Pharmpy and assemblerr — Two Novel Tools to Simplify the Model Building Process in NONMEM

Rikard Nordgren (1), Sebastian Ueckert (1), Stella Belin (1), Gunnar Yngman (1), Simon Carter (1), Simon Buatois (2), João A. Abrantes (2), Andrew C. Hooker (1), Mats O. Karlsson (1)
(1) Department of Pharmacy, Uppsala University, Sweden (2) Roche Pharma Research and Early Development, Roche Innovation Center Basel, Basel, Switzerland

The development of pharmacometric nonlinear mixed effect models is a complex, multi-step process that requires knowledge in multiple domains, including pharmacology and statistics. The technical component, i.e. how to implement and manipulate models, is an additional burden. In this work, we present two novel open-source packages to decrease the technical hurdles in the model building process; assemblerr for the generation of models from predefined components, and Pharmpy for manipulating models and processing of results. Both software are designed as standalone tools but are especially powerful when used together.

assemblerr

assemblerr is an open-source R package [1] for constructing pharmacometric models from scratch. It is intended to simplify the specification of pharmacometric models and provides a mechanism to generate them in an automatic way. With assemblerr, models are specified using R code which assemblerr can render into a NONMEM [2] control stream (support for other tools is planned).

Example code

In assemblerr, models are described via predefined components that are added together in a similar manner to `ggplot` graphics. The package provides components with different levels of abstraction, both high-level pharmacokinetic (PK) building blocks such as an absorption transit compartment, and low-level building blocks such as flows between compartments. Model tasks, e.g. estimation, are defined via dedicated components. After defining the model, you can check for potential issues in the model. In a final step, the defined model and the requested tasks can be converted to NONMEM code using the render function. As an example, using high-level building blocks, the following code defines a simple PK model as well as some tasks, and renders them:

```
# define a model
m <- pk_model() +
    pk_absorption_fo() +
    pk_distribution_1cmp() +
    pk_elimination_linear() +
    obs_additive(conc~C["central"])
# define estimation and output tasks
tsks <- tsk_estimation("foce") +
        tsk_output("pars", variables = vars_prms())
# check for potential issues and create NONMEM code
check(m)
render(model = m, tasks = tsks)
```

This snippet yields the following code:

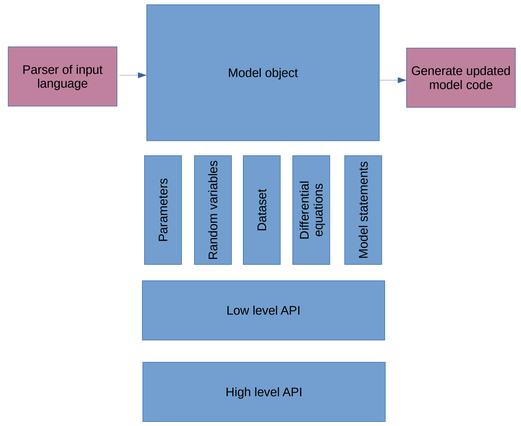
```
$PROBLEM
$INPUT ID TIME DV AMT
$DATA data.csv IGNORE=@
$SUBROUTINES ADVAN2  TRANS2
$PK
MAT = THETA(1) * EXP(ETA(1))
VC = THETA(2) * EXP(ETA(2))
CL = THETA(3) * EXP(ETA(3))
KA = 1/MAT
V = VC
$ERROR
CONC = A(2)/VC
Y = CONC + EPS(1)
$ESTIMATION METHOD=COND MAXEVAL=999999
$TABLE MAT VC CL FILE=pars NOAPPEND NOPRINT
$THETA (0, 0.5, Inf) ; POP_MAT
$THETA (0, 100, Inf) ; POP_VC
$THETA (0, 50, Inf) ; POP_CL
$OMEGA 0.1 ; IIV_MAT
$OMEGA 0.1 ; IIV_VC
$OMEGA 0.1 ; IIV_CL
$SIGMA 1; RUV_ADD
```

Features

- Flexible assembly of models from predefined components: parameter model, observation model, compartment, flow, algebraic relationship.
- Support for high-level components of PK models: absorption, distribution, elimination.
- Automatic recognition and optimized code generation for linear ODEs and ODEs with an analytic solution.
- Simple model task specification with `tidyselect` syntax for output variable selection.
- Option to use mu-referencing for models.

Pharmpy

Pharmpy is an open-source Python package [3] for pharmacometric modeling. The package can be used directly in Python, in R via the `pharmr` package or from the command line interface. It has functionality ranging from reading and manipulating model files/datasets to executing workflows and presenting results. Pharmpy is intended to be useful to tool developers, methodological researchers and modelers, and has different Application Programming Interface (API) layers to cater for the different needs of the groups.



The main design principles of Pharmpy are modularity, so that parts of tools can be reused independently, and tool agnosticism, so that multiple model languages and tools for estimation and simulation of models can be used. At the core of Pharmpy lies its abstraction for non-linear mixed effects models. Models are internally separated into components. The parameters, random variables, differential equations and model statements have their own classes with APIs allowing for low level manipulation.

Example code

Continuing the assemblerr example, the NONMEM model can be read, transformed by adding a peripheral compartment, and finally be run in NONMEM. This can be done as a pipeline using the `pharmr` interface:

```
m <- read_model('path/to/model/run1.mod') %>%
      add_peripheral_compartment() %>%
      update_source() %>%    # update NONMEM code
      fit()                  # run NONMEM
```

The NONMEM code will now be updated accordingly:

```
...
$SUBROUTINES ADVAN4  TRANS4
...
$PK
VP1 = THETA(5)
QP1 = THETA(4)
...
KA = 1/MAT
V2 = VC
Q = QP1
V3 = VP1
...
$THETA (0, 50, Inf) ; POP_CL
$THETA  (0,50.0) ; POP_QP1
$THETA  (0,5.0) ; POP_VP1
...
```

After fitting the model, different NONMEM results such as OFV and parameter estimates can be accessed from the model object.

Features

- Parsing of NONMEM model files (created manually or by using assemblerr), dataset, and result files.
- Transformation of models (e.g. adding and removing compartments, adding covariate effects, changing the IIV structure).
- Run NONMEM models directly from Python/R.
- Create and read in results from PsN tools (e.g. FREM, cdd, simeval) in .json-format.

Availability

assemblerr is available on CRAN and can be installed using `install.packages("assemblerr")`. More information on the package is available here:

`https://uupharmacometrics.github.io/assemblerr/`

Pharmpy is available from the Python package index and can be installed via `pip install pharmpy-core`. The Pharmpy webpage (with information about `pharmr` as well) can be found here:

`https://pharmpy.github.io/`

Acknowledgments

This work was supported by F. Hoffmann-La Roche Ltd., Basel, Switzerland. A special thanks to Dr. Emilie Schindler, Dr. Sylvie Retout and Dr. Valérie Cosson for conducting testing and giving feedback.

[1] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021.
[2] Robert J. Bauer. NONMEM Tutorial Part I: Description of Commands and Options, With Simple Examples of Population Analysis. *CPT: Pharmacometrics & Systems Pharmacology*, 8(8):525–537, 2019.
[3] Drake FL Van Rossum G. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.