PMDatR: A Pharmacometric Data Manipulation Package for R

Jason Chittenden, Jan Huisman, Kevin Dykstra

qPharmetra, Andover MA

Introduction:

The creation of NONMEM or similar datasets often involves interactions between multiple groups:

- ❑ a vendor that creates the raw datasets
- ❑ the pharmacometrician that specifies the desired content and format of the analysis data set
- ❑ data programmers who construct the dataset according to specifications
- ❑ quality control personnel who verify the dataset contents.

Tools that preserve the rigor of the traditional approach while reducing the cycle time and enabling pharmacometricians to take greater ownership of the final dataset can increase the efficiency of the overall modeling and simulation workflow.

Objective:

To create an R package that can be used from R scripts as well, as automated through a Graphical User Interface (GUI), enabling a flexible, customizable repeatable, templatized workflow.

Figure 1. PMDatR key functions and usage

Loading Data

- Specify input files
- Default load, or pass a function or reusable mapping file to customize

Functions for mapping data

- getDomain()
  - Pull columns from domains and preprocess
- getDV(), getIndividualDosing(), getCov(), getCovT()
  - Some required mappings (e.g. ID, Time).
  - Optional/additional mappings can be specified.
  - Automatic transposition of stacked covariates (e.g. LB, VS domains)

Functions for merging

- append.events(), append.CovT() – stack dosing, observations, and covariate data
- merge.Cov() – join in keyed covariates (e.g. matching ID and VISIT)
- post.merge.refactoring – do smart things with raw merged data
  - Elapsed time, record ID, … more
  - Custom transform, filter, and exclusion functions passed to it

Example: Loading EX Domain

Often we want to do the same types of operations to these domains. In this case we can load a mappings file and use the domain mappings when creating the domain. When column mappings are given in the mappings file, a function is created to process the data.

```
maps_yaml = load.domain.mappings("default_mappings.map.yaml")
EX = create.domain("EX", path=datadir, mappings=maps_yaml$EX)
```

The function that will be run as part of the domain load is shown here:

```
formatR::tidy_source(text=EX$fnPreProc)
```

```
## preprocess_EX <- function(dom) {
##     dom$Data = getDomain(dom$Data, EXDTC = iso_to_posix(EXSTDTC), EXSTDTC = iso_to_posix(EXSTDTC),
##         EXENDTC = iso_to_posix(EXENDTC), STUDYID = STUDYID, DOMAIN = DOMAIN,
##         USUBJID = USUBJID, EXSEQ = EXSEQ, EXTRT = EXTRT, EXDOSE = EXDOSE, EXDOSU = EXDOSU,
##         EXDOSFRM = EXDOSFRM, EXDOSFRQ = EXDOSFRQ, EXOCCUR = EXOCCUR, EXROUTE = EXROUTE,
##         VISITNUM = VISITNUM, VISIT = VISIT, EPOCH = EPOCH, EXSTDY = EXSTDY,
##         EXENDY = EXENDY, EXTPT = EXTPT, EXTPTNUM = EXTPTNUM, II = replace_values(EXDOSFRQ,
##             QD = 24, ONCE = 0, BID = 12, WEEKLY = 168))
##     dom
## }
```

The we can load the data and see that the columns were created/transformed:

```
EX = load.domain(EX)
```

```
##
## Domain loaded: EX
##  Source File: C:/Users/jason.chittenden.QPHARMETRA/Documents/R/win-library/3.3/PMDatR//tests/testdata/qp2/EX.sas7bdat
##  Dimensions: 20 rows x 21 columns
##  MD5 sum: d9d3e53d54a45f2be51bb7e06969bab2
##  Last Modified: 2018-04-04 11:11:31
```

Warning in validate.domain(x): Warning D16: Optional column mapping 'EXOCCUR=EXOCCUR' : ## names [EXOCCUR] are not found in Domain 'EX'

```
##  Pre-processed Data [20 x 21] at 2018-04-04 11:14:09
```

- The mappings file generates the loading function.
- An expected column is missing, we get a warning.

Example: Processing covariates

```
demog = DM$Data %>%
  getCov(ID = USUBJID, cov.keys = c("ID"), AGE = AGE, RACE = RACE,
         SEX = SEX, ETHNIC = ETHNIC, ACTARMCD = ACTARMCD) %>% # assign units
  convert_units_from_list(.ul = list(AGE = "years"))

vitals = VS$Data %>%
  getCov(ID = USUBJID, cov.val = VSSTRESN, cov.col = VSTESTCD,
         Units = VSSTRESU, cov.filter = VISIT == "SCREENING", cov.keys = c("ID"),
         WEIGHT = WEIGHT,HEIGHT = HEIGHT, BMI = BMI)

covs.l = list(demog, vitals)

demog %>% glimpse
```

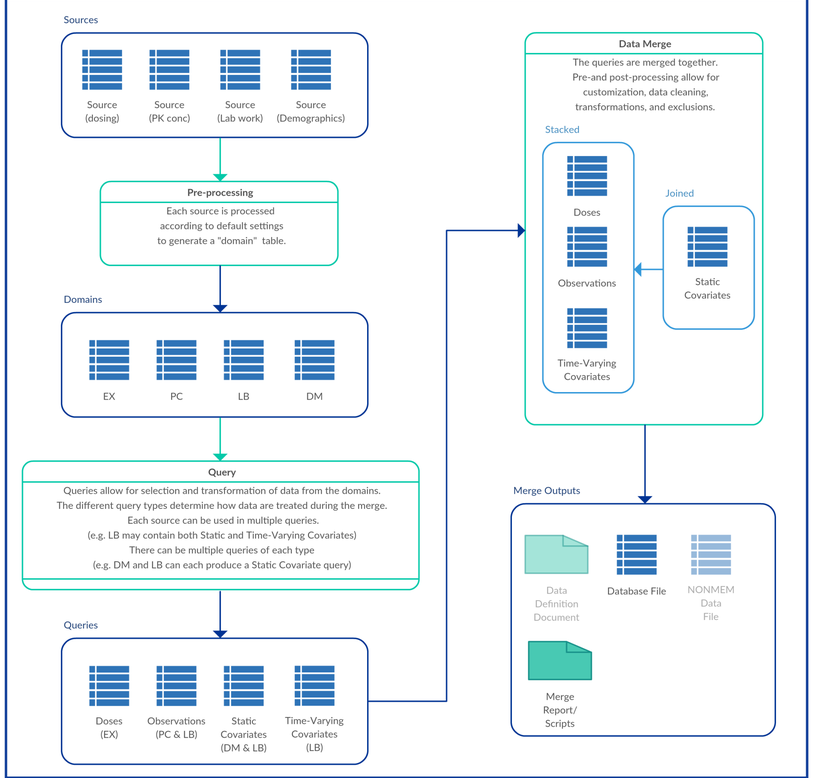
```
## Observations: 16
## Variables: 6
## $ ID       <chr> "ST1-1-01", "ST1-1-02", "ST1-1-03", "ST1-1-04", "ST1-...
## $ AGE      <years> 35, 33, 51, 29, 33, 50, 50, 38, 26, 46, 48, 51, 31,...
## $ RACE     <chr> "BLACK OR AFRICAN AMERICAN", "BLACK OR AFRICAN AMERIC...
## $ SEX      <chr> "M", "F", "F", "M", "M", "F", "M", "M", "M", "F", "F"...
## $ ETHNIC   <chr> "NOT HISPANIC OR LATINO", "NOT HISPANIC OR LATINO", "...
## $ ACTARMCD <chr> "PLACEBO", "PLACEBO", "PLACEBO", "PLACEBO", "PLACEBO"...
```

```
vitals %>% glimpse
```

```
## Observations: 16
## Variables: 4
## $ ID     <chr> "ST1-1-01", "ST1-1-02", "ST1-1-03", "ST1-1-04", "ST1-1-...
## $ WEIGHT <kg> 61, 67, 60, 78, 92, 65, 65, 82, 95, 79, 74, 59, 79, 94, ...
## $ HEIGHT <cm> 167, 152, 159, 161, 177, 151, 167, 175, 185, 166, 159, 1...
## $ BMI    <kg/m2> 21, 28, 23, 29, 28, 27, 22, 26, 27, 28, 28, 24, 22, 3...
```

- Note unstacking of codes, and units are associated with the data

Figure 2. Dataset creation process



Methods:

PMDatR builds on top of popular R packages such as dplyr[1] and tidyr[2], so much of the syntax is well known and documented. The key features that facilitate data work flows include:

- ❑ functions for transforming, reformatting (such as automatically unstacking covariate columns in result domains like LB), and verifying inputs
- ❑ common transformations such as filling forward
- ❑ computation of ADDL dosing
- ❑ unit aware columns and transformations;
- ❑ automated code generation from a settings file.

Some common transformations include:

- ❑ automatic conversion of date/time formats
- ❑ change from baseline
- ❑ time after dose and similar calculations
- ❑ filling of missing covariate values.

The settings file, provided in YAML format, allows for integration with customized graphical user interfaces. In addition, the settings file can be used to provide sensible and standardized default settings.

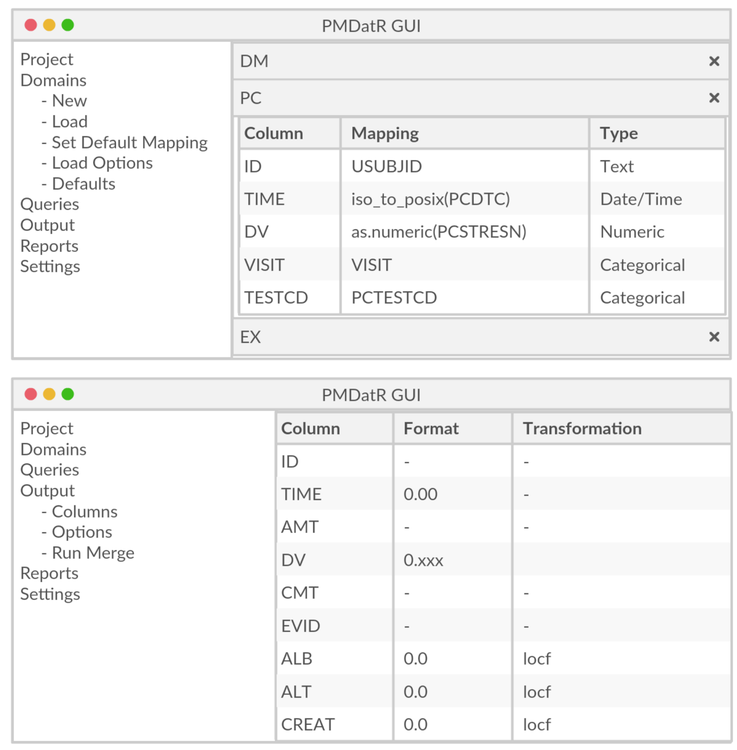
The overall process for dataset construction follows:

1) load source data and convert to standardized formats using customizable mappings
2) assign source data sets to a 'type' of data (observation, dose, merged covariates, event covariates) and apply mappings and transformations
3) stack event type data and merge covariates by key columns
4) apply transformations and filters that require the entire dataset (e.g. time after dose).

The data set creation process is run through Rmarkdown files, which allow for:

- ❑ generation of reports, with error reporting and samples of the processed inputs
- ❑ simple execution of the full process from command line or RStudio IDE (e.g. without the GUI).

Figure 3. GUI mockups (abstracted from existing proprietary GUI)



Results:

PMDatR is already in use in-house where it enables standardization of scripting style and quality control efforts. It is also in use at a major pharmaceutical company where it underpins a dataset creation tool having a graphical user interface to provide settings to the PMDatR package and collect and display results. The approach that links PMDatR as a back-end to a graphical user interface allows for additional features such as:

- ❑ drag-and-drop selection of columns and transformations
- ❑ point-and-click selection of options and templates
- ❑ syntax and semantic error checking
- ❑ additional help features for less experienced R programmers.

Conclusions

PMDatR provides a framework for pharmacometric dataset creation that is useful both as a standalone R package that provides a few additional tools for data manipulation, and as a powerful backend to more feature rich graphical user interface base applications that can integrate it into a data management ecosystem. In both cases, the benefits of a reusable, templatized, workflow can result in faster dataset creation and improved dataset quality.

The PMDatR package will be made available on CRAN.

Key References

[1] Hadley Wickham and Romain Francois (2016). dplyr: A Grammar of Data Manipulation. R package version 0.5.0. https://CRAN.R-project.org/package=dplyr

[2] Hadley Wickham (2016). tidyr: Easily Tidy Data with `spread()` and `gather()` Functions. R package version 0.4.0. https://CRAN.R-project.org/package=tidyr