pmparams: an R Package for Defining and Formatting Parameter Tables in Pharmacometric Modeling

Grace O'Brien, M.S., Eric Anderson, M.S., Michael McDermott, M.E., Kyle Barrett, Kyle T. Baron, Pharm.D., Ph.D., Timothy Waterhouse, Ph.D., Seth Green, M.S., Katherine Kay, Ph.D.

Metrum Research Group, Boston, MA, USA

Abstract

Objectives: A key output of pharmacometric modeling is an informative, well-formatted table summarizing the estimated parameters. For each parameter, users must define all names and units, calculate confidence intervals (CIs) and/or relative standard errors (RSEs), and perform any required back-transformations; an essential but often tedious and time-consuming activity. The aim was to provide a simple, reproducible, and traceable method for generating parameter tables in R for NONMEM® models. Attained via a new package (**pmparams**): a stable and easy-to-use tool intended to integrate with, and extend the functionality of, existing packages in the Metrum Research Group Ecosystem (MeRGE) [1].

Methods: **pmparams** is an open-source R package [2] to define consistent and well-formatted parameter tables for NONMEM® models that leverage existing features of MeRGE packages (**bbr** [3] and **pmtables** [4]). Metrum Research Group applied a transparent software development life cycle with robust planning, iterative development, testing, validation, and release. The R package was formally documented and included a comprehensive test suite. New features are continually added to refine and expand package applications.

Results: To create a parameter table, users pass a data frame of raw parameter estimates and a parameter key to **pmparams** functions, creating a well-formatted data frame of transformed parameter estimates in three steps. The parameter key (commonly defined in YAML format) includes: parameter abbreviations and units (including Greek symbols via LaTeX if desired), descriptive names, panel labels for parameter type (e.g., "structural", "covariate", etc.), and required transformations. Defining the parameter key in a YAML file, rather than in the model control file, provides additional flexibility to update the parameter table without modifying the control stream. If the underlying model structure is unchanged, a single parameter key can often be used to make tables for the base model, covariate model, and bootstrap runs (where **pmparams** also summarizes the variability of bootstrap estimates). The **pmparams** output (a data frame) can be passed to **pmtables** to seamlessly render a report-ready table (.tex, .png, or .jpeg file).

Conclusion: The **pmparams** R package is a flexible tool for creating reproducible, traceable, report-ready parameter tables for pharmacometric modeling. Future releases will focus on parameter tables for Bayesian models run using Bayesian estimation methods in NONMEM® or Stan.

MeRGE

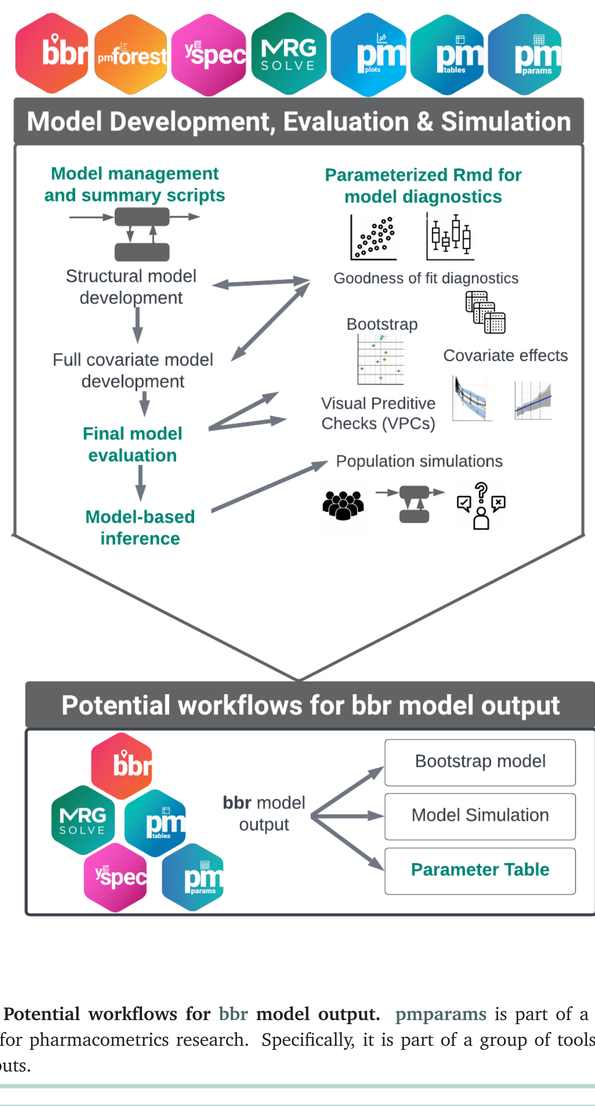

Figure 1. Potential workflows for bbr model output. **pmparams** is part of a suite of tools developed for pharmacometrics research. Specifically, it is part of a group of tools that use **bbr** model outputs.

Coming soon: Bootstrap parameter table

Metrum Research Group is developing a **pmparams** function that will help make a table summarizing bootstrap parameter estimates.

Metrum Research Group Publications and Posters

Workflow

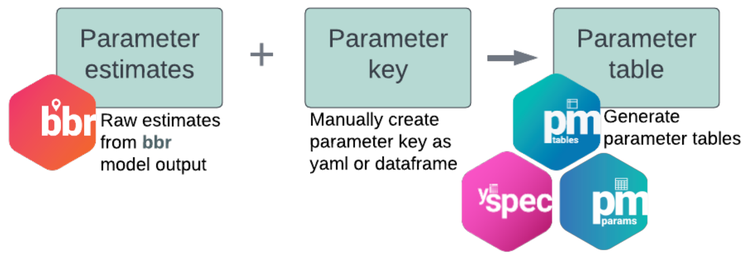

Figure 2. Parameter table workflow. Packages within MeRGE can be used to (1) extract your parameter estimates from NONMEM® (**bbr**), (2) describe your parameters in a key (**yspec**), and (3) format your parameters for reporting with functions in **pmparams** and **pmtables**.

Obtain parameter estimates

```
mod <- read_model("path/to/model")
model_summary(mod)
```

parameter_names	estimate	stderr	shrinkage
THETA1	0.434	0.0629	
THETA2	4.12	0.0276	
THETA3	1.12	0.0328	
THETA4	4.21	0.0192	
THETA5	1.29	0.0354	
OMEGA(1,1)	0.221	0.0530	17.9
OMEGA(2,2)	0.0827	0.00983	6.02
OMEGA(3,3)	0.169	0.0197	0.587
SIGMA(1,1)	0.0399	0.00123	5.28

Figure 4. Parameter estimates. Example of raw parameter estimates from a **bbr** model.

Create parameter key

Users can conveniently describe NONMEM® parameters in a yaml file. A single parameter key can often be used to make tables for the base model, covariate model, and bootstrap runs. Users can specify the following:

- name or yaml variable
- *abb* : abbreviation (use LaTeX coding)
- *desc* : description
- *panel* : indicate which panel each parameter should appear under
- *trans* : define how the parameter will be transformed

```
THETA1:
  abb: "KA (1/h)"
  desc: "First order absorption
        rate constant"
  panel: struct
  trans: logTrans
THETA2:
  abb: "V2/F (L)"
  desc: "Apparent central volume"
  panel: struct
  trans: logTrans
OMEGA11:
  abb: "IIV-KA"
  desc: "Variance of absorption"
  panel: IIV
  trans: lognormalOm
OMEGA21:
  abb: "V2/F-KA"
  desc: "Covariance of V2/F - KA"
  panel: IIV
  trans: none
SIGMA11:
  abb: "Proportional"
  desc: "Variance"
  panel: RV
  trans: propErr
```

Panel and transformation options:

Panel options	
struct	Structural model parameters
cov	Covariate effect parameters
IIV	Interindividual covariance or variance parameters
IOV	Interoccasion variance parameters
RV	Residual variance

Transformation options	
none	Untransformed parameters, e.g. THETAs or off-diagonals
logTrans	THETAs estimated in the log domain
logitTrans	THETAs estimated using a logit transform
lognormalOm	log-normal OMEGAs, e.g. CL = THETA(1) * exp(ETA(1)) - returns est.+CV%
OmSD	OMEGAs where you want to return SD only, e.g. additive OMEGAs CL = THETA(1) + ETA(1) - returns est.+SD
logitOmSD	OMEGAs using logit transform, requires the associated THETA separated with a '~', e.g. logitOmSD ~ THETA3 - returns est.+SD
addErr	Additive error terms- returns est.+SD
propErr	Proportional error terms- returns est.+CV%

Figure 3. Parameter key. Example of an entry in a parameter key yaml file.

Customization options

pmparams has flexible input options. For the parameter estimates, user can input the path to a **bbr** model, a **bbr** model object, a **bbr** model summary object, or a data frame of parameter estimates. For the parameter key, user can input the path to a parameter key yaml or parameter key data frame.

Customization options	
define_param_table()	
.ci	Change confidence interval
format_param_table()	
.prse	Add Relative standard error
.cleanup_cols	Return all columns in data set
.digits	Change number of significant digits
.maxex	Change maximum number of significant digits before moving to scientific notation
make_pmtable()	
.pmtype	Change parameter table type: full, fixed, random, stuctural, covariate

Figure 5. Customizations. There are several argument options users can leverage to achieve desired table.

References

[1] MeRGE Expo. https://merge.metrumrg.com/expo/expo1-nonmem-foce.

[2] pmparams package documentation. https://github.com/metrumresearchgroup/pmparams.

[3] bbr package documentation. https://github.com/metrumresearchgroup/bbr.

[4] pmtables package documentation. https://github.com/metrumresearchgroup/pmtables.

Make parameter tables

Parameter data frame:
Users can view their transformed and formatted parameters in a data frame. This data frame can then be either saved out as a csv or further formatted for the user's needs.

```
full <- define_param_table(.estimates = mod, .key = paramKeyPath) %>%
format_param_table()
```

```
                                type      abb
1          Structural model parameters KA (1/h)
2  Interindividual variance parameters   IIV-KA
3 Interindividual covariance parameters  V2/F-KA
                 greek                                desc
1 $\\exp(\\theta_{1})$ First order absorption rate constant
2    $\\Omega_{(1,1)}$               Variance of absorption
3    $\\Omega_{(2,1)}$             Covariance of V2/F - KA
                value           ci shrinkage
1                1.54  1.36, 1.75         -
2  0.221 [CV\\%=49.7]  0.117, 0.324      17.9
3 0.0690 [Corr=0.511] 0.0299, 0.108         -
```

LaTeX parameter table:
Here we demonstrate how to use pmparams and pmtables together to create a LaTeX formatted table.

```
footnote <- param_notes(.ci = 95)

make_pmtable(full) %>%
st_notes(footnote$ci, footnote$cv, footnote$corr) %>%
st_notes_str() %>%
st_notes(footnote$ciEq, footnote$cvOmegaEq,
         footnote$cvSigmaEq) %>%
stable()
```

			Estimate	Shrinkage (%)	95% CI
Structural model parameters					
KA (1/h)	$\exp(\theta_1)$	First order absorption rate constant	1.54	-	1.36, 1.75
V2/F (L)	$\exp(\theta_2)$	Apparent central volume	61.5	-	58.2, 64.9
CL/F (L/h)	$\exp(\theta_3)$	Apparent clearance	3.05	-	2.86, 3.25
V3/F (L)	$\exp(\theta_4)$	Apparent peripheral volume	67.4	-	64.9, 69.9
Q/F (L/h)	$\exp(\theta_5)$	Apparent intercompartmental clearance	3.62	-	3.38, 3.88
Interindividual variance parameters					
IIV-KA	$\Omega_{(1,1)}$	Variance of absorption	0.221 [CV%=49.7]	17.9	0.117, 0.324
IIV-V2/F	$\Omega_{(2,2)}$	Variance of central volume	0.0827 [CV%=29.4]	6.02	0.0634, 0.102
IIV-CL/F	$\Omega_{(3,3)}$	Variance of clearance	0.169 [CV%=42.9]	0.587	0.130, 0.208
Interindividual covariance parameters					
V2/F-KA	$\Omega_{(2,1)}$	Covariance of V2/F - KA	0.0690 [Corr=0.511]	-	0.0299, 0.108
CL/F-KA	$\Omega_{(3,1)}$	Covariance of CL/F - KA	0.134 [Corr=0.694]	-	0.0878, 0.180
CL/F-V2/F	$\Omega_{(3,2)}$	Covariance of CL/F - V2/F	0.0735 [Corr=0.622]	-	0.0528, 0.0942
Residual variance					
Proportional	$\Sigma_{(1,1)}$	Variance	0.0399 [CV%=20.0]	5.28	0.0375, 0.0423

CI: confidence intervals; CV: coefficient of variation; Corr: correlation coefficient
CI = estimate $\pm$ 1.96 $\cdot$ SE
CV% of log-normal omegas = sqrt(exp(estimate) - 1) $\cdot$ 100
CV% of sigma = sqrt(estimate) $\cdot$ 100

Figure 6. Full parameter table. Use *param_notes()* to obtain relevant footnotes in a full parameter table.

Customization examples:
Below are examples of how users can customize their tables to include different confidence intervals, the percent RSE, or a subset of the parameters. When subsetting parameters, users have the option of (1) full, (2) fixed, (3) random, (4) structural, and (5) covariate.

```
define_param_table(.estimates = mod, .key = paramKeyPath,
                   .ci = 90) %>%
format_param_table(.prse = TRUE) %>%
make_pmtable(.pmtype = "structural") %>%
stable()
```

			Estimate	90% CI	RSE (%)
Structural model parameters					
KA (1/h)	$\exp(\theta_1)$	First order absorption rate constant	1.54	1.39, 1.71	6.29
V2/F (L)	$\exp(\theta_2)$	Apparent central volume	61.5	58.8, 64.3	2.76
CL/F (L/h)	$\exp(\theta_3)$	Apparent clearance	3.05	2.89, 3.22	3.29
V3/F (L)	$\exp(\theta_4)$	Apparent peripheral volume	67.4	65.3, 69.5	1.93
Q/F (L/h)	$\exp(\theta_5)$	Apparent intercompartmental clearance	3.62	3.42, 3.84	3.54

Figure 7. Structural parameter table. Use 90% CI and include RSE in a structural parameter table.

```
define_param_table(.estimates = mod, .key = paramKeyPath) %>%
format_param_table(.digit = 4) %>%
make_pmtable(.pmtype = "random") %>%
stable()
```

		Estimate	95% CI	Shrinkage (%)
Interindividual variance parameters				
IIV-KA	$\Omega_{(1,1)}$	0.2206 [CV%=49.7]	0.1168, 0.3244	17.90
IIV-V2/F	$\Omega_{(2,2)}$	0.08269 [CV%=29.4]	0.06343, 0.1020	6.023
IIV-CL/F	$\Omega_{(3,3)}$	0.1691 [CV%=42.9]	0.1304, 0.2078	0.5872
Interindividual covariance parameters				
V2/F-KA	$\Omega_{(2,1)}$	0.06901 [Corr=0.5109]	0.02988, 0.1081	-
CL/F-KA	$\Omega_{(3,1)}$	0.1340 [Corr=0.6938]	0.08778, 0.1802	-
CL/F-V2/F	$\Omega_{(3,2)}$	0.07350 [Corr=0.6216]	0.05283, 0.09418	-
Residual variance				
Proportional	$\Sigma_{(1,1)}$	0.03992 [CV%=20.0]	0.03751, 0.04232	5.275

Figure 8. Random effects parameter table. Use four significant digits for all values in a random effects parameter table.