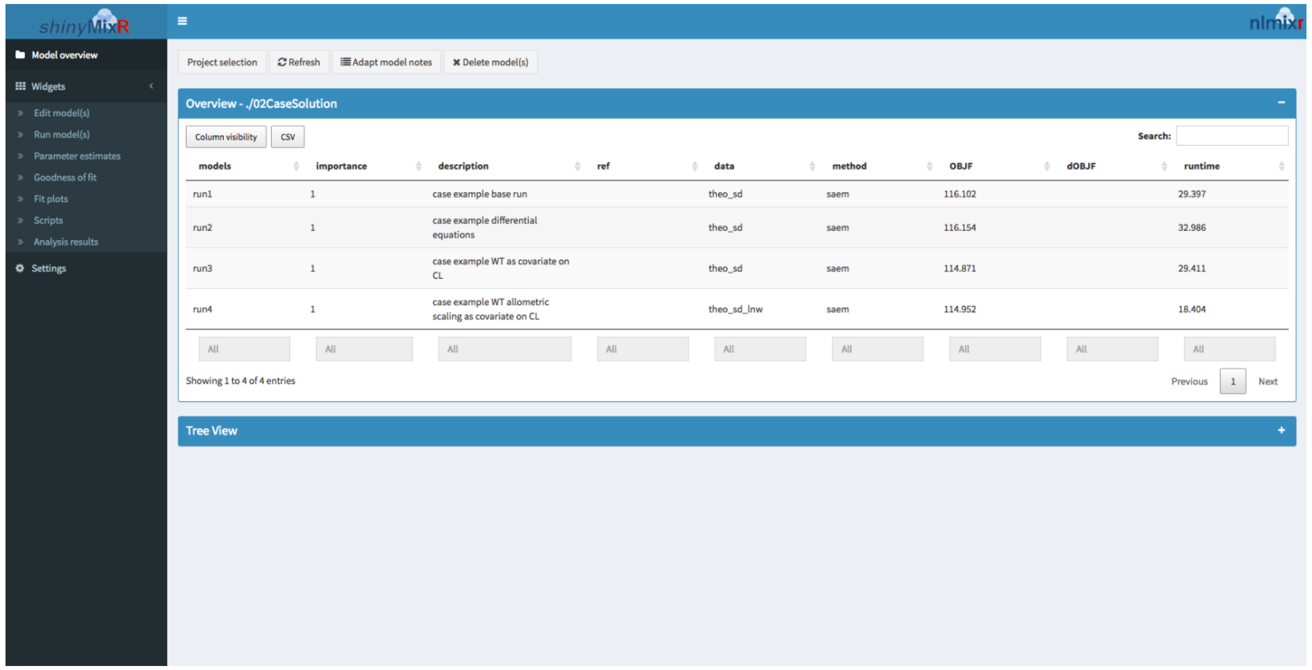
A shiny GUI for nlmixr: shinyMixR



Overview - ./02CaseSolution

models	importance	description	ref	data	method	OBJF	dOBJF	runtime
run1	1	case example base run		theo_sd	saem	116.102		29.397
run2	1	case example differential equations		theo_sd	saem	116.154		32.986
run3	1	case example WT as covariate on CL		theo_sd	saem	114.871		29.411
run4	1	case example WT allometric scaling as covariate on CL		theo_sd_lnw	saem	114.952		18.404
All	All	All	All	All	All	All	All	All

Showing 1 to 4 of 4 entries

Stockholm, Sweden, June 11th 2019

Mirjam N. Trame

On behalf of the nlmixr development team:

Matt Fidler, Richard Hooijmaijers, Teun Post, Rik Schoemaker, Mirjam Trame, Justin Wilkins, Yuan Xiong and Wenping Wang

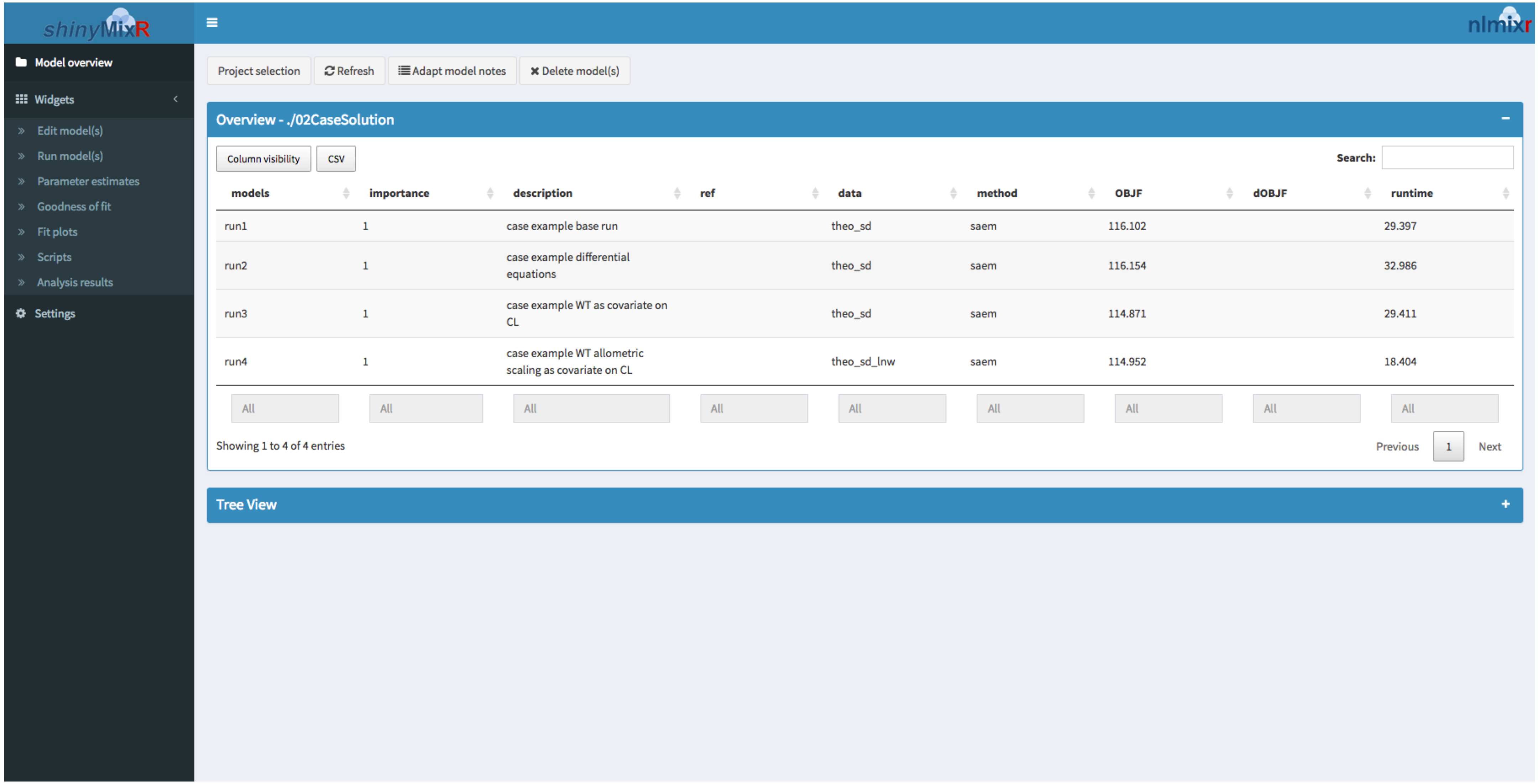
nlmixr and shinyMixR - background

There are two main ways of working with nlmixr models:

- Via the `nlmixr` package
  - `nlmixr` is the engine for running models
  - Provides output in a model fit object, which can be read out and approached
  - R cannot be used while running models – new R session can be opened
  - Model fit object is in the global environment

- Via the `shinyMixR` package
  - Provides a **graphical user interface** around `nlmixr`
  - Structures a project
  - Models are submitted in separate sessions – R can be used while running models
  - Model fit object is automatically saved to disk
  - Modeling output and models run via `nlmixr` cannot be imported
  - *Note*: several `shinyMixR` package functions can also be used interactively on command line

shinyMixR https://richardhooijmaijers.github.io/shinyMixR/index.html



Overview - ./02CaseSolution

models	importance	description	ref	data	method	OBJF	dOBJF	runtime
run1	1	case example base run		theo_sd	saem	116.102		29.397
run2	1	case example differential equations		theo_sd	saem	116.154		32.986
run3	1	case example WT as covariate on CL		theo_sd	saem	114.871		29.411
run4	1	case example WT allometric scaling as covariate on CL		theo_sd_lnw	saem	114.952		18.404
All	All	All	All	All	All	All	All	All

Showing 1 to 4 of 4 entries

Introduction

- The `shinyMixR` package is a **graphical user interface (GUI)** tool for managing pop PKPD projects with `nlmixr` as the estimation engine
- The package is intended to **view, edit, run, compare, analyze and report** nlmixr models
- It organizes your project – model, data, metadata, settings and results are kept together
- The interface enables browsing between specific project folders
- The application can be started via a shortcut or via the R command line in the project folder
- The interface is created using the `shiny` and `shinydashboard` packages
- *Note*: most functions within the package can also be used in an interactive R session
- The package is open source and is available at: https://richardhooijmaijers.github.io/shinyMixR/index.html

Project Structure

The **folder structure of a shinyMixR project is fixed** and should be followed to enable to work with the package*:

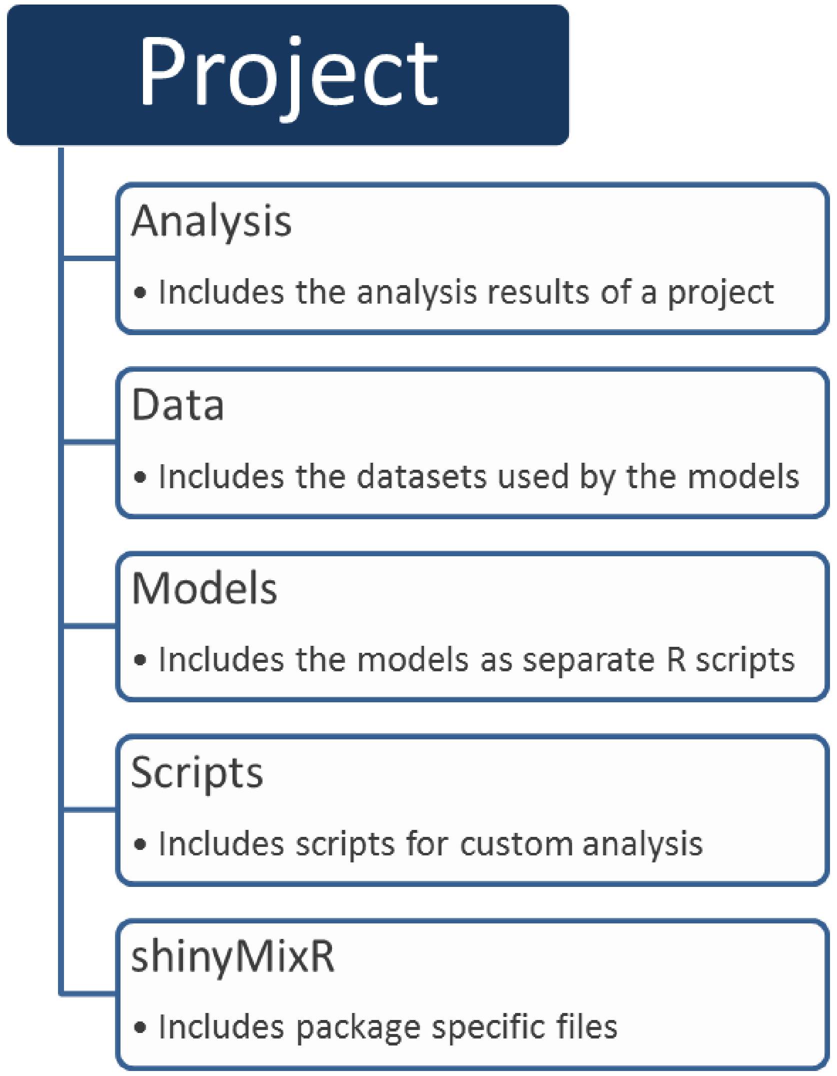
The folder structure is important because:

1. The package monitors changes in specific folders and keeps track of this in a project object

2. Files are read and saved from specific locations to disk

3. When working with many models an organized folder structure is key – model, data, metadata, setting and results are kept together

*functionality to automatically build the folder structure is present in the package



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

Project Object

To manage the information within the project structure, a **project object** is maintained:

- **The object has information regarding the available models, meta data, high level results, etc.**
- All changes within a project are monitored/saved to disk in this object:
    - When model is changed
    - When new results are generated
    - When data is deleted
- Within the interface this is done automatically, or using refresh buttons
  (e.g. the interface does not “know” when a model is finished and new results are present)
- *Note*: for an interactive session, in some cases updating of this object can be done using the `get_proj()` function

nlmixr and shinyMixR - nuances

nlmixr – model and nlmixr fit function

```
m1 <- function() {
  ini({
    tka <- .5
    tcl <- -3.2
…
fit1 <- nlmixr(m1, theo_sd, est="saem", … )
```

run nlmixr() via R or use the **Run model(s)** widget via shinyMixR

shinyMixR – metadata and run button or run_nmx function

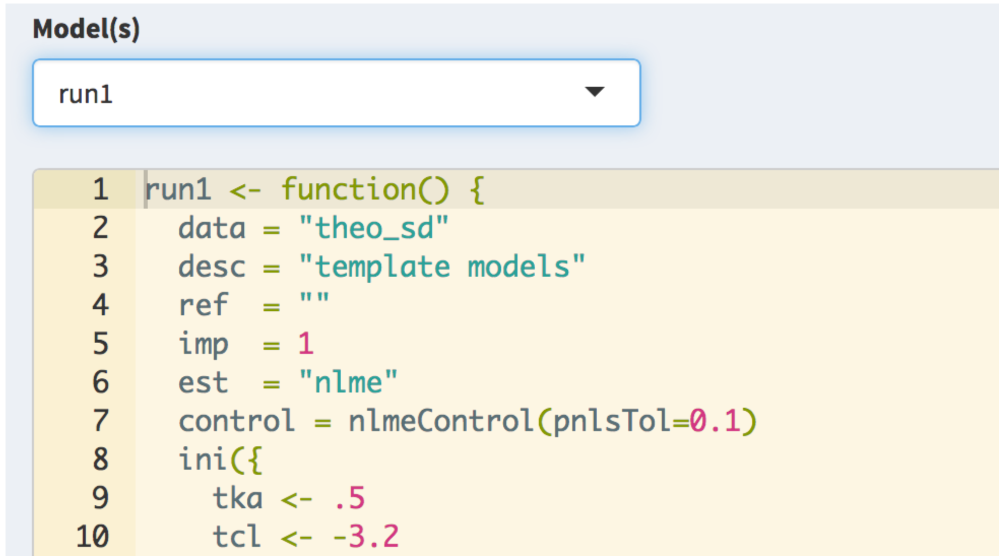
```
run1 <- function() {
  data = "theo_sd"                    # csv or rds file
  desc = "case example base run"      # model description
  ref  = ""                           # model reference
  imp  = 1                            # model importance
  est  = "saem"                       # estimation method
  control = list()                    # est control options
  ini({
    tka <- .5
    tcl <- -3.2
…
```

nlmixr arguments moved to metadata within the model code (run1.R model file) – analogous to NONMEM $PROBLEM, $DATA, $EST, etc.

shinyMixR stores results in R data objects (e.g. run1.res.rds)

Overview of shinyMixR functionality

Edit Models

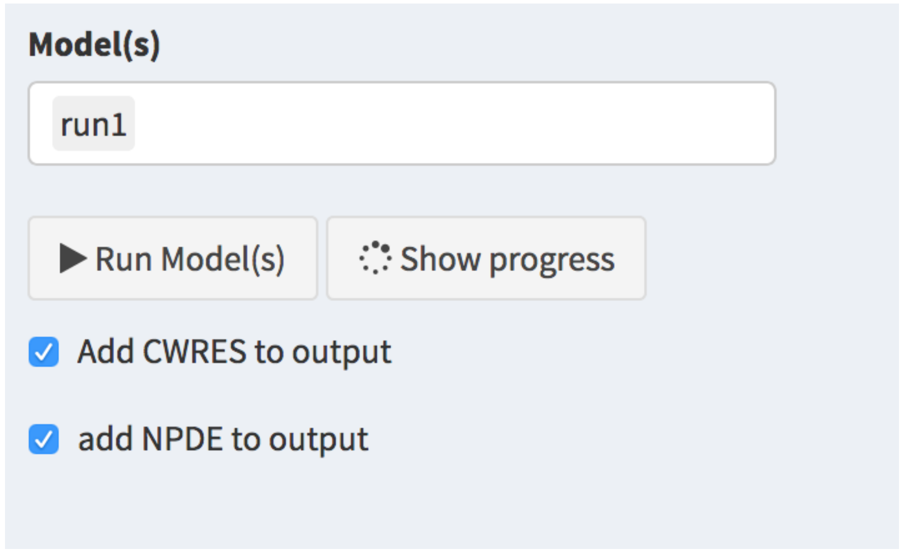


```
run1 <- function() {
  data = "theo_sd"
  desc = "template models"
  ref  = ""
  imp  = 1
  est  = "nlme"
  control = nlmeControl(pnlsTol=0.1)
  ini({
    tka <- .5
    tcl <- -3.2
```

The **edit model(s)** widget is used to edit models within an editor including syntax coloring (using **shinyAce**).

It is also possible to create new models using various templates or to duplicate existing models.

Run Models



Add CWRES to output [✓]

add NPDE to output [✓]

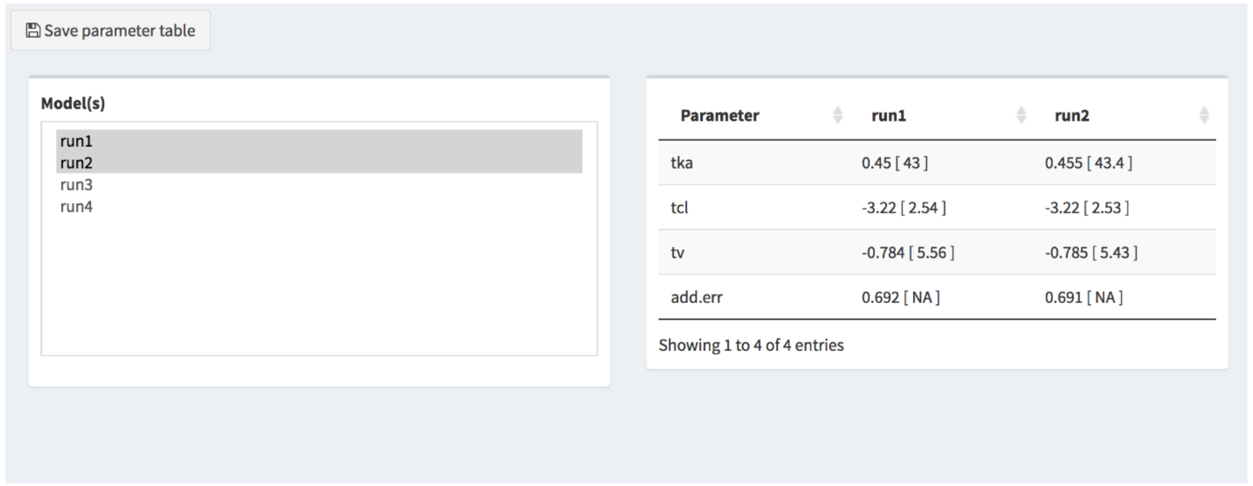
The **run model(s)** widget is used to run models. It is possible to run one or multiple models at once.

It is also possible to assess the intermediate output or progress for an nlmixr run.

data in the **data** folder; models in the **model** folder

Overview of shinyMixR functionality

Parameter Estimates



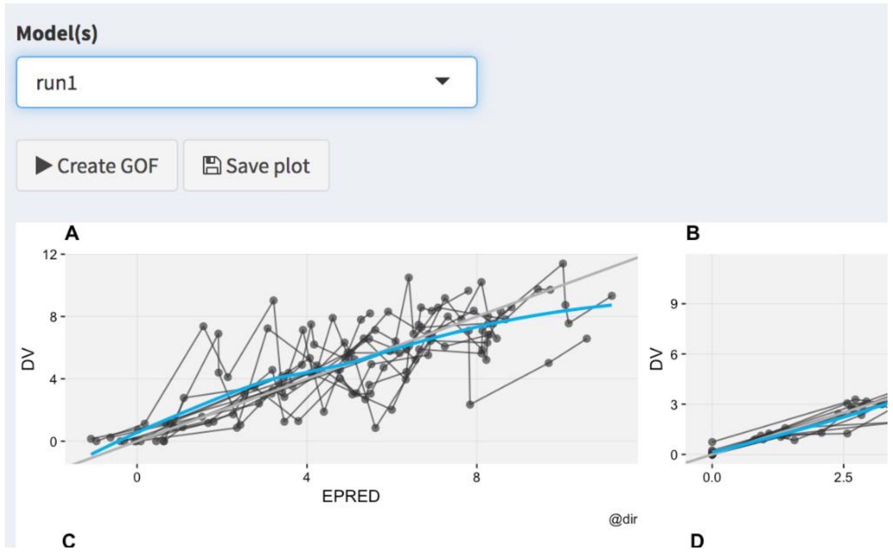
Parameter	run1	run2
tka	0.45 [43]	0.455 [43.4]
tcl	-3.22 [2.54]	-3.22 [2.53]
tv	-0.784 [5.56]	-0.785 [5.43]
add.err	0.692 [NA]	0.691 [NA]

Showing 1 to 4 of 4 entries

The **parameter estimates** widget is used to generate a table with parameter estimates. In case multiple models are selected the table will show the results of each run in a separate column.

Output stored in the **analysis** folder; fit results and specific files in the **shinyMixR** folder

GOF Plots

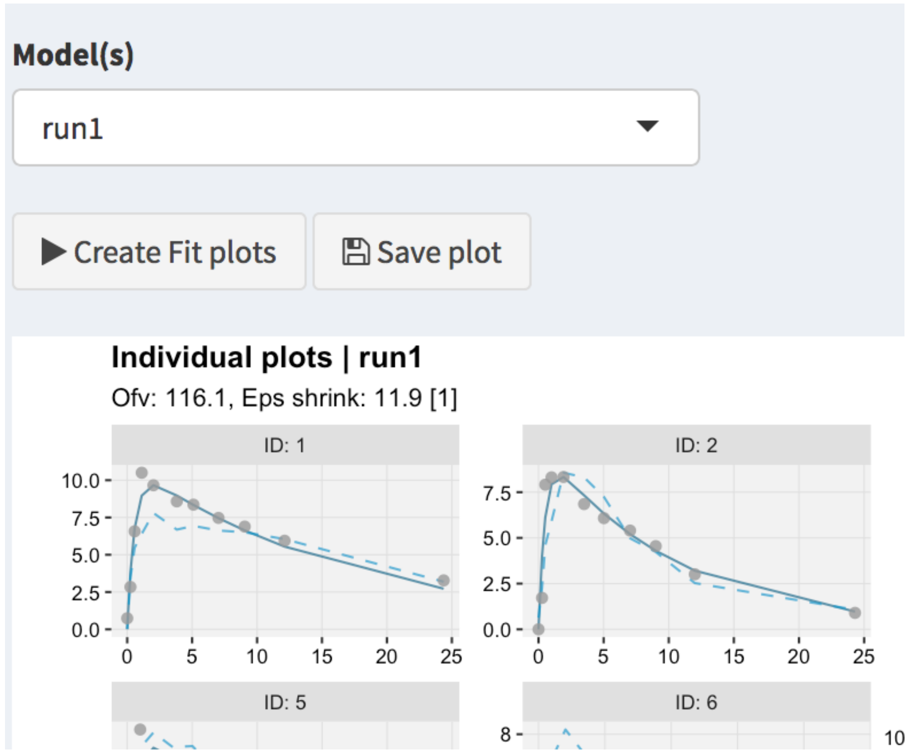


The **goodness of fit** widget is used to generate a combination of 4 goodness of fit plots combined.

By default **nlmixr.xpose** is used but direct **ggplot2** can also be used directly by specifying this in the **settings** widget.

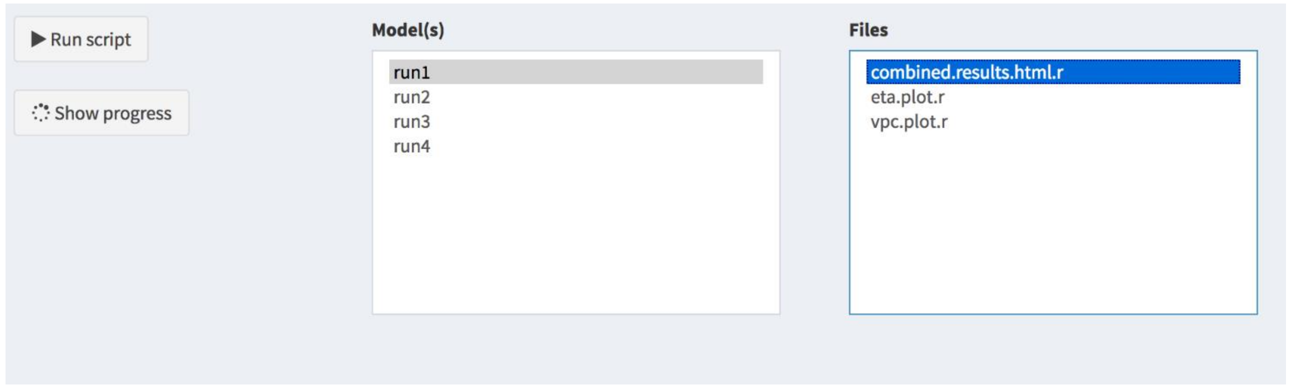
Overview of shinyMixR functionality

Fit Plots



The **fit plots** widget is used to generate individual fit plots. The same plotting options are present here as for the goodness of fit plots.

Scripts

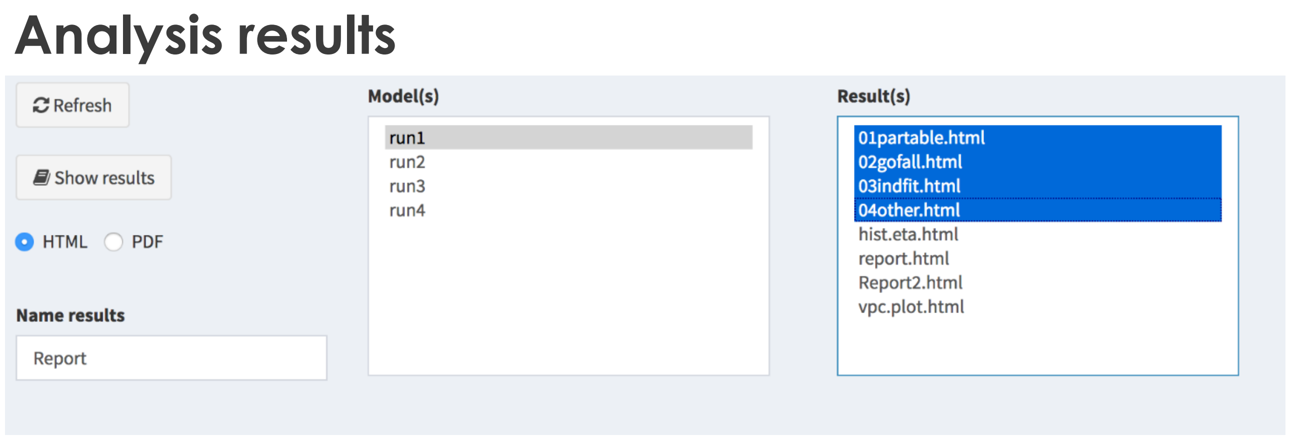


It is possible to write your own scripts (See **scripts** folder and **scripts** widget) that can be used to analyze model results.

The scripts can be used to process the result for one or multiple models at once (the interface will include the name of the selected models in the script).

Output stored in the **analysis** folder; scripts in the **scripts** folder; fit results and specific files in the **shinyMixR** folder

Overview of shinyMixR functionality



Analysis results

Refresh

Show results

[✓ HTML] PDF

Name results

Report

Model(s)

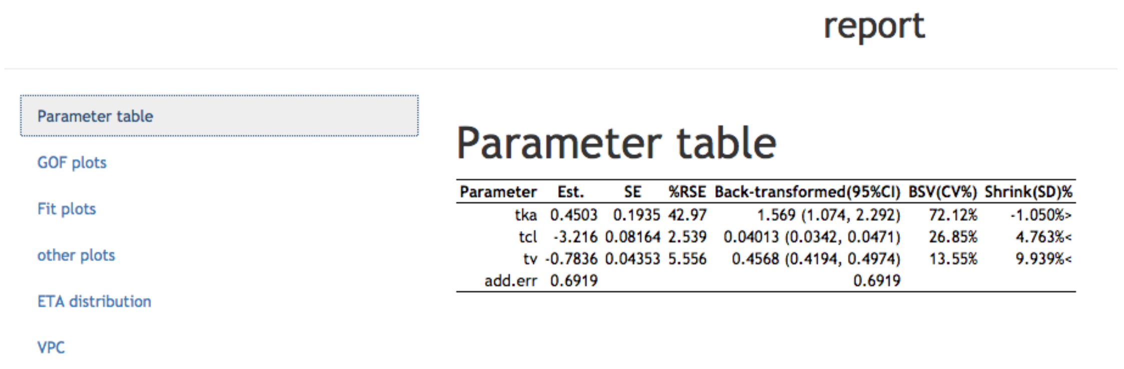
run1
run2
run3
run4

Result(s)

01partable.html
02gofall.html
03indfit.html
04other.html
hist.eta.html
report.html
Report2.html
vpc.plot.html

Output from Scripts and Widgets (plots and tables) are available in the **Analysis results** widget.

It is possible to save, view and combine the results from the models within a project into an HTML or PDF (if LaTeX is present) document.



report

Parameter table
GOF plots
Fit plots
other plots
ETA distribution
VPC

Parameter table

Parameter	Est.	SE	%RSE	Back-transformed(95%CI)	BSV(CV%)	Shrink(SD)%
tka	0.4503	0.1935	42.97	1.569 (1.074, 2.292)	72.12%	-1.050%>
tcl	-3.216	0.08164	2.539	0.04013 (0.0342, 0.0471)	26.85%	4.763%<
tv	-0.7836	0.04353	5.556	0.4568 (0.4194, 0.4974)	13.55%	9.939%<
add.err	0.6919			0.6919		

Output stored in the **analysis** folder

Summary of shinyMixR

- The `shinyMixR` package is a **graphical user interface (GUI)** tool for managing popPKPD projects with `nlmixr` as the estimation engine
- It structures your project – model, data, metadata, settings and results are kept together and saves the results to disk
- The interface enables browsing between specific project folders
- The package has functionalities to view, edit, run, compare, analyze and report nlmixr models
- Models are submitted in separate sessions – R can be used while running models

https://richardhooijmaijers.github.io/shinyMixR/index.html

Back-up

Create New Project (general) – via `shinyMixR` command line

1. Create a folder for your project (e.g., <ProjectFolder>)
2. Open R or RStudio and set the working directory to your project folder (e.g., <ProjectFolder>)
   - Use `setwd()`, or
   - via Rstudio: *Session > Set working directory > Choose directory*
3. On the command line or in a script run
   - `library(shinyMixR, quietly=TRUE)`
   - `create_proj()` (only needed once per project)

- By default, the folder structure is created within the current directory, if not present. The following folders are created:
  - **analysis, data, models**, **scripts**, **shinyMixR**

- Once there is a folder structure present, the interface can be started:

4. On the command line or in a script run
   - `library(shinyMixR, quietly=TRUE); run_shinymixr(launch.browser=TRUE)`
   - *Note*: other functions like `run_nmx()` can also be used directly on the command line or in a script to run models

- Start creating and running your models
  - If needed, supplied Model(s) can be copied to the models folder
  - If needed, supplied Data can be copied to the data folder

Return To Existing Project (general)

1. Click the **shortcut** (Windows, Mac or Linux specific)
   - ➢ *The shortcut should only be copied the first time you start using shinyMixR*
2. The shinyMixR application opens
3. Browse to your project folder (e.g., <ProjectFolder>)

OR

1. Open R or Rstudio and set the working directory to your project folder (e.g., <ProjectFolder>)
   - ➢ Use `setwd()`, or
   - ➢ via RStudio: *Session > Set working directory > Choose directory*
2. On the command line or in a script run
   - ➢ `library(shinyMixR, quietly=TRUE)`
   - ➢ `run_shinymixr(launch.browser=TRUE)` → if interface run in web browser

- The interface will open up showing all models previously stored in this directory.

Working with Multiple Locations using shinyMixR

- **library**(shinyMixR)
- *# Create two different example projects:*
- **create_proj**("./exampleA")
- **create_proj**("./exampleB")
- *# It is also possible to define projects as absolute paths:*
- *# create_proj("C:/absolute/path/exampleC")*
- *# Work with shinyMixR in first location (working directory is automatically set to this location)*

- **run_shinymixr**(wd="./exampleA",launch.browser=TRUE)
- *# Work with shinyMixR in second location*

- **run_shinymixr**(wd="./exampleB",launch.browser=TRUE)
- *# Example for absolute paths*
- *# run_shinymixr(wd="C:/absolute/path/exampleC",launch.browser=TRUE)*

Installation of shinyMixR

- To get started, first install the package using:
  - devtools::install_github("richardhooijmaijers/shinyMixR")
- Be aware that the `nlmixr`, `nlmixr.xpose` and `R3port` package should be installed before installing `shinyMixR`, e.g.:
  - devtools::install_github("richardhooijmaijers/R3port")
  - devtools::install_github("nlmixrdevelopment/nlmixr")
  - devtools::install_github("nlmixrdevelopment/xpose.nlmixr")

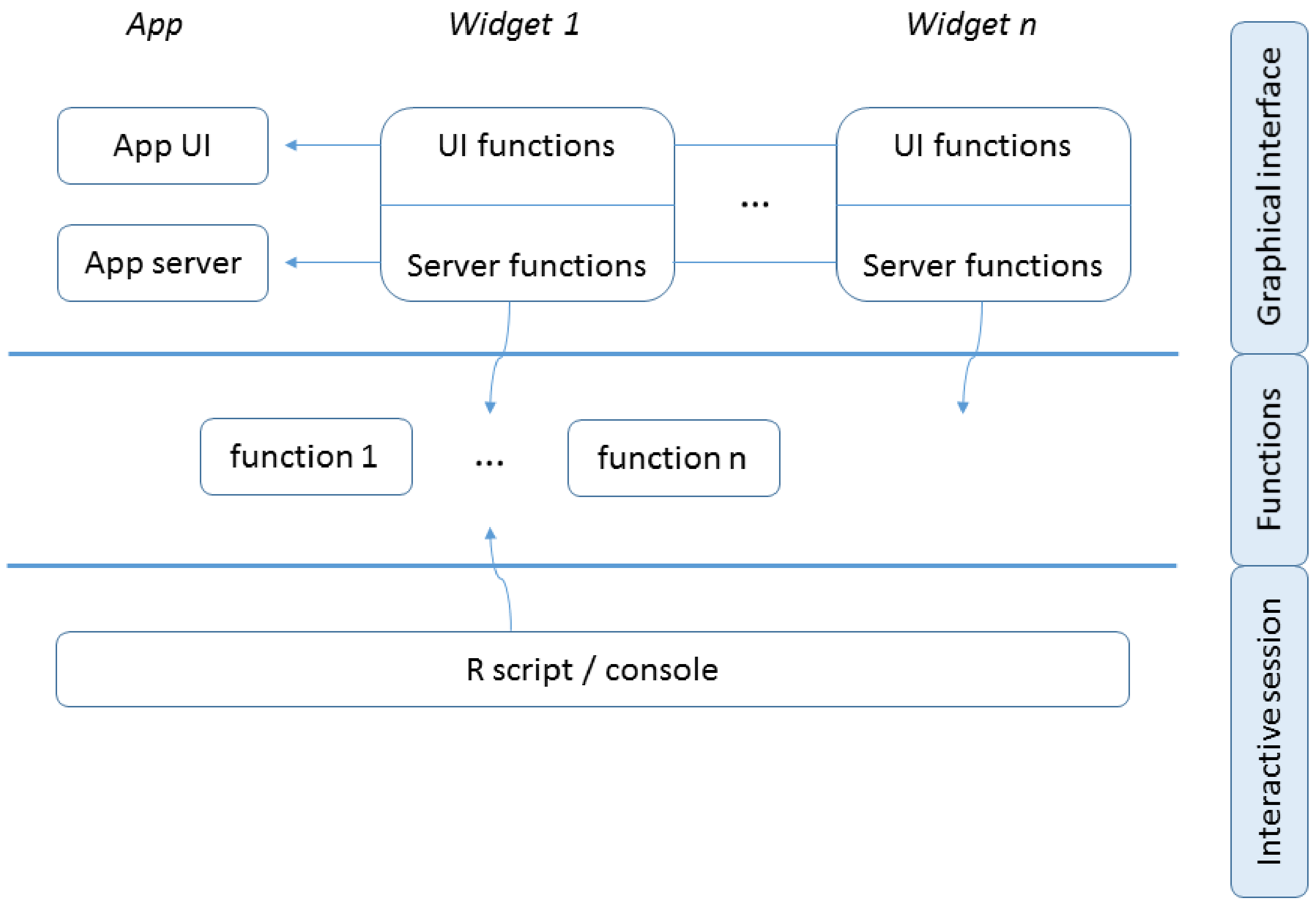
Package Information for shinyMixR

shinyMixR is a shell around nlmixr and needs the package to fully operate:

- Running models is done using the `nlmixr` package (indirectly)
- Plotting is done using the `xpose.nlmixr` package (or `ggplot2`)

- Managing is done using the `DT` and `collapsibleTree` packages
- Editing is done using the `shinyAce` package
- Reporting is done using the `R3port` package

Package structure



- The interface is build using ui/server scripts as done in other shiny apps
- Due to the size of the dashboard, widget *modules* were created that are used by ui/server
- More generic functions are available that are used by the interface as well as interactive usage

Differences between Interactive Session and Interface

Functionality available in <u>both</u>

- View overview of available models
- View hierarchical overview
- Run models externally
- Create parameter table
- Create GOF plots
- Create fit plots

Functionality only available in <u>interface</u>

- Export overview
- Adapt model meta data
- Edit, duplicate, create model
- Show progress of model runs
- Combine analysis results in report
- Run user created R template script

- Functionality only available in interface can be done in many cases indirectly in an interactive session as well
- It is more user-friendly/quicker to perform certain tasks using the interface
- It is easy to switch between interface and interactive session

Workflow – via nlmixr

```r
run1 <- 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)
  })
}

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

fit <- nlmixr(run1, dat, est="saem")
```

A model can be directly run in nlmixr:

1. Define model using the unified user interface.
2. Import, create and/or adapt the required data.
3. Use the nlmixr function to run the model.

Workflow – model diagnostics

```r
print(fit)
plot(fit)



xpdb <- xpose_data_nlmixr(fit)

dv_vs_idv(xpdb)
ipred_vs_idv(xpdb)
pred_vs_idv(xpdb)
dv_preds_vs_idv(xpdb)
dv_vs_pred(xpdb)
dv_vs_ipred(xpdb)
res_vs_idv(xpdb, res = 'CWRES')
res_vs_pred(xpdb, res = 'CWRES')
```

High level results can be printed in console and default plots can be created using `nlmixr`.

More elaborate plots can be generated using the `xpose.nlmixr` package

Most important function is `xpose_data_nlmixr` which transforms the nlmixr output to xpose format

Subsequently almost all xpose functions can be used to create results for nlmixr output
Only a few functions are displayed, for more examples see:
https://uupharmacometrics.github.io/xpose/index.html

Workflow – via shinyMixR command line

```
# show first part of model
cat(readLines("models/run1.r")[1:7],sep="\n")

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

# command line
run_nmx("run1")
res <- readRDS("shinyMixR/run1.res.rds")
gof_plot(fit)
fit_plot(res,type="user")


# interface
run_shinymixr()
```

The model is defined in a separate file (run1.r) and includes metadata used by the package

A model is submitted by default in a separate R session. Plot functions are available to create `xpose.nlmixr` or ggplot2 type plots

The interface can be started using a single function

nlmixr