr from both the R command line (R, related GUIs and RStudio[7]) as well as a user-friendly Shiny dashboard application[8]. The 'shinydashboard' package[9] provides a layer on top of shiny to produce an easy-to-use dashboard which can be used for controlling and tracking runs with an nlmixr project, and was the basis for setting up the modular interface. Most of the functions underlying the interface are written such that these can be called independently from the R command line, and also work in combination with the graphical interface *and vice versa*.

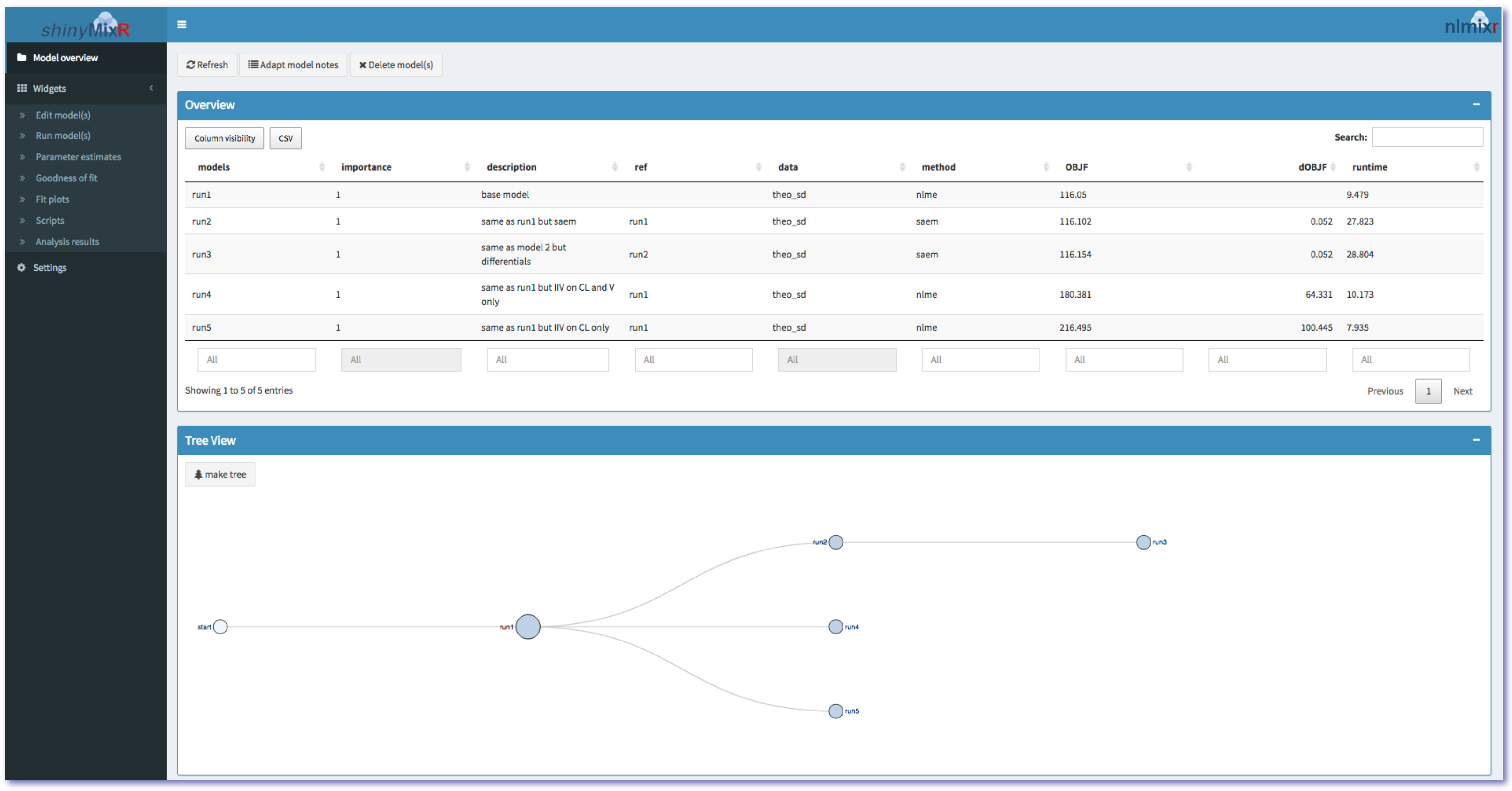


```
run1 <- function() {
  data = "theo_sd"
  desc = "base model"
  ref  = ""
  imp  = 1
  est  = "nlme"
  control<-list()
  ini({
    tka <- .5
    tcl <- -3.2
    tv <- -1
    eta.ka ~ 1
    eta.cl ~ 2
    eta.v ~ 1
```



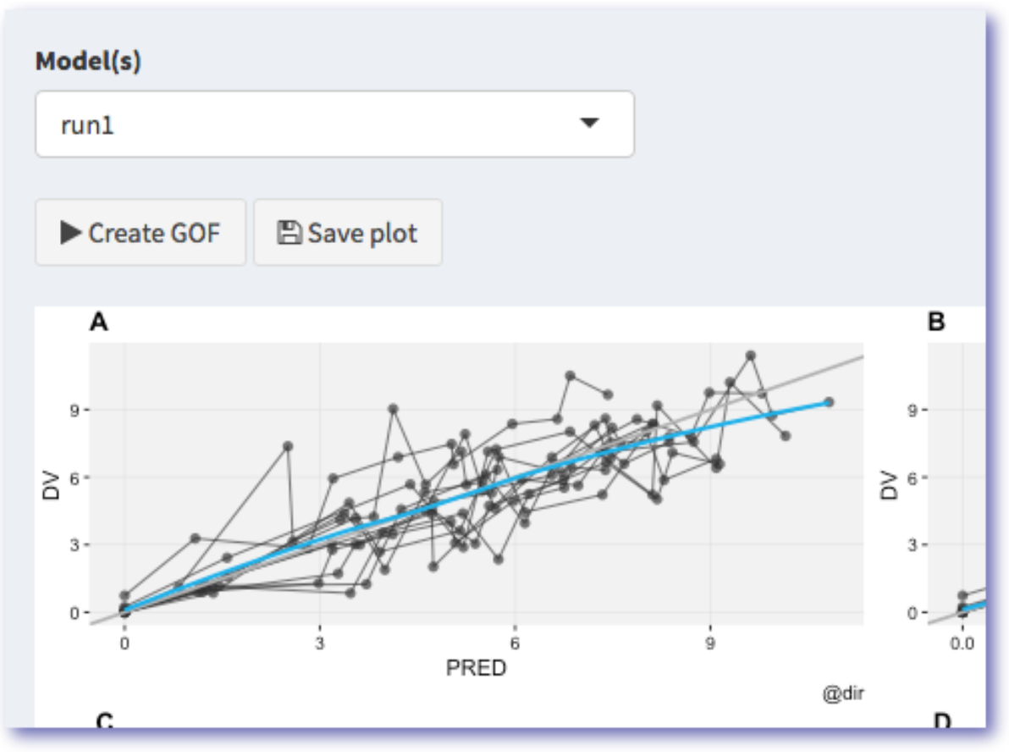
```
 ***************./shinyMixR/temp/run1.prog.txt**************
Loading required package: nlme
Loading required package: RxODE

**Iteration 1
LME step: Loglik: -182.2318, nlminb iterations: 1
reStruct  parameters:
      ID1       ID2       ID3
0.2783009 1.0022617 2.0758984
 Beginning PNLS step: ..  completed fit_nlme() step.
```

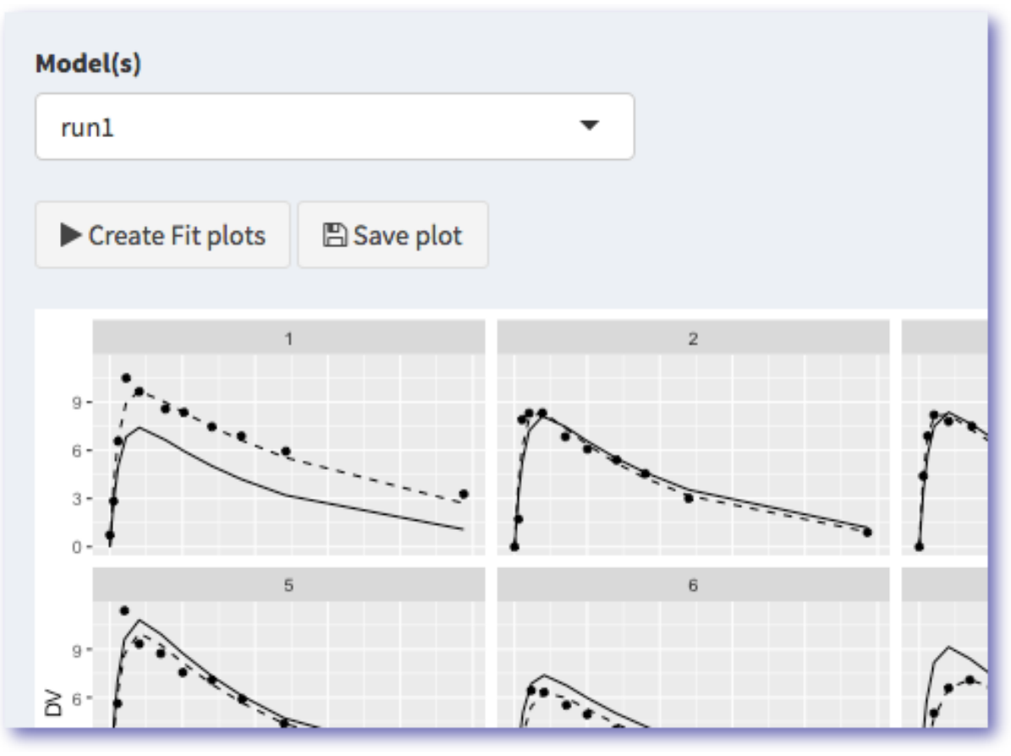
- Customize, create tree and export the model overview.
- Edit models using syntax highlighting, customize editor, use template models, duplicate and save models.
- Run models in external R session, run multiple models side-by-side, keep track of progress.
- Combine separate results in one report, use LaTeX or HTML to make beautiful and easy to navigate reports.



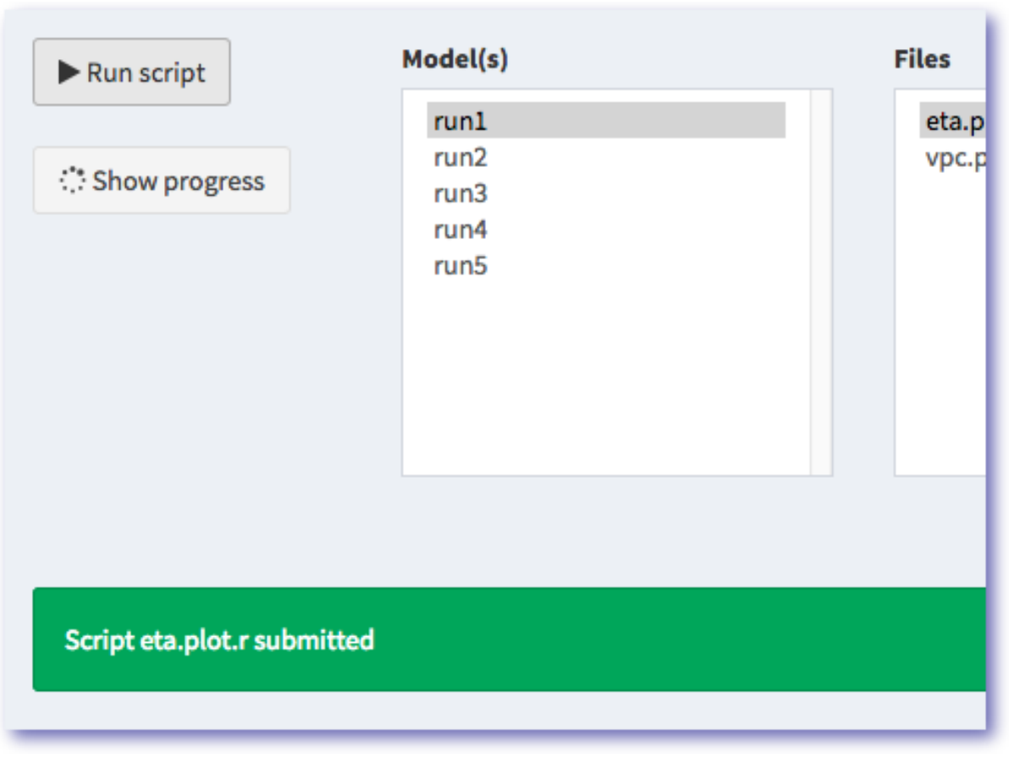
models	importance	description	ref	data	method	OBJF	dOBJF	runtime
run1	1	base model		theo_sd	nlme	116.05		9.479
run2	1	same as run1 but saem	run1	theo_sd	saem	116.102	0.052	27.823
run3	1	same as model 2 but differentials	run2	theo_sd	saem	116.154	0.052	28.804
run4	1	same as run1 but IIV on CL and V only	run1	theo_sd	nlme	180.381	64.331	10.173
run5	1	same as run1 but IIV on CL only	run1	theo_sd	nlme	216.495	100.445	7.935
All	All	All	All	All	All	All	All	All

Showing 1 to 5 of 5 entries

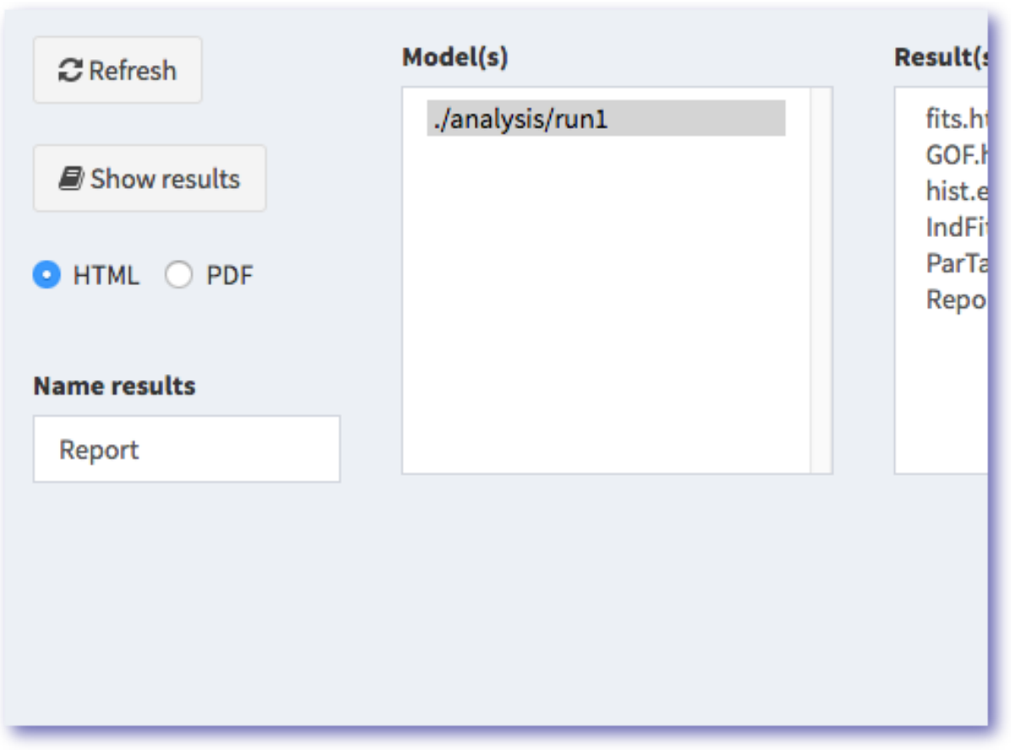


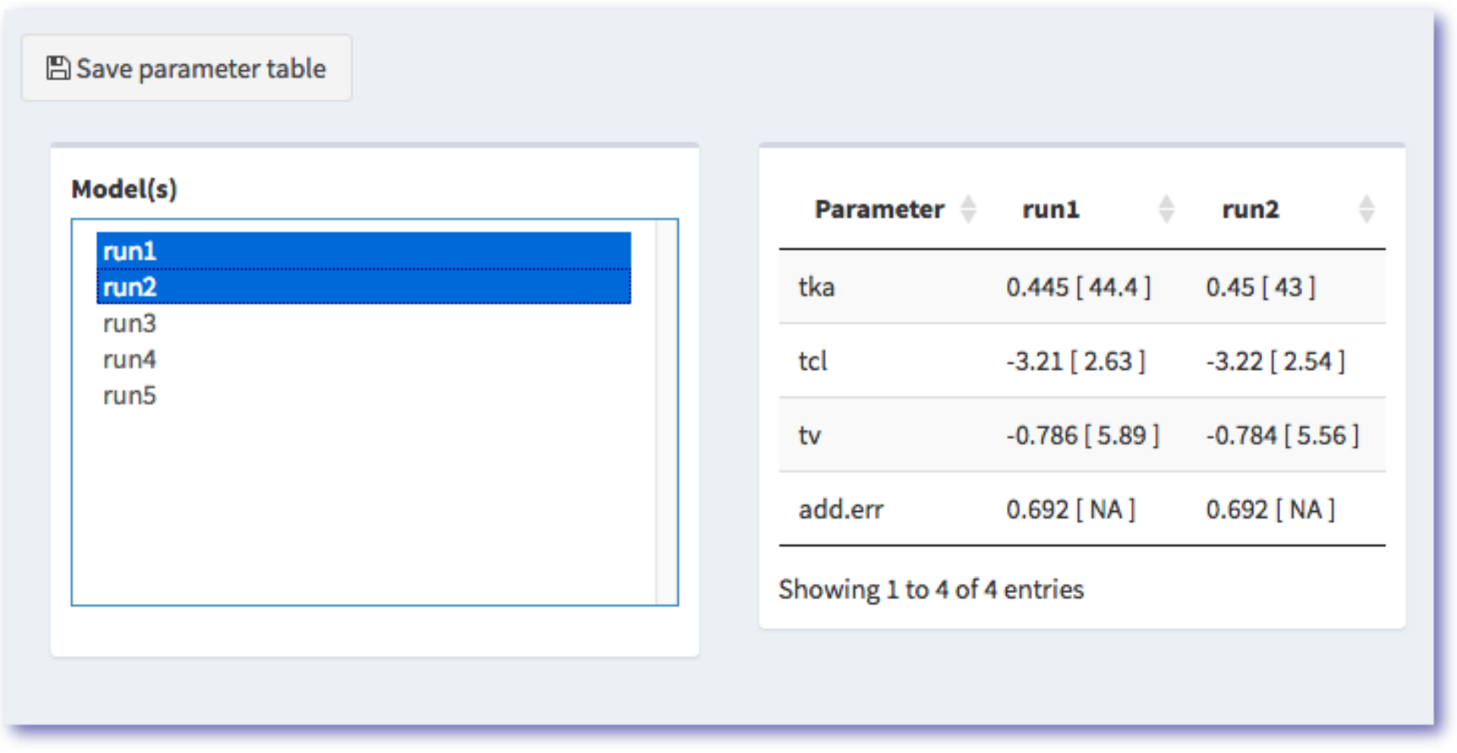


- Create most important plots using xpose.nlmixr[10] or ggplot package and save results with R3port package.
- Quickly see and compare parameter estimates, save results that can directly be used in LaTeX or powerpoint.
- Create custom R scripts to perform project specific analyses, directly apply scripts on multiple model runs.



Script eta.plot.r submitted





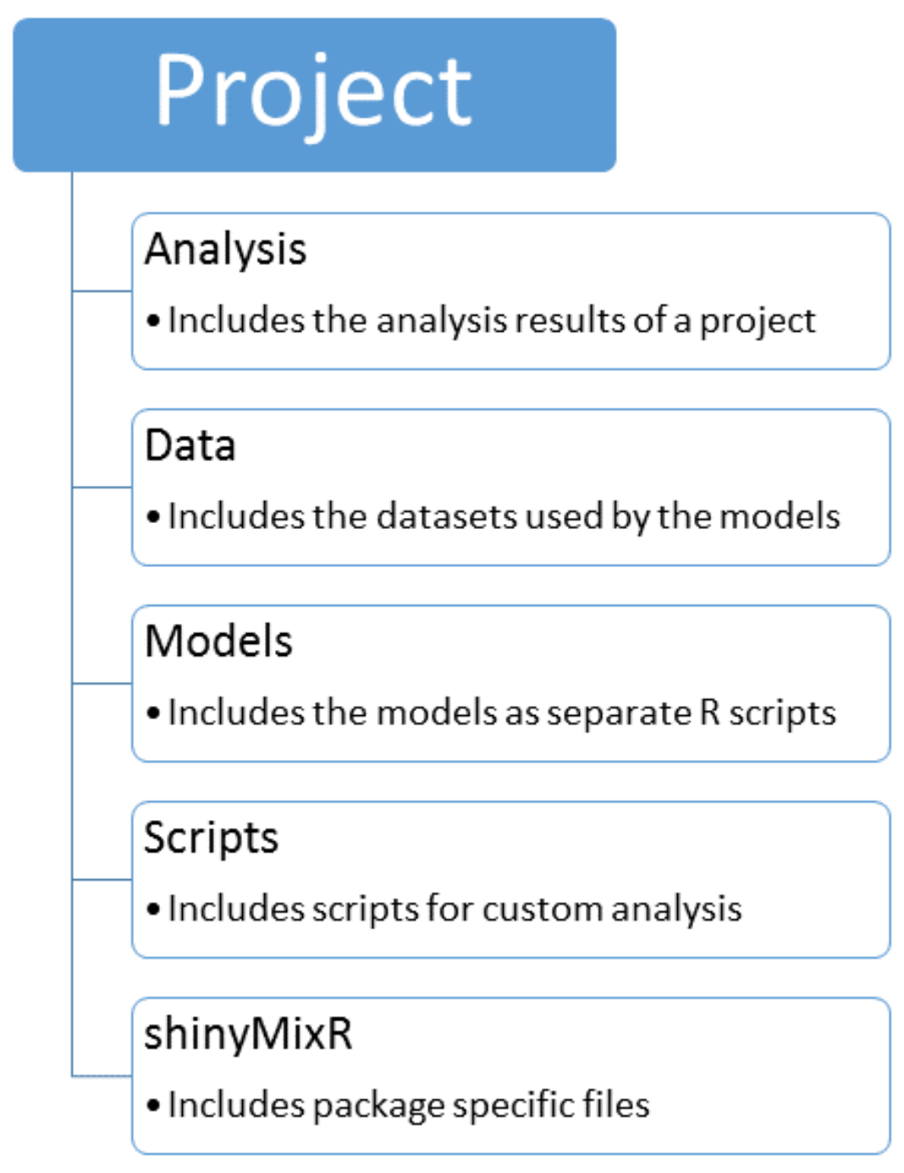
Parameter	run1	run2
tka	0.445 [44.4]	0.45 [43]
tcl	-3.21 [2.63]	-3.22 [2.54]
tv	-0.786 [5.89]	-0.784 [5.56]
add.err	0.692 [NA]	0.692 [NA]

Showing 1 to 4 of 4 entries

Usage

General

To be able to work with the package a specific folder structure for a project is required. This structure can be created using the `create_proj` function and will create a set of folders and files:



Project

Analysis
- Includes the analysis results of a project

Data
- Includes the datasets used by the models

Models
- Includes the models as separate R scripts

Scripts
- Includes scripts for custom analysis

shinyMixR
- Includes package specific files

- This structure is used by the package to manage models and (graphical) results, and maintained in a project object.
- The structure is monitored by the package to identify changes in order to provide up-to-date information.
- The structure should be created once at the start of a project.
- The function will include sample files to create a starting point for a project.
- The package can handle files created/deleted outside the package if naming conventions are followed.

Interactive usage

Most important functions for interactive usage trough the command line:

`create_proj()`	Create a folder structure for a shinyMixR project
`run_nmx()`	Run a nlmixr model, possibly in a separate R session to overcome "freezing" of current session
`overview()`	Create overview of all models in a project
`tree_overview()`	Create a collapsible tree overview for visualizing relationship between models
`par_table()`	Create dense parameter table for one or multiple models
`gof_plot()`	Create a combination of most important goodness of fit plots
`fit_plot()`	Create individual fit plots
`get_proj()`	Get project information with available models and high level results

Interface usage

The interface can be started from the projects root folder using `run_shinymixr()`

The app can be opened in an Rstudio window or web browser. The start window displays a dashboard with in the main window a (tree) overview of the models in the project structure. The interface can be started at all times – even if the project was initially started in an interactive way; and vice versa.

Conclusions

- The ShinyMixR package provides a means to build a project-centric workflow around nlmixr from the R command line and from a streamlined Shiny application.
- This project tool was developed to enhance the usability and attractiveness of nlmixr, facilitating dynamic and interactive use in real-time for rapid model development.

References

[1] CRAN: https://cran.r-project.org/web/packages/nlmixr/index.html and GitHub: https://github.com/nlmixrdevelopment/nlmixr

[2] CRAN: https://cran.r-project.org/web/packages/RxODE/index.html and GitHub: https://github.com/nlmixrdevelopment/RxODE

[3] Wang W et al. CPT:PSP (2016) 5, 3–10.

[4] R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

[5] https://cran.r-project.org/web/packages/R3port/index.html

[6] https://github.com/RichardHooijmaijers/shinyMixR

[7] RStudio Team (2015). RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/

[8] http://shiny.rstudio.com/

[9] https://cran.r-project.org/web/packages/shinydashboard/shinydashboard.pdf

[10] https://github.com/nlmixrdevelopment/xpose.nlmixr

nlmixr

shinyMixR