A Showcase of Open-Source Tools for Scalable, Reproducible Pharmacometrics Workflows

12th Annual Indiana CTSI Disease and Therapeutic Response Modeling and Simulation Symposium

February 23rd, 2023

Schedule

2:30 pm - 2:45 pm Speaker introductions and overview of the ecosystem (Sam)

2:45 pm - 3:30 pm Model Output (Tim)

- Introduction to yspec and pmplots
- Model Diagnostics (bbr, yspec, pmplots)

3:30 pm - 4:00 pm Live demo example with yspec and pmplots (Sam)

4:00 pm - 4:30 pm Model management with bbr (Tim)

4:30 pm Q&A

Introductions

Tim Waterhouse, PhD

Group Leader Statistics, Principal Scientist II

Sam Callisto, PhD

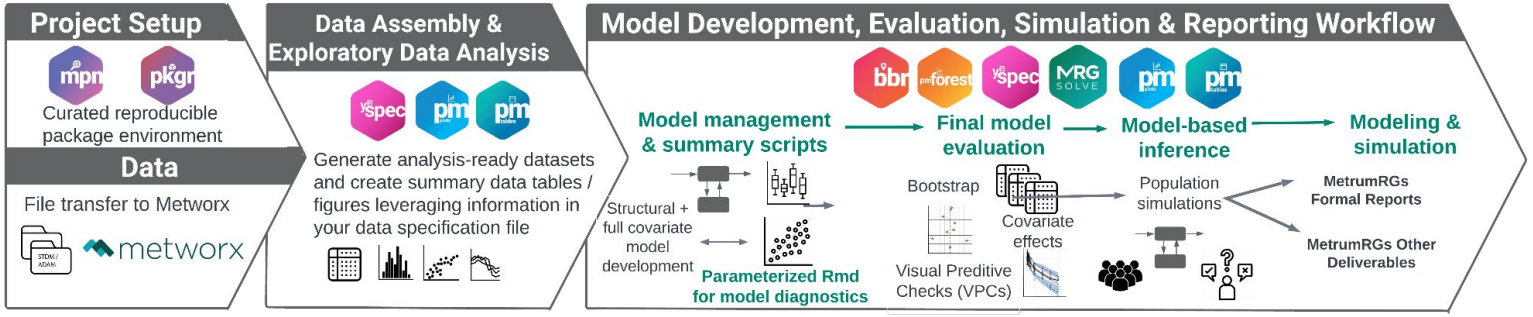
Senior Scientist I, PKPD

MeRGE

Metrum **R**esearch **G**roup **E**cosystem

METRUM RESEARCH GROUP

We introduce MeRGE through a **user-friendly** Expo that showcases a suite of tools in the context of a simplified population pharmacokinetic (PPK) model. It demonstrates how to proceed step-by-step through a PPK modeling and simulation (M&S) analysis, using the same process and suite of tools that we use at Metrum Research Group, to ensure traceable and reproducible pharmacometrics research.



Metworx is a secure, highly-scalable, cloud-native Platform-as-a-Service that brings reproducible tools and computing to scientific teams of all sizes.

metworx

Value	How	Metworx 4.0 *coming enhancements*
Scientific Excellence Built-In	• MIDD at its core • Designed, maintained and guided by the scientific excellence of MRG • Comes with industry-leading tools and technologies	• Inclusion of best-practice examples via MeRGE
Reproducible / Traceable	• Rapid validation • Consistent, controllable state of compute environments • MPN: Immutable snapshots of packages and dependencies for long-term reproducibility	• Tighter integration with MPN • Inclusion of best-practice examples via MeRGE • Enhanced audit trails
Scaleable	• No shared clusters • Each workflow is its own scalable grid • Allows multiple workflows per user • Fast ramp-up across large, distributed teams	• Improved visibility and cost-efficiency of cloud resources • More controls across large groups with different usage needs
Security	• Secure data and compute isolation • Client-controlled permissions • SSO integration	• Enhanced user/group administration

MeRGE Main Website:

https://merge.metrumrg.com/

Example Scripts and Slides Presented Today:

https://github.com/metrumresearchgroup/iu-ctsi-2023-merge

https://bit.ly/ctsi-merge

Why MeRGE?

Support traceable, reproducible, and scalable pharmacometric analyses

Example 1: working on a project with a team … consistency, efficiency

For example, tables can be VERY time-consuming to make, especially in a traceable manner. The look and content of tables can vary considerably when made by different individuals, or even by the same person at different times!

yspec + pmtables makes this MUCH easier!!

Same goes for figures:
yspec + pmplots + mrgsolve + pmforest also makes this MUCH easier!!

Why MeRGE?

Support traceable, reproducible, and scalable pharmacometric analyses

Example 2: working with stakeholders, I'd like to give them an update … consistency, expectations, ease of communications

"Hey Matt, explain to me why you chose a two vs a one compartmental model? And what about that variance structure, are we certain that we have appropriate random effects for IIV and residual variability?"

bbr + yspec + pmtable + pmplots + Rmarkdown makes this easy!!

METRUM
RESEARCH GROUP

Why MeRGE?

Support traceable, reproducible, and scalable pharmacometric analyses

Example 3a: work done 3 years ago, new reg submission and we need to recreate (or update) an analysis

"Hey Kyle, remember that empagliflozin work we did a few years ago [1]; we need to some more work to bridge into T1DM [2]…"

mpn + pkgr + yspec + bbr + pmtable + pmplots + Rmarkdown (would have) made this easy (ier)!!

[1] https://link.springer.com/article/10.1007/s13300-016-0174-y
[2] https://accp1.onlinelibrary.wiley.com/doi/abs/10.1002/jcph.1051

Why MeRGE?

Support traceable, reproducible, and scalable pharmacometric analyses

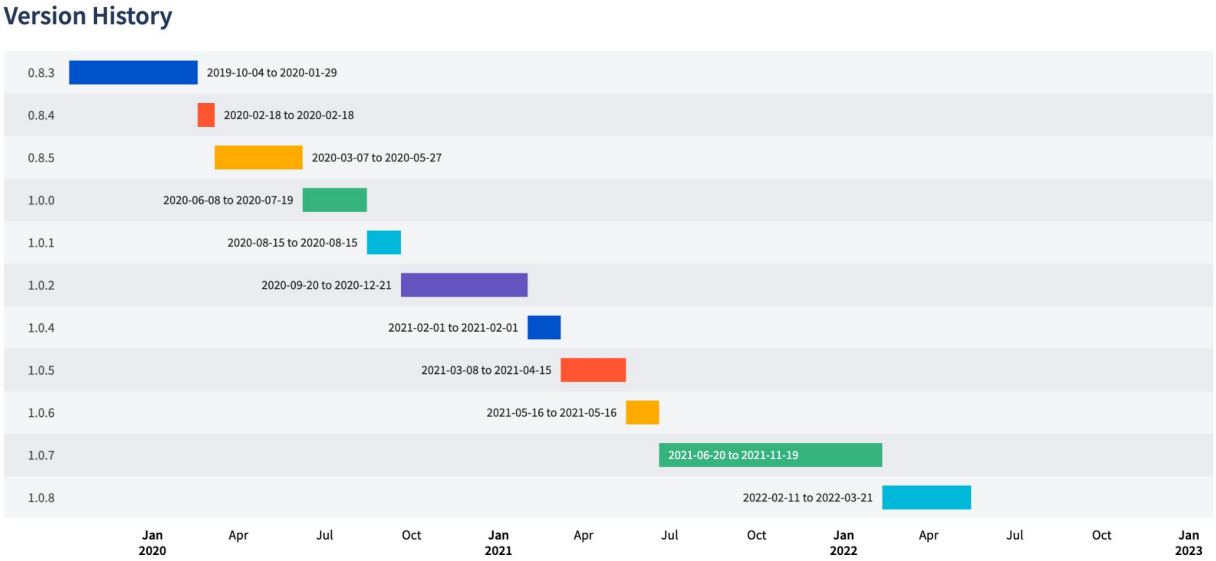
Example 3b: work done 3 years ago, the people that did the work are not available, we need to track down what they did to continue on for a new indication…

“Hey Curtis, guess what, we need you to do some work on the empagliflozin program…”

Reproducibility emphasis: MPN

MPN is a frequently-updated public repository of R packages which allows users to access current and previous package versions

- Snapshots taken approximately every 2 months.
- Allows you to easily install a package version along with all dependencies time-locked to a calendar date



Example of MPN history of the dplyr package

METRUM RESEARCH GROUP

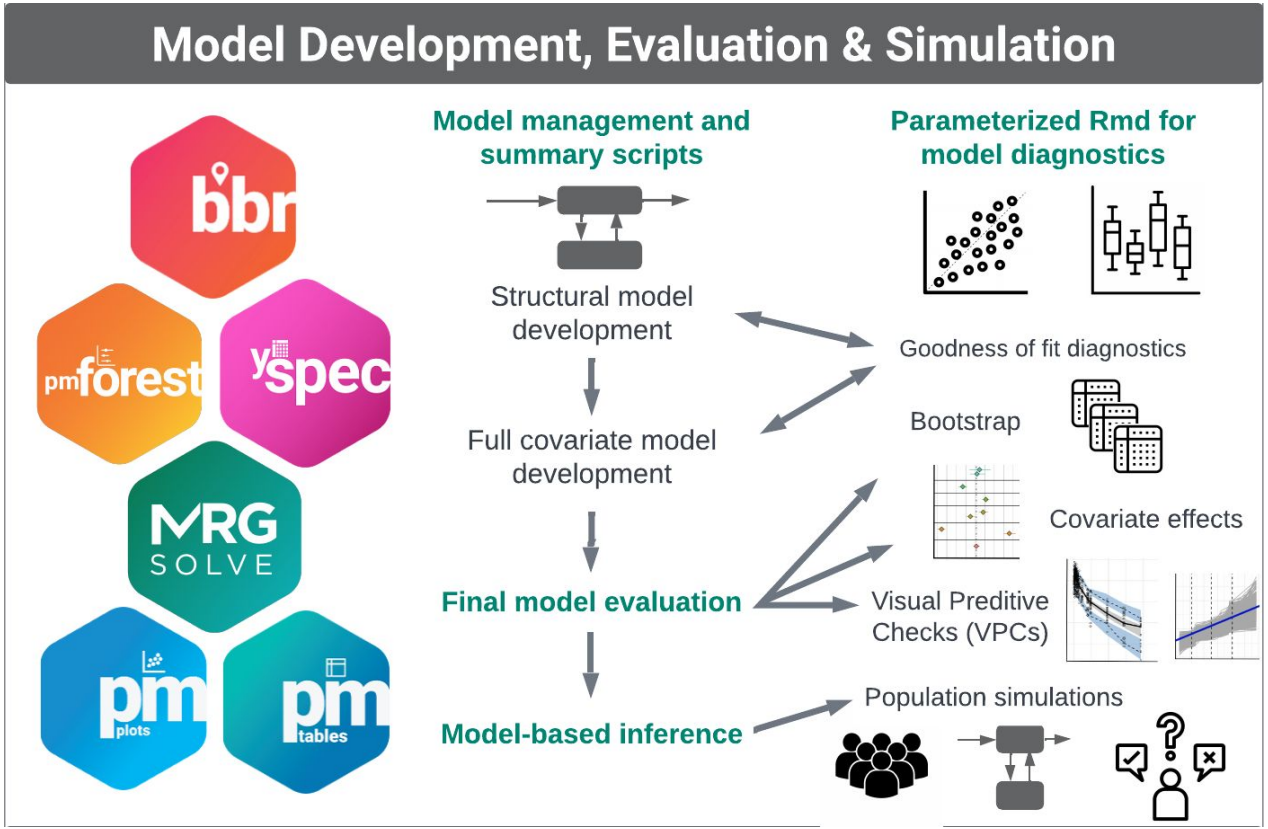
Reproducibility emphasis: pkgr


```yaml
Version: 1
# top level packages
Packages:
  - rmarkdown
  - bitops
  - caTools
  - knitr
  - tidyverse
  - shiny
  - logrrr

# any repositories, order matters
Repos:
  - MPN: "https://mpn.metworx.com/snapshots/stable/2020-09-20"
  - CRAN: "https://cran.rstudio.com"

# path to install packages to
Library: "<path/to/install/library>"

# package specific customizations
Customizations:
  Packages:
    - tidyverse:
        Suggests: true
```

pkgr is a terminal-based tool which simplifies R package installation using yaml files

- Multiple repositories can be specified
- Package-specific modifications can be applied (e.g. install all Suggests)
- Threaded processes for quick installation

A pop PK workflow using NONMEM



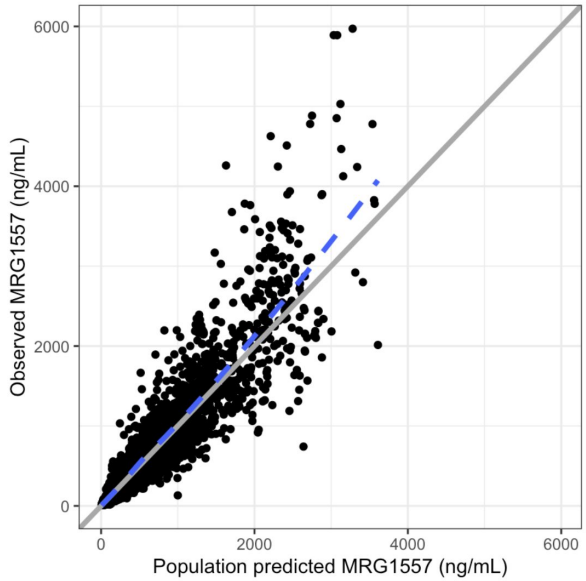
METRUM RESEARCH GROUP

Overview of R Packages

Introduction to pmplots

- Standardized plots
  - Exploratory
  - Diagnostic
- Simple / efficient syntax
- Expects standard inputs
  - TIME
  - DV
  - PRED
  - IPRED
  - CWRES
- Batch processing
- "Enough" customization
- Not a new grammar of graphics

```
dv_pred(df, yname = .yname)
```



Introduction to pmplots

Conditional weighted residuals versus time

```
p1 <- cwres_time(data)
```

Residuals versus population predicted value

```
p2 <- res_pred(data)
```

NPDE boxplots in each study

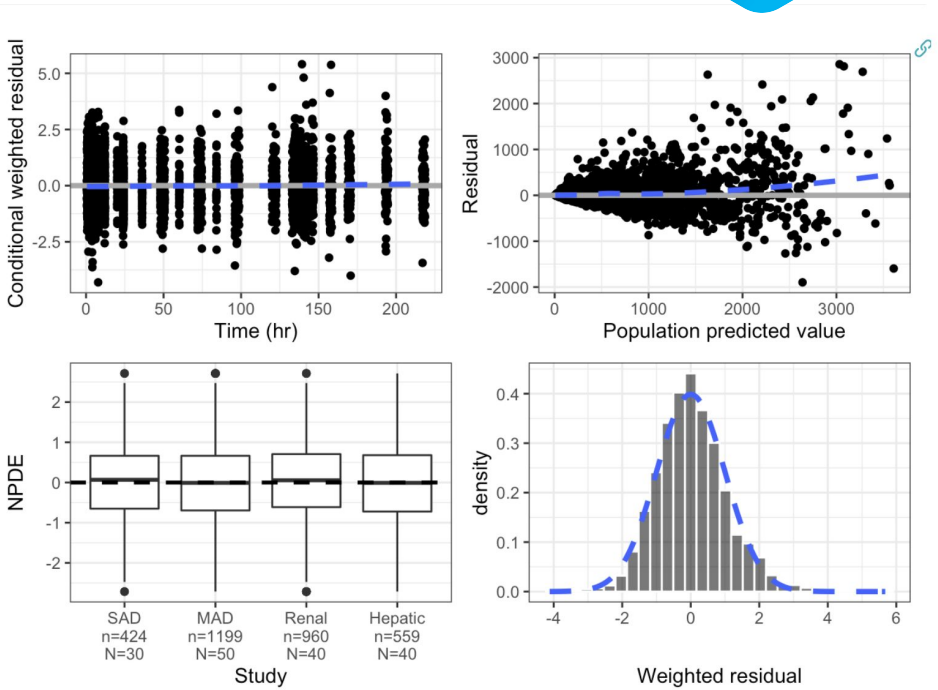
```
p3 <- npde_cat(data, x = "STUDYc//Study")
```

Histogram of weighted residuals

```
p4 <- wres_hist(data)
```

With output

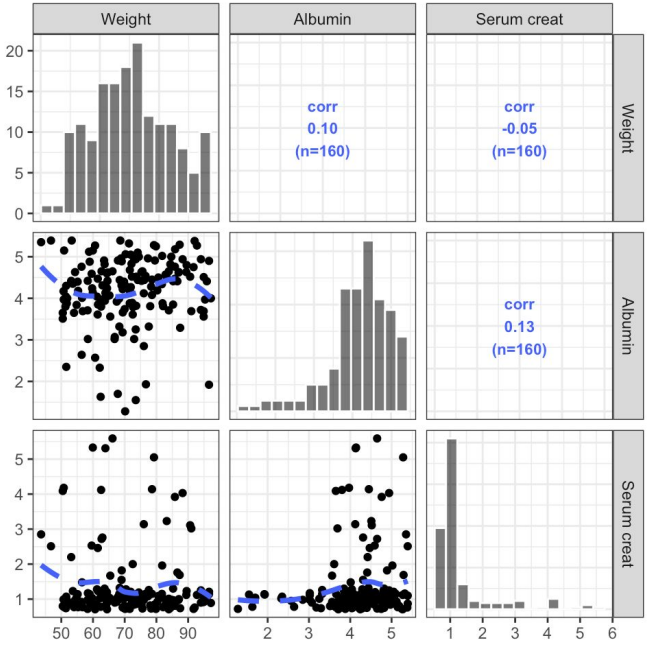
```
(p1+p2)/(p3+p4)
```



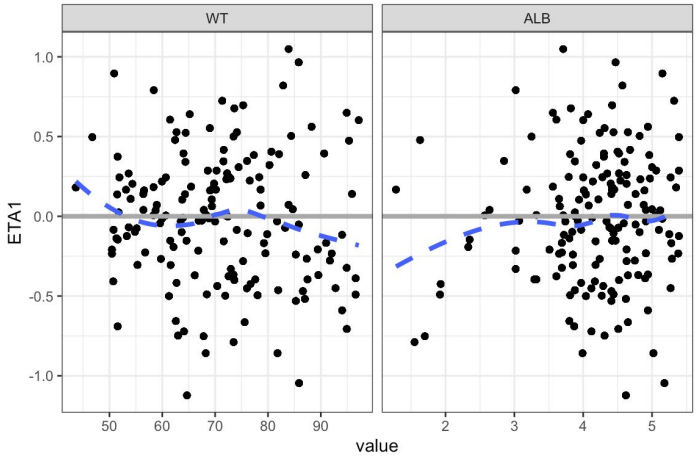
METRUM RESEARCH GROUP

Introduction to pmplots


```
cols <- c("WT//Weight", "ALB//Albumin", "SCR//Serum creat")
pairs_plot(id, cols)
```

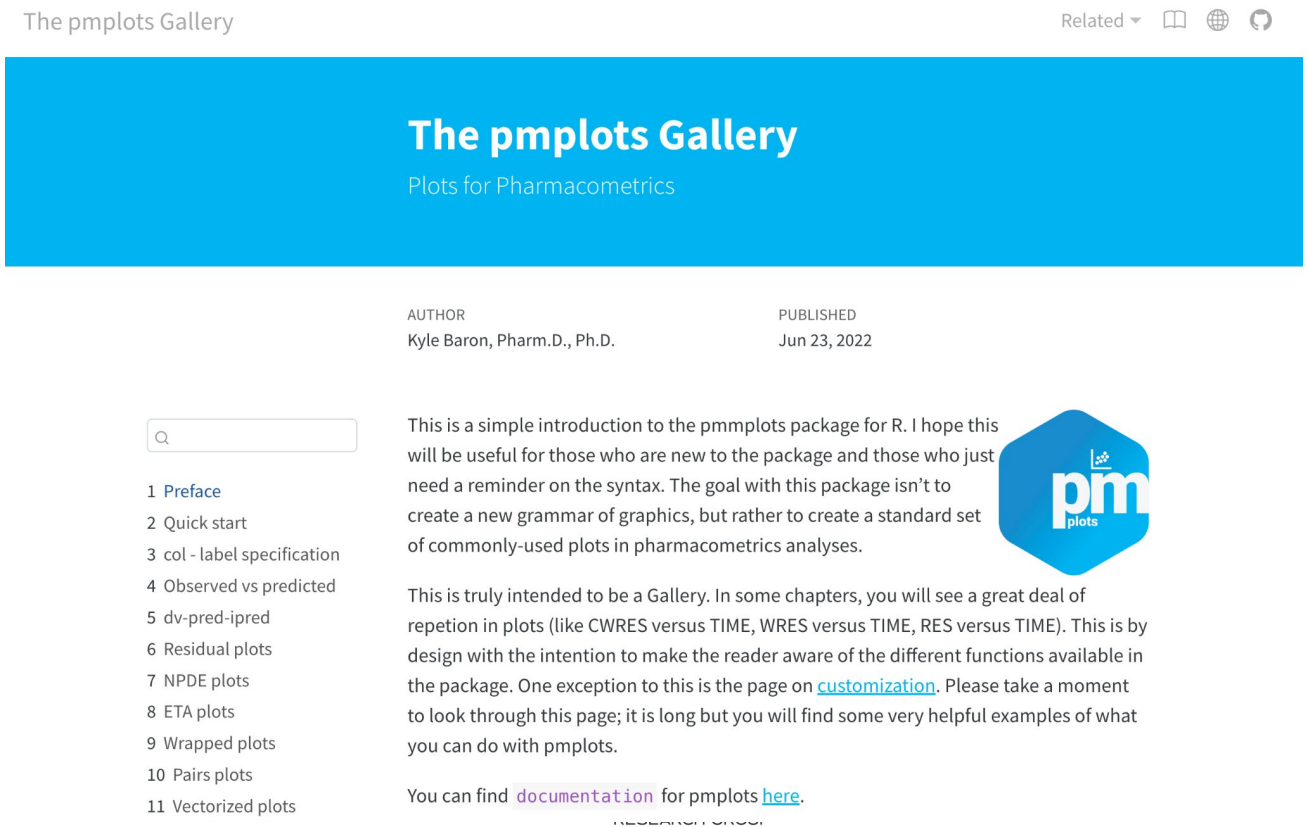



```
wrap_eta_cont(
  df,
  y = "ETA1",
  x = c("WT", "ALB"),
  scales = "free_x"
)
```



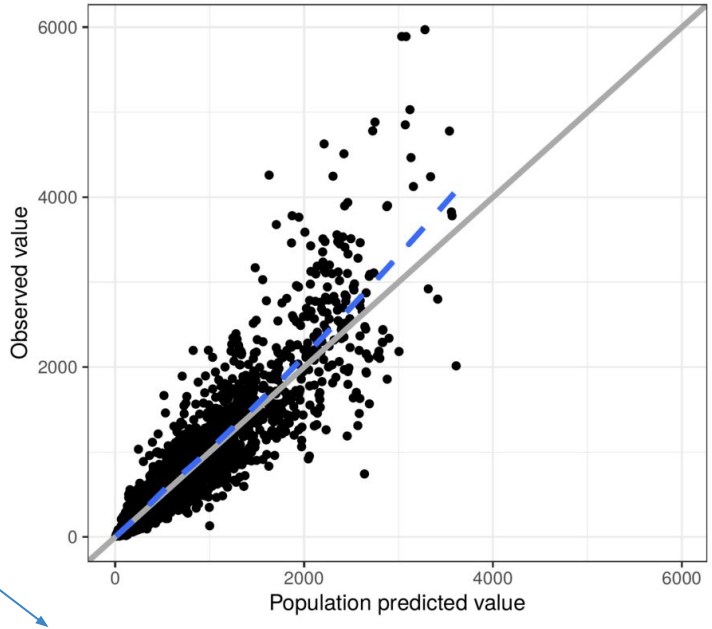
pmplots gallery

https://metrumresearchgroup.github.io/pmp-book/



mrggsave - save annotated images


```
mrggsave(p, stem = "intro-1", dir = tempdir(), script = "mrggsave.Rmd")
```



Source code: mrggsave.Rmd
Source graphic: intro-1.pdf

- Annotation
  - Source code file name
  - Image output file name
- Save lists of plots
- Interpolate variables into file names
- Save to multiple devices
  - pdf, png, both …
- Set height and width with sensible defaults

mrggsave - save lists of plots


```
run <- 101

p <- list(
  dv_pred(data),
  npde_time(data),
  cwres_hist(data)
)

ans <- mrggsave(p, stem = "diagnostics-{run}-", dev = "png")

basename(ans)
```

```
[1] "diagnostics-101-001.png" "diagnostics-101-002.png"
[3] "diagnostics-101-003.png"
```

mrggsave - save lists of plots


```
run <- 101

p <- list(
  dv_pred(
  npde_tim
  cwres_hi
)

ans <- mrg

basename(a
```

```
[1] "diagn
[3] "diagn
```

```
p <- list(
  `dv-v-pred`       = dv_pred(data),
  `npde-v-time`     = npde_time(data),
  `cwres-histogram` = cwres_hist(data)
)

ans <- mrggsave(p, tag = run, dev = "png", use_names = TRUE)

basename(ans)
```

```
[1] "dv-v-pred-101.png"       "npde-v-time-101.png"
[3] "cwres-histogram-101.png"
```

mrggsave - save lists of plots


```
run <- 101

p <- list(
  dv_pred(
  npde_tim
  cwres_hi
)

ans <- mrg

basename(a
```

```
[1] "diagn
[3] "diagn
```

```
p <- list(
  `dv-v-pr
  `npde-v-
  `cwres-h
)

ans <- mrg

basename(a
```

```
[1] "dv-v-p
[3] "cwres-
```

```
p <- named_plots(
  dv_pred(data),
  npde_time(data),
  cwres_hist(data)
)

ans <- mrggsave(p, tag = run, dev = "png")

basename(ans)
```

```
[1] "dv-pred-101.png"    "npde-time-101.png"  "cwres-hist-101.png"
```

mrggsave - save lists of plots


```
run <- 101

p <- list(
  dv_pred
  npde_tim
  cwres_hi
)

ans <- mrg

basename(a
```

```
[1] "diagn
[3] "diagn
```

```
p <- list(
  `dv-v-pr
  `npde-v-
  `cwres-h
)

ans <- mrg

basename(a
```

```
[1] "dv-v-p
[3] "cwres-
```

```
p <- named
  dv_pred
  npde_tim
  cwres_hi
)

ans <- mrg

basename(a
```

```
[1] "dv-pr
```

```
dv_versus_pred <- dv_pred(data)

p <- named_plots(dv_versus_pred)

ans <- mrggsave(p, tag = run, dev = "png", use_names = TRUE)

basename(ans)
```

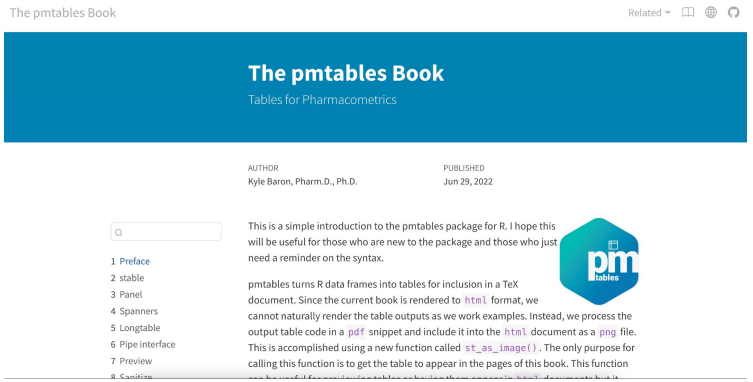
```
[1] "dv-versus-pred-101.png"
```

pmtables - tables for latex

Statistic	Study				
	12-DEMO-001 n = 30	12-DEMO-002 n = 50	11-DEMO-005 n = 40	13-DEMO-001 n = 40	Summary n = 160
Weight					
Mean (SD)	72.2 (14.3)	72.4 (11.5)	68.9 (14.5)	69.4 (11.6)	70.7 (12.8)
Min / Max	50.9 / 97.2	51.5 / 96.6	43.6 / 92.8	50.7 / 96.6	43.6 / 97.2
Missing	1	1	1	0	3
CRCL					
Mean (SD)	106 (9.46)	103 (8.35)	58.8 (29.7)	102 (8.19)	92.1 (25.5)
Min / Max	93.2 / 126	90.6 / 121	15.4 / 103	90.7 / 119	15.4 / 126
Missing	1	1	1	3	6
Sex					
male	10 (33.3)	18 (36.0)	29 (72.5)	23 (57.5)	80 (50.0)
female	20 (66.7)	32 (64.0)	11 (27.5)	17 (42.5)	80 (50.0)
Formulation					
tablet	25 (83.3)	42 (84.0)	30 (75.0)	33 (82.5)	130 (81.2)
capsule	3 (10.0)	6 (12.0)	3 (7.5)	3 (7.5)	15 (9.4)
troche	2 (6.7)	2 (4.0)	7 (17.5)	4 (10.0)	15 (9.4)

Categorical summary is count (percent)
n: number of records summarized
SD: standard deviation
Min: minimum; Max: maximum
Source code: _snippets.Rmd

```
pt_demographics(
  pmt_first,
  cols_cont = c(Weight = "WT", "CRCL"),
  cols_cat = c(Sex = "SEXf", Formulation = "FORM"),
  span = c(Study = "STUDYf")
) %>% stable() %>% st_as_image()
```



Introduction to yspec

- Documentation of analysis data sets
  - Write definitions in yaml format
  - Load into R as object
- Use along all phases of project work
  - Interactive query during DA
  - Generate define.pdf
  - Annotate plots and tables
  - Generate table for report

1 Datasets

Description	Location
Example PopPK analysis data set	analysis3.xpt

1.1 Example PopPK analysis data set (`analysis3.xpt`)

VARIABLE	LABEL	TYPE	CODES
C	comment character	character	C = comment, . = non-comment
NUM	record number	numeric	
ID	subject identifier	numeric	
TIME	time after first dose (unit: hour)	numeric	
SEQ	data type	numeric	0 = dose, 1 = observation
CMT	compartment number	numeric	
EVID	event ID	numeric	0 = observation, 1 = dose
AMT	dose amount (unit: mg)	numeric	
DV	dependent variable	numeric	
AGE	age (unit: years)	numeric	
WT	weight (unit: kg)	numeric	
HT	height (unit: cm)	numeric	
EGFR	estimated glomerular filtration rate (unit: mL/min/1.73m^2)	numeric	
ALB	albumin (unit: g/dL)	numeric	
BMI	BMI (unit: kg/m^2)	numeric	
SEX	SEX	numeric	0 = male, 1 = female

METRUM RESEARCH GROUP

Coding data definitions in yaml format

Column name

```yaml
WT:
  short: weight
  unit: kg
  range: [40, 140]

ARM:
  short: treatment arm
  type: character
  values: [200 mg qd, 400 mg qd]
  make_factor: true

FORM:
  short: formulation
  values: [0, 1]
  decode: [tablet, capsule]

STUDY:
  short: study number
  type: numeric
  values: [101, 102, 201]
  decode:
    - DRGX-55-101
    - DRGX-55-102
    - DRGX-66-201
```

Continuous data item

Categorical data items

METRUM RESEARCH GROUP

Using a project-wide lookup file

Lookup file
(all definitions used on the project)

```yaml
WT:
  short: weight
  unit: kg
  range: [40, 140]

ARM:
  short: treatment arm
  type: character
  values: [200 mg qd, 400 mg qd]
  make_factor: true

FORM:
  short: formulation
  values: [0, 1]
  decode: [tablet, capsule]

STUDY:
  short: study number
  type: numeric
  values: [101, 102, 201]
  decode:
    - DRGX-55-101
    - DRGX-55-102
    - DRGX-66-201
```

In the PK file

```yaml
ARM: !look
STUDY: !look
FORM: !look
DV:
  short: concentration
  unit: ng/mL
```

In the AE file

```yaml
ARM: !look
STUDY: !look
DV:
  short: grade 4 thrombocytopenia
  values: {no: 0, yes: 1}
```

METRUM RESEARCH GROUP

Load, preview and validate

Load

```
data <- read_csv("my-data-file.csv")

spec <- ys_load("my-data-spec.yml")
```

Preview

```
head(spec)
```

```
   name info         unit              short        source
1     C  cd-            .  comment character ysdb_internal
2   NUM  ---            .       record number ysdb_internal
3    ID  ---            . subject identifier ysdb_internal
4  SUBJ  c--            . subject identifier ysdb_internal
5  TIME  ---         hour               TIME          look
6   SEQ  -d-            .                SEQ             .
7   CMT  ---            . compartment number ysdb_internal
8  EVID  -d-            .           event ID ysdb_internal
9   AMT  ---           mg        dose amount ysdb_internal
10   DV  --- micrograms/L dependent variable ysdb_internal
```

Validate

```
ys_check(data, spec, error_on_fail = FALSE)
```

```
Messages:
 - spec has more items than cols in the data
 - names in spec but not in data:
    - AAG

#--------------------------------------

[1] FALSE
```

Access data as list or through api

Query Continuous

```
spec$WT
```

```
name  value
col   WT
type  numeric
short weight
unit  kg
range 40 to 100
```

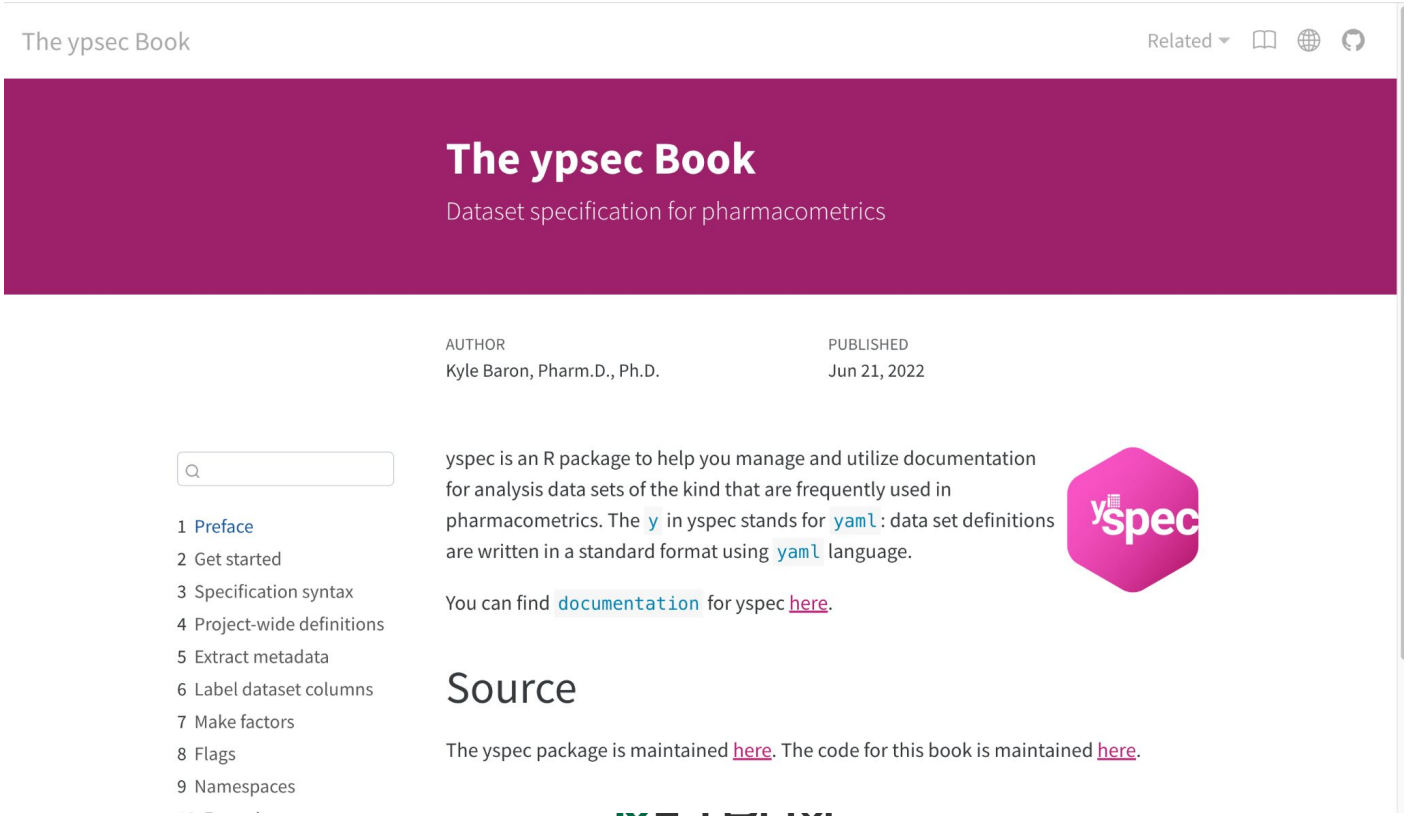
Query Categorical

```
spec$FORM
```

```
name  value
col   FORM
type  numeric
short formulation
value 0 : tablet
      1 : capsule
```

The yspec Book

https://metrumresearchgroup.github.io/ysp-book/



METRUM RESEARCH GROUP

lastdose - calculate time after dose


```
data
```

	ID	SUBJ	TIME	CMT	EVID	AMT	II	ADDL
1	1	1	0.00	1	1	5	6	23
2	1	1	0.61	2	0	0	0	0
3	1	1	1.15	2	0	0	0	0
4	1	1	1.73	2	0	0	0	0
5	1	1	2.15	2	0	0	0	0
6	1	1	3.19	2	0	0	0	0
7	1	1	4.21	2	0	0	0	0
8	1	1	5.09	2	0	0	0	0
9	1	1	6.22	2	0	0	0	0
10	1	1	8.09	2	0	0	0	0
11	1	1	12.03	2	0	0	0	0
12	1	1	20.07	2	0	0	0	0
13	1	1	24.20	2	0	0	0	0

```
lastdose(data)
```

	ID	SUBJ	TIME	CMT	EVID	AMT	II	ADDL	TAD	LDOS
1	1	1	0.00	1	1	5	6	23	0.00	5
2	1	1	0.61	2	0	0	0	0	0.61	5
3	1	1	1.15	2	0	0	0	0	1.15	5
4	1	1	1.73	2	0	0	0	0	1.73	5
5	1	1	2.15	2	0	0	0	0	2.15	5
6	1	1	3.19	2	0	0	0	0	3.19	5
7	1	1	4.21	2	0	0	0	0	4.21	5
8	1	1	5.09	2	0	0	0	0	5.09	5
9	1	1	6.22	2	0	0	0	0	0.22	5
10	1	1	8.09	2	0	0	0	0	2.09	5
11	1	1	12.03	2	0	0	0	0	0.03	5
12	1	1	20.07	2	0	0	0	0	2.07	5
13	1	1	24.20	2	0	0	0	0	0.20	5

https://github.com/metrumresearchgroup/lastdose

Live Demo Material

Model Diagnostics - Parameterized Reports

Report diagnostics

Purpose

To produce a set of diagnostic plots that will be included in the report. Please note that these plots are just meant to provide an example of what could be created and how. They are not an exhaustive list of every possible plot and were chosen with the project aims in mind.

While this *should* give users examples of plots generated with the most up-to-date packages and methods, we're always happy to have feedback. If you know of more efficient methods or want to suggest alternative ways of plotting the figures please open an issue with the details.

Set up

Model location

Define `modelName` and path to the model directory (`MODEL_DIR`).

Figure location

If saving figures out to pdf, define where those pdfs should be saved to. Here the figures are saved to `deliv > figure > model_run_number`

Model Diagnostics - Parameterized Reports

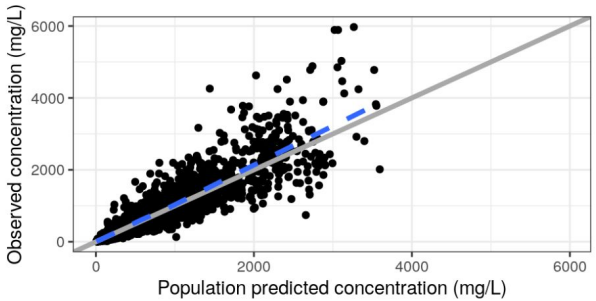
General diagnostic plots

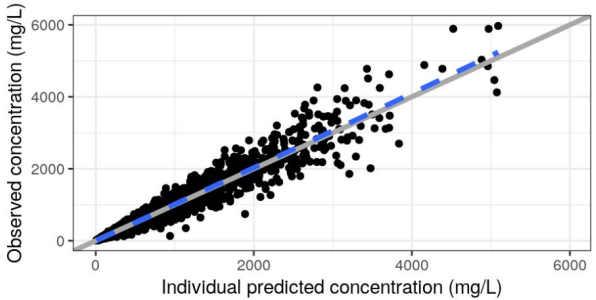
The following plots assume that the preferred x-axis labels are defined here.

DV vs PRED and IPRED

Create plots of DV vs PRED and IPRED for the full dataset and stratified by renal function and hepatic function.

```
## [1] "DV vs PRED and IPRED"
```





Model Diagnostics - Spec file

- Read in your spec file

```
spec <- ys_load(here("data","spec","analysis3.yml"))
head(spec)
```

```
  name info  unit                 short        source
1    C  cd-     .     comment character             .
2  NUM  ---     .         record number ysdb_internal
3   ID  ---     .     subject identifier ysdb_internal
4 TIME  ---  hour time after first dose             .
5  SEQ  -d-     .              data type             .
6  CMT                 compartment number ysdb_internal
```

- Change the namespace

```
spec <- ys_namespace(spec, "plot")
```

- Use the spec flags

```
diagContCov <- pull_meta(spec, "flags")$diagContCov
diagCatCov <- pull_meta(spec, "flags")$diagCatCov
```

Model Diagnostics - Data

- Read in your model output
  - `read_model`
  - `model_summary`

```
mod <- read_model(here("model","pk","106"))
sum <- mod %>% model_summary()
```

- Read in your data
  - `nm_join` - to join your NONMEM tables with the original dataset
  - `filter` to the observation records
  - `yspec_add_factors` to decode categorical covariates

```
data0 <- nm_join(mod)
```

```
data <-
  data0 %>%
  filter(EVID==0) %>%
  yspec_add_factors(spec, .suffix = "")
```


Model Diagnostics

NPDE vs continuous covariates plot

- Get covariates of interest from the spec file and make a list of axis labels
  - `pull_meta` to pull in information about the flags and select the appropriate flag
  - `ys_select` those covariates
  - `axis_col_labs` will convert the selected covariates to column axis labels

```
diagContCov <- pull_meta(spec, "flags")$diagContCov

NPDEco <-
  spec %>%
  ys_select(all_of(diagContCov)) %>%
  axis_col_labs(title_case = TRUE, short_max = 10) %>%
  as.list()

NPDEco
```

```
$AGE
[1] "AGE//Age (years)"

$WT
[1] "WT//Weight (kg)"

$ALB
[1] "ALB//Albumin (g/dL)"

$EGFR
[1] "EGFR//EGFR (mL/min/1.73m2)"
```

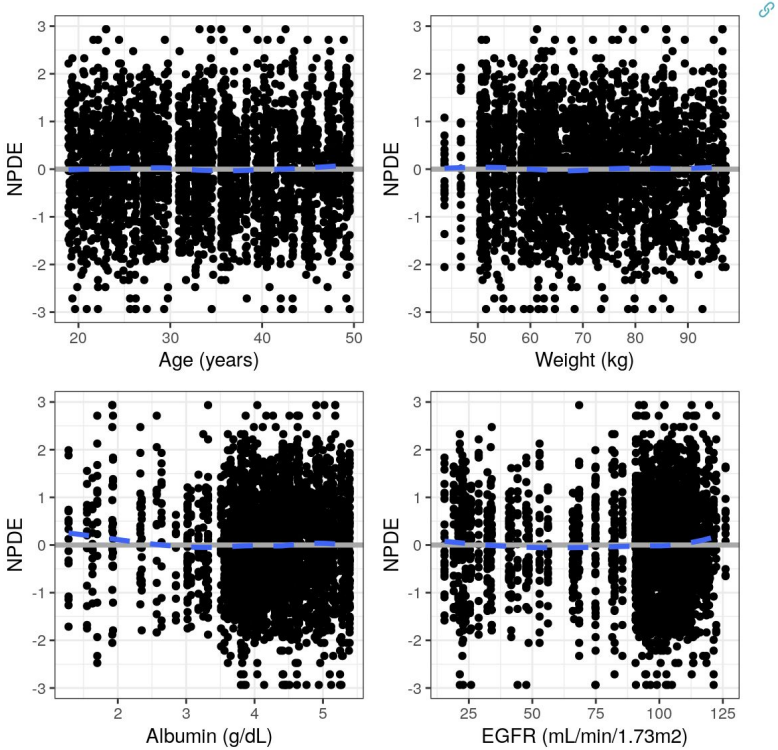
Model Diagnostics

NPDE vs continuous covariates plot

- Get covariates of interest from the spec file and make a list of axis labels

```
pList <- purrr::map(NPDEco, ~ npde_cont(data, x = .x))

pm_grid(pList, ncol = 2)
```

- `map` across the covariate list to create all plots using `npde_cont`
- `pm_grid` to display all plots in a grid



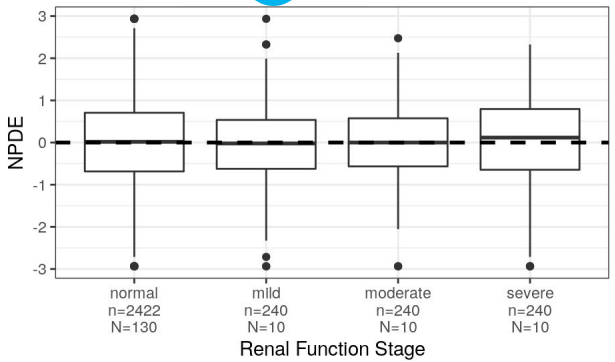
METRUM RESEARCH GROUP

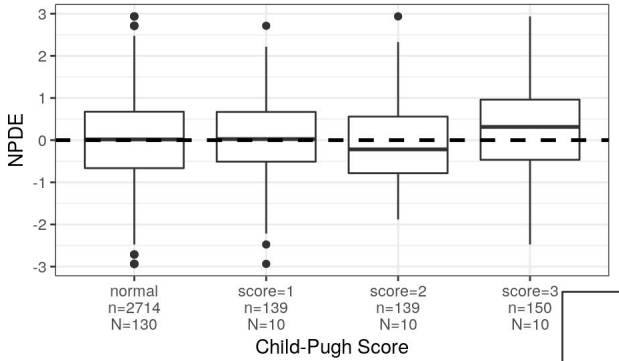
Model Diagnostics

NPDE vs categorical covariates plot

- Use similar methods to create NPDE plots for categorical covariates

```
NPDEca <-
  spec %>%
  ys_select("RF", "CP") %>%
  axis_col_labs(title_case = TRUE, short_max = 20) %>%
  as.list()
pList_cat = purrr::map(NPDEca, ~ npde_cat(data, x = .x))
pm_grid(pList_cat, ncol=1)
```





Hands-on

Model Diagnostics

- The ETA based plots require a dataset filtered to one record per subject

```
id <- distinct(data, ID, .keep_all=TRUE)
```

- pmplots package has series of ETA plot functions
  - `eta_pairs` correlation and distribution of model ETAs
  - `eta_cont` ETA vs continuous covariates
  - `eta_cat` ETA vs categorical covariates
- Leverage the information in the spec object in several ways:
  - Extract covariates of interest from the spec flags with `pull_meta` and `ys_select`
  - Axis labels are renamed with the short label in the spec using `axis_col_labs`
  - Numerical categorical covariates are decoded with the `yspec_add_factors` function

Model Diagnostics

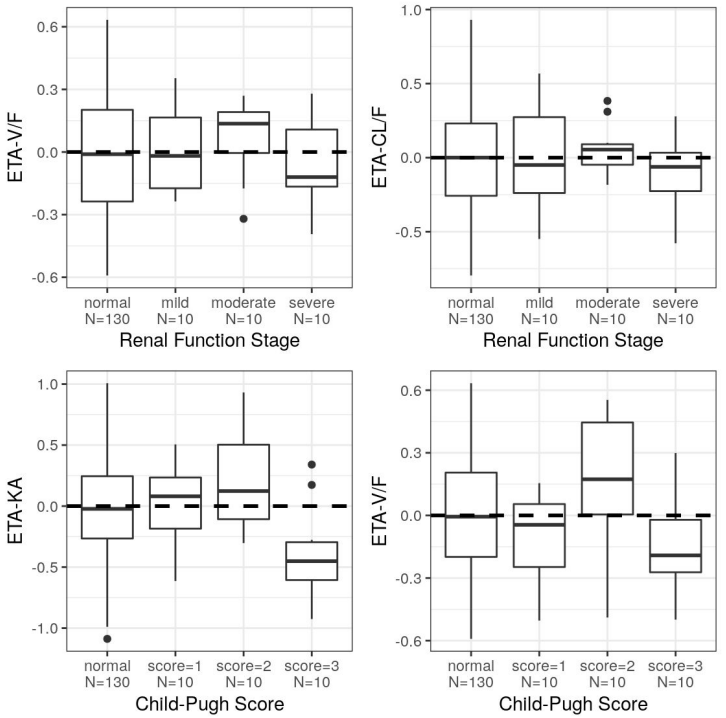
ETA vs categorical covariates plot

- Define the ETAs of interest

```
etas <- c("ETA1//ETA-KA", "ETA2//ETA-V/F", "ETA3//ETA-CL/F")
```

- Get the covariates from the spec file and use the `eta_cat` function to create a list of plots for each ETA and covariate pairing

```
ca <-
  spec %>%
  ys_select(diagCatCov) %>%
  axis_col_labs(title_case=TRUE, short_max = 20)
p <- eta_cat(id, ca, etas)
pRenal <- (p[[5]] + p[[6]]) / (p[[7]] + p[[8]])
pRenal
```

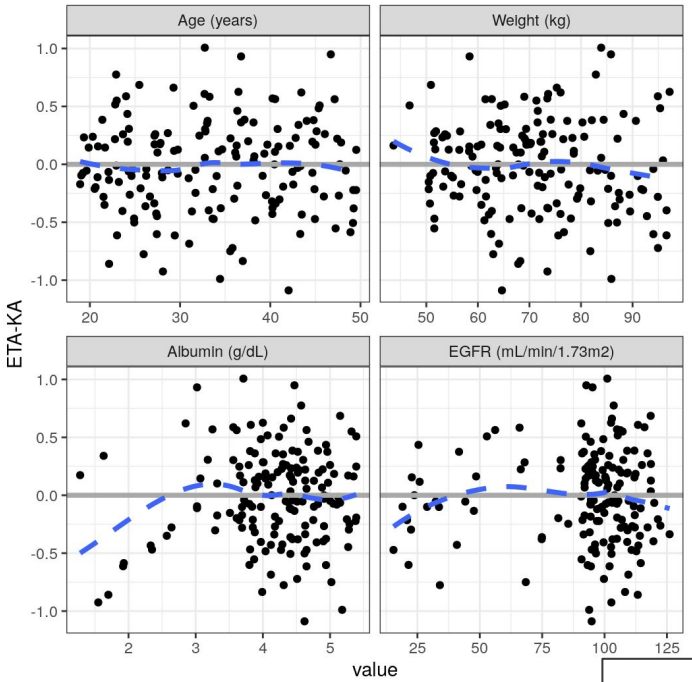


Model Diagnostics

ETA vs continuous covariates plot

- `wrap_eta_cont` makes an ETA plot faceted by continuous covariates
- `map` over the ETAs to create multiple plots

```
map_wrap_eta_cont = function(.id, .co, .etas){
  p <- wrap_eta_cont(.id,
                     x = .co, y = .etas,
                     use_labels = TRUE,
                     scales= "free_x")
}

p = purrr::map(.x = etas, ~ map_wrap_eta_cont(id, contCo, .x))
p[[1]]
```



METRUM RESEARCH GROUP

Hands-on

Reporting templates using Rmarkdown

Title for the page

Output to an html file with a floating table of contents

Parameters that can be updated each time the Rmarkdown is rendered

```yaml
---
title: "Report diagnostics"
output:
  html_document:
    toc: true
    toc_float: true
    depth: 2
params:
  run: 102
  modelDir: "model/pk"
  script: "diagnostics-report.Rmd"
  yspec: "analysis3.yml"
  contCov: !r c("AGE","WT","ALB","EGFR")
  catCov: !r c("STUDY", "RF", "CP", "DOSE")
  etas: !r c("ETA1//ETA-KA", "ETA2//ETA-V/F", "ETA3//ETA-CL/F")
  drugNameUnits: "concentration (mg/L)"
  include_code: FALSE
  include_plots: TRUE
  run_mrggsave: TRUE
```

Rendering templates using R

- Helper function to render the templates (see backups for alternative)
  - Only need to define parameters that differ from the defaults provided in the template yaml section
  - Use our `model_diagnostics` helper function to render the plot and `browseURL` to pop open the html after creation

```
mod <- bbr::read_model(file.path(modelDir, 100))

mod %>%
  model_diagnostics(
    modelSpecifics,
    template = rmd_template
  )
```

```
model_diagnostics(
  file.path(modelDir, 102),
  modelSpecifics,
  template = rmd_template
) %>%
  browseURL()
```

Introduction to bbr

Using bbr for model development

bbr is an R package developed by MetrumRG. It serves three primary purposes:

- Submit NONMEM models, particularly for execution in parallel and/or on a high-performance compute (HPC) cluster (e.g. Metworx).
- Parse NONMEM outputs into R objects to facilitate model evaluation and diagnostics in R.
- Annotate the model development process for easier and more reliable traceability and reproducibility.

Walk though:

- Creating and submitting a model
- Iterative model development
- Preview of model evaluation and diagnostics
- Annotation of models with tags, notes, etc.

Follow along on the "Model Management" page and associated code.

* bbr developed in Linux; compatible with Mac OS; continued development in Windows

bbr: Creating and submitting a model

Creating a model object from a NONMEM control stream file:

```
# create the first model
mod100 <- new_model(file.path(MODEL_DIR, 100))
```

Submitting models for execution:

```
submit_model(mod100)
```

bbr: Creating and submitting a model

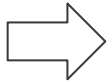
Creating a model object from a NONMEM control stream file:

```
# create the first model
mod100 <- new_model(file.path(MODEL_DIR, 100))
```

Submitting models for execution:

```
submit_model(mod100)
```

```
submit_model(
  mod,
  .bbi_args = list(
    overwrite = TRUE,
    parallel = TRUE,
    threads = 8
  )
)
```



These other arguments let you parallelize the run, too!

bbr: Running in parallel and/or on a grid

Models are run on Sun Grid Engine (SGE), unless otherwise specified:

```
proc <- submit_model(
  mod,
  .mode = "local",
  .wait = FALSE
)
```

Test the optimum number of threads for a model:

```
mods <- test_threads(mod, .threads = c(2, 4))
check_run_times(mods, .wait = FALSE)
```

- Currently runs on SGE; Slurm will be added by end of 2023

bbr: Iterative model development

Creating a new model based on an existing model:

```
mod101 <- copy_model_from(
    .parent_mod = mod100,
    .new_model = 101,
    .inherit_tags = TRUE
  )
```

This will copy an existing model (“100”) and make a new one (“101”). You can then edit and save 101.ctl accordingly.

Housekeeping:

- it will “remember” the lineage (you’ll see that later),
- and… can carry over tags.

bbr: Iterative model development

Once you've created a new model based on an existing model:

```
mod101 <- copy_model_from(
    .parent_mod = mod100,
    .new_model = 101,
    .inherit_tags = TRUE
  )
```

Compare that model to its "parent" model:

```
# shows the difference between control streams
model_diff(mod101)
```

bbr: Model evaluation and diagnostics

Parse NONMEM outputs into an R list object:

```
sum100 <- model_summary(mod100)
```

Create a simple tibble with parameter estimates:

```
# helper function to extract parameter table
sum100 %>% param_estimates()
```

bbr: Adding model annotation

Add notes to the model:

```
mod100 <- mod100 %>%
  add_notes("systematic bias, explore alternate compartme
```

Add tags to the model:

```
mod100 <- mod100 %>%
  add_tags(c(
    TAGS$one_compartment_absorption,
    TAGS$eta_cl,
    TAGS$eta_ka,
    TAGS$eta_v,
    TAGS$proportional_ruv
  ))
```

bbr: Leveraging model annotation

Create a “run log” table:

```
# create a run log and do some basic formatting
run_log(MODEL_DIR, .recurse = FALSE) %>%
  collapse_to_string(based_on, tags, notes) %>%
  select(run, based_on, description, tags, notes) %>% knitr::kable()
```

run	based_on	description	tags	notes
100	NA	NA	one-compartment + absorption, ETA-CL, ETA-KA, ETA-V, proportional RUV	systematic bias, explore alternate compartmental structure
101	100	NA	two-compartment + absorption, ETA-CL, ETA-KA, ETA-V2, proportional RUV	eta-V2 shows correlation with weight, consider adding allometric weight
102	101	Base Model	two-compartment + absorption, ETA-CL, ETA-KA, ETA-V2, CLWT-allo, V2WT-allo	Allometric scaling with weight reduces eta-V2 correlation with weight. Will consider additional RUV structures. Proportional

bbr: Joining model output with input data

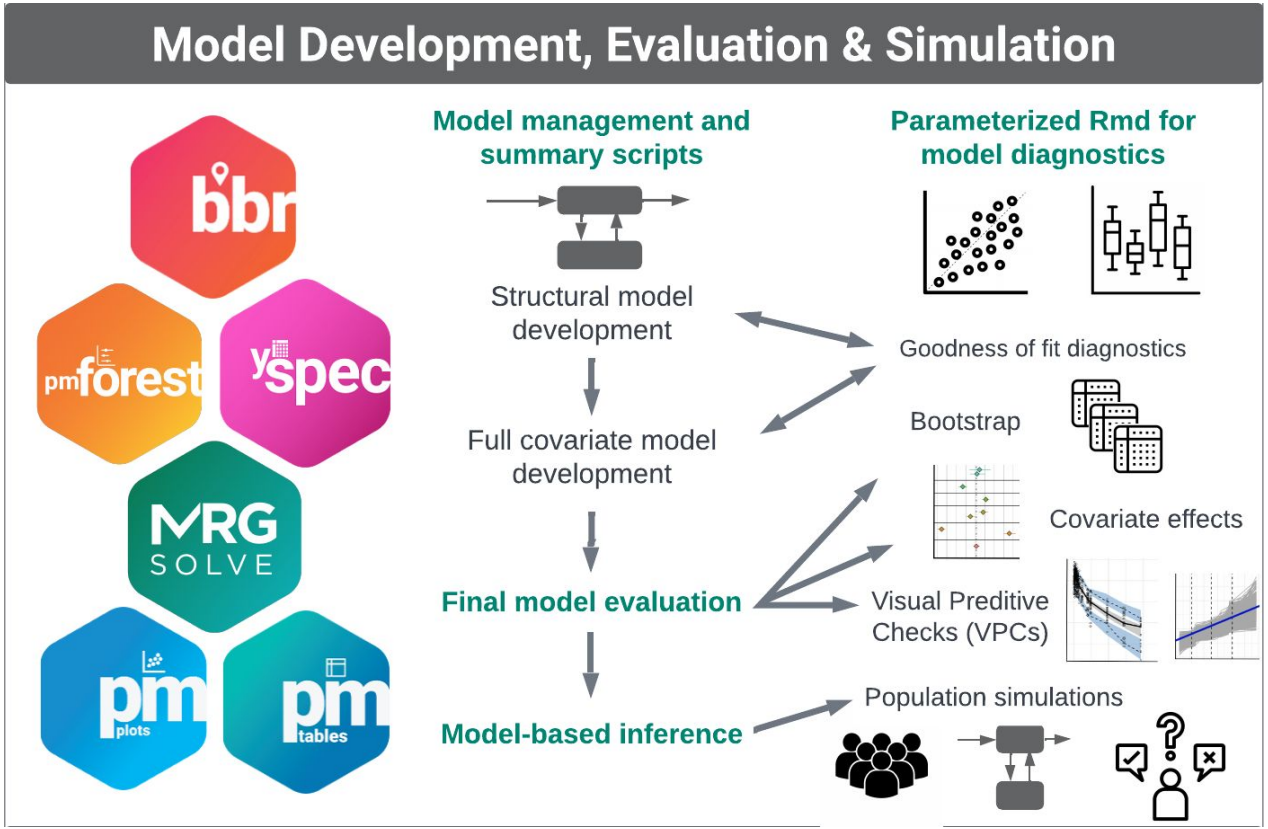
Join model output with input data:

```
join_df <- nm_join(mod1)
join_df %>%
  select(ID, TIME, DV, CL, V, NPDE, CWRES)
#> # A tibble: 779 × 7
#>       ID  TIME    DV    CL     V  NPDE CWRES
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1     1  0     1.22  43.4  463. 0.185 0.291
#>  2     1  0.25 12.6   43.4  463. 2.47  3.01
```

A single column uniquely identifying each row (e.g., NUM) means only one simple $TABLE is needed:

```
$TABLE NUM CL V1 V2 Q ETAS(1:LAST) IPRED NPDE NOPRINT ONEHEADER FILE=100.tab
```

A pop PK workflow using NONMEM



METRUM RESEARCH GROUP

Additional Resources

- MeRGE Expo 1 website:
  http://merge.metrumrg.com/expo/expo1-nonmem-foce/
- Package management: MPN and pkgr
  - https://kb.metworx.com/Users/Managing_R_Packages/r-package-management/
- VPCs using mrgsolve
  - https://merge.metrumrg.com/expo/expo1-nonmem-foce/posts/pk-vpc-final.html
- Right sizing workflow
  - https://kb.metworx.com/Users/Getting_Started/rightsizing-workflows/
  - https://metrumresearchgroup.github.io/bbr/articles/nonmem-parallel.html
- General bbr “cheat sheet”:
  https://metrumresearchgroup.github.io/cheatsheets/bbr_nonmem_cheat_sheet.pdf
- Today’s demo script and slide deck:
  https://github.com/metrumresearchgroup/iu-ctsi-2023-merge

metworx

Metworx is a proven scientific platform for quantitative analyses. It offers reproducible, traceable, and scalable cloud computing, accessible via a simple web browser. By combining per-user high performance computing (HPC) clusters, best-in-class security, and IT support, it truly delivers HPC for everyone.

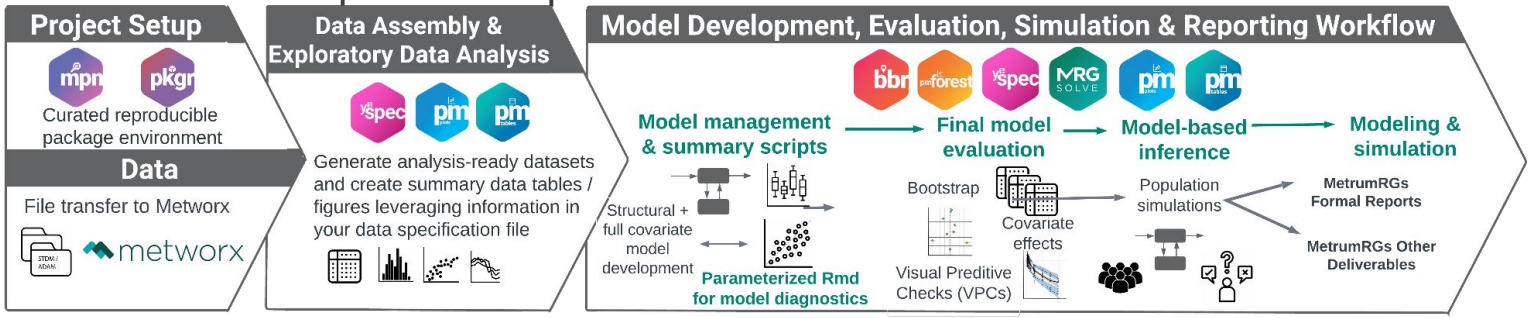
The Metworx platform ecosystem brings scientific and technological excellence together, enabling your team of scientists with expertise in the fields of drug development, statistics, data science, mathematics, pharmacology, pharmacometrics, and engineering to take control

MeRGE

Metrum Research Group Ecosystem

METRUM RESEARCH GROUP

We introduce MeRGE through a **user-friendly Expo** that showcases a suite of tools in the context of a simplified population pharmacokinetic (PPK) model. It demonstrates how to proceed step-by-step through a PPK modeling and simulation (M&S) analysis, using the same process and suite of tools that we use at Metrum Research Group, to ensure traceable and reproducible pharmacometrics research.



What you'll find in this Expo:

- Our approach to project set-up, data assembly, M&S activities, and reporting.
  Access to example code in a Github repository.
  Information and vignettes on MetrumRG's suite of tools.

Backups

Rendering templates using R

- Define the model specifics

```
modelSpecifics <- list(
  yspec = "analysis3.yml",
  contCov = c("AGE","WT","ALB","EGFR"),
  catCov = c("STUDY", "RF", "CP", "DOSE"),
  etas = c("ETA1//ETA-KA", "ETA2//ETA-V/F", "ETA3//ETA-CL/F"),
  include_code = TRUE,
  include_plots = TRUE,
  run_mrggsave = TRUE)
```

- Render the Rmd template

```
rmarkdown::render(
  here("script", "diagnostic-templates", "diagnostics-report.Rmd"),
  params = modelSpecifics,
  output_dir = here(modelDir, "100"),
  output_file = "diagnostic-report-100.html"
)
```

METRUM RESEARCH GROUP