Pharmpy: a versatile open-source library for pharmacometrics

Rikard Nordgren, Stella Belin, Xiaomei Chen, Andrew C. Hooker, Mats O. Karlsson
Department of Pharmacy, Uppsala University, Sweden

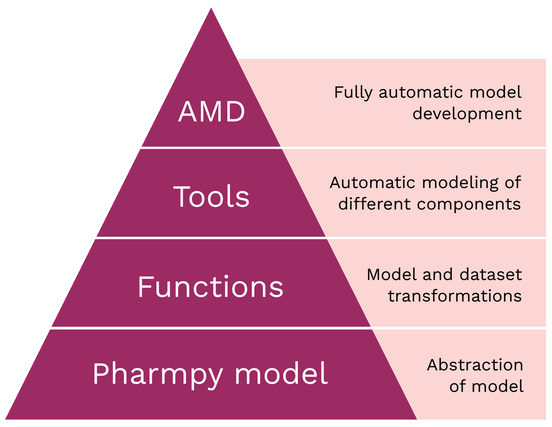
https://pharmpy.github.io/

INTRODUCTION

Pharmpy is an open-source software package for R and Python. It is intended for researchers and modelers as well as tool developers to aid in pharmacometric modeling. It provides functionality ranging from reading and manipulating model files and datasets to running advanced tools. The two cornerstones of Pharmpy are:

Flexibility in usage

Pharmpy has three main layers: the Pharmpy model, modeling functions to manipulate models, and tools that can fully or partly develop models. Each layer can be used by itself, so you can use whichever suits your use case. Pharmpy can be used in Python, R (via the pharmr package), and the command line.



Independence of modeling language

The model abstraction is independent from the estimation language, allowing transformations and tools to be agnostic about model type. This general representation of a model facilitates using different software for different tasks (e.g. estimation and simulation) while using the same Pharmpy model object as the base. Furthermore, Pharmpy can translate models between languages, and update the model code as needed.

New in Pharmpy 2.1.1

- New tool pdsearch to build PD and KPD models
- New tool ModelRank for strictness and ranking
- Extended functionality of IIVSearch (supports additive IIVs and MFL to fine-tune search space)
- Support N-transits and Weibull absorption in ModelSearch
- Various new modeling functions

SCRIPT YOUR WORKFLOWS

Modeling software support

- NONMEM (read, write, execute)
- nlmixr2 (write, execute)
- rxode2 (write)

When a model is parsed from NONMEM or is created from scratch in Pharmpy, it is represented as a general Pharmpy model object. The model is broken down into core components (model statements, parameters, random variables, execution steps, and the dataset), each of which can be manipulated via low-level operations or higher-level functions. Pharmpy provides many different functions to simplify pharmacometric model building, from simple tasks which are tedious to perform manually (e.g. handling BLQ observations) to full scripting of a workflow, including fitting and presenting results.

Example 1: BLQ handling and BIC

```
init_ests <- list(POP_CL=30.0, POP_VC=100.0, POP_MAT=0.3)
m <- create_basic_pk_model('oral', 'moxo.csv') |>
     set_initial_estimates(init_ests) |>
     transform_blq(method='m3', lloq=0.1) |>
     convert_model(to_format='nonmem')

res <- fit(m)
bic <- calculate_bic(m, res$ofv, type='mixed')
```

Example 2: scripting from manually built model

```
run1 <- read_model('run1.ctl')
res1 <- read_modelfit_results('run1.ctl')

run2 <- set_initial_estimates(m, inits=res$parameter_estimates) |>
        set_michaelis_menten_elimination() |>
        set_name(new_name='run2')

res2 <- fit(run2)
plot_dv_vs_pred(run2, predictions=res2$predictions)
```

Example 3: running bootstrap

```
m <- load_example_model('pheno')
res <- load_example_modelfit_results('pheno')

res_boot <- run_bootstrap(m, res, samples=10)
res_boot$covariance_matrix
res_boot$parameter_estimates_correlation_plot
```

AUTOMATICALLY DEVELOP MODELS

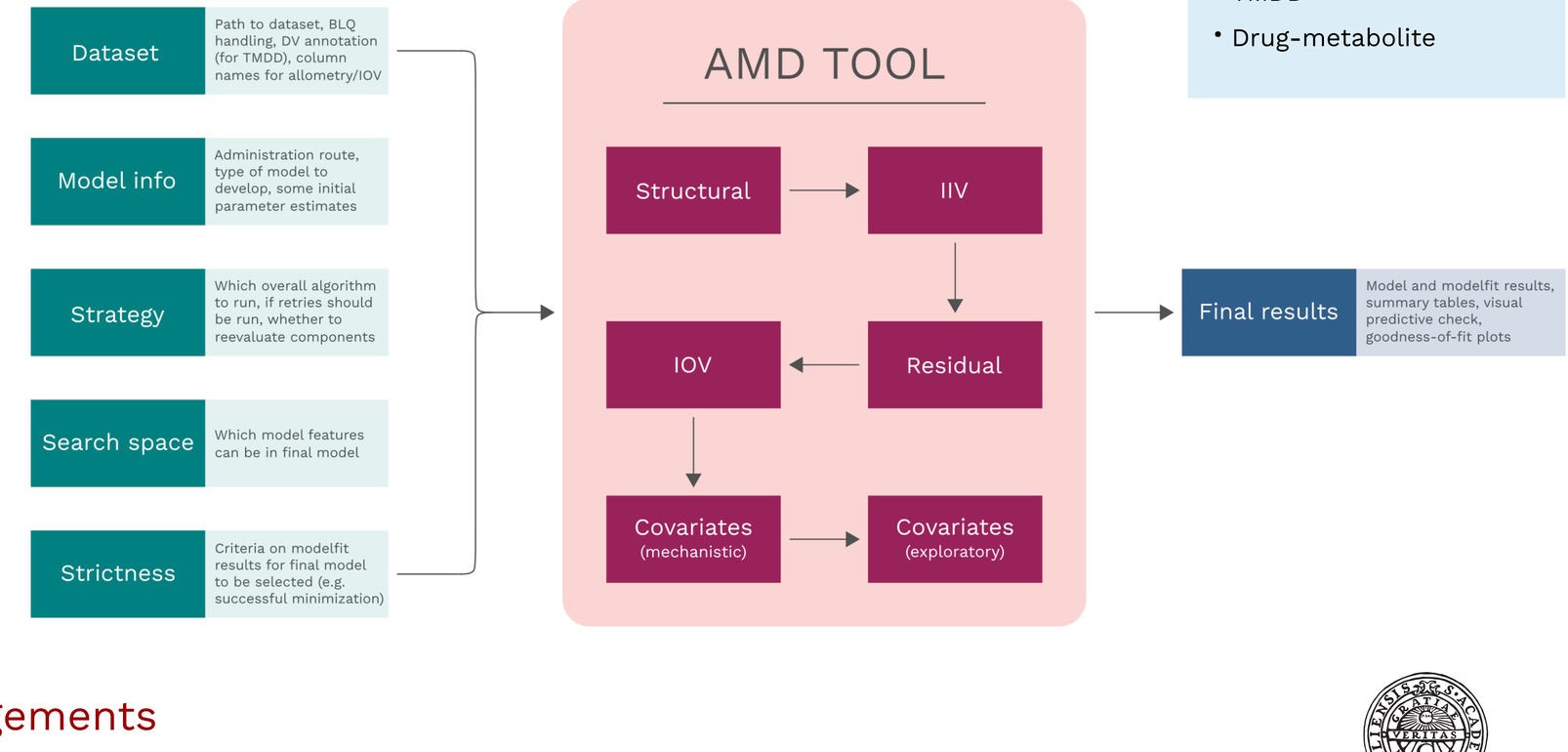
Pharmpy has multiple tools that can automatically develop different aspects of a model, and a fully automatic model development (AMD) tool that develops a model given a dataset. The AMD tool integrates several subtools that selects the structural model, the IIV/IOV structure, the RUV model, and covariate effects. These subtools can also be run standalone. All AMD tools produce results such as a final model and reports with model quality plots and tables. Pharmpy also has tools for other aspects of modeling, including bootstrap, visual prediction check (VPC), and a comparison tool to evaluate estimation methods, solvers, and parameter uncertainty methods (estmethod).

Supported types

- PK
- PD
- PKPD
- KPD
- TMDD
- Drug-metabolite

```
search_space <- 'COVARIATE?(@IIV,[WT,AGE,RF],
[LIN,EXP]);COVARIATE?(@IIV,[SEX,NYHA],CAT)'

res <- run_amd(input='moxo.csv',
               modeltype='basic_pk',
               administration='oral',
               cl_init=30.0,
               vc_init=100.0,
               mat_init=0.3,
               search_space=search_space,
               mechanistic_covariates=c('WT'),
               occasion='VISI')
```



In collaboration with

Acknowledgements

This work was supported by F. Hoffmann-La Roche Ltd., Basel, Switzerland. A special thanks to Dr. Orwa Albitar and Dr. José Calderin for conducting testing and providing feedback. Thanks also to Pharmetheus and Dr. Aurélien Ooms for optimizing NONMEM parsing

UPPSALA UNIVERSITET