PharmaSUG 2023 - Paper SD-257

Using R to Create Population Pharmacokinetic Data Set

Yangwei Yan, Prema Sukumar, and Neelima Thanneer, Bristol-Myers Squibb

ABSTRACT

Population Pharmacokinetic (PopPK) analyses are essential to evaluate the drug safety and efficacy among population subgroups in drug development. One of the key steps in PopPK modeling is to provide a structured input data set for NONMEM® and/or other modeling software. While SAS® has long been a dominant tool for statistical programming and analysis in the pharmaceutical industry, R has become a trending programming tool and is widely used in multiple areas due to its power and flexibility in supporting statistical analysis and advanced visualization. However, the use of R in PopPK data set preparation has not been discussed much.

This paper demonstrates a step-by-step process to generate data sets for PopPK analyses in RStudio using R markdown with example study data. It offers flexibility to use another programming language for programmers and pharmacometricians. We utilized typical R packages including tidyverse, lubridate, etc. and source data sets such as ADaM ADPC, ADEX, ADSL, etc. to create an ADPPK data set. It complies with the latest ISoP (International Society of Pharmacometrics) standard that's in the process of becoming CDISC ADaM PopPK Data set. The paper will mainly focus on the process of handling the variables with date and time format, conducting dose time imputation, and deriving time-varying variables. We also briefly summarize the advantages of using R in preparing PopPK analyses data set.

INTRODUCTION

Population pharmacokinetic (PopPK) analyses have been widely used to evaluate safety and efficacy of drugs, optimize dosing regimen and predict plasma concentrations in individual patients (FDA 1999, p1). The success of PopPK analyses relies heavily on the quality and structure of the data set used as input. Historically, SAS® has been utilized as the primary programming language in the pharmaceutical industry due to its robustness in dealing with complex data sets. However, R is increasingly being adopted by statisticians and data scientists because it offers more flexibility than SAS®. R is an open-source statistical software package with extensive libraries designed for various types of statistical analyses, data wrangling, and data visualization, which makes it suitable for exploring large data sets quickly and efficiently. Moreover, R offers better integration with other languages like Python through Application Programming Interface (API) calls which allows users to easily incorporate external scripts into their workflows thus reducing development effort significantly.

In this paper we will discuss how one can use RStudio along with typical packages available from CRAN/GitHub repository such as "tidyverse", "lubridate" etc., to create a structured data set specifically tailored towards Population Pharmacokinetic (PopPK) modeling tasks while complying with ISoP standards that's currently in the process of becoming CDISC ADaM PopPK (ADPPK) data set (ISoP 2020). We will mainly focus on explaining the steps required to create the ADPPK data set, performing date/time format handling, conducting dose time imputation, and deriving time-varying variables.

POPULATION PHARMACOKINETIC DATA SET PREPARATION USING RSTUDIO

Our program's primary output is a structured input data set called ADPPK for PopPK analyses, which conforms to the latest ISoP standard. In the example study described in this paper, subjects received intravenous doses once daily on days 1 through 7 of a 28-day cycle. For all subjects, PK samples were collected on days 5,6, and 7 of cycle 1. The primary source to generate ADPPK is subject-level information (ADSL), dosing data (ADEX), and PK concentration samples (ADPC).

We begin by installing the required R packages and importing source data into the R environment. It's always recommended to perform initial exploratory analysis before proceeding further towards building actual data set as this will help identify any issues related to missing values/outliers etc., which might result in erroneous results downstream. After successful importation, we follow five steps to create the

ADPPK data set: 1. Preparing the baseline data set. 2. Preparing the dosing data set. 3. Preparing the PK data set. 4. Combining the dosing and PK data set and conducting dose imputation. 5. Preparing the standard PopPK data set.

We use the following required R packages throughout the paper:

```
library(tidyverse)
library(haven)
library(labelled)
library(lubridate)
library(hms)
```

STEP1: PREPARE BASELINE DATA SET

For PopPK analysis data sets, baseline covariates such as age, sex, race, country, baseline weight, baseline height, etc. are typically sourced from ADSL. Extracting subject-level variables from the source data is straightforward as these variables are in a standard format.

As a pre-processing step, we run the R codes below to convert blank character variables to NA, the corresponding R representation of missing values:

```
adsl <- adsl %>%
  mutate(across(where(is.character), na_if, ""))
```

To facilitate the wrangling of baseline covariates, we utilize commonly used functions from the “tidyverse” package as follows:

```
baseline <- adsl %>%
  select(USUBJID, STUDYID, ARM, AGE, SEX, RACE, BLWEIGHT) %>%
  rename(WTB = BLWEIGHT) %>%
  mutate(
    SEXN = ifelse(SEX == "Female", 2, 1),
    RACEN = case_when(
      RACE == "White" ~ 1,
      RACE == "Others" ~ 4,
      RACE == "Unknown" ~ 5,
      TRUE ~ NA_real_
    ),
    BMIB = WTB / ((HTB / 100)^2)
  ) %>%
arrange(USUBJID)
```

The “select” function allows for the selection of relevant variables, while the “mutate” function can be used to modify and derive variables as necessary. We also use the “rename” function to rename variable names and the “arrange” function to sort the data set by the “USUBJID” variable. Typical tools for creating values that depend on different conditions include “ifelse” and “case_when” functions. The primary difference is that “ifelse” only allows for two possibilities, while “case_when” can handle multiple conditions.

STEP2: PREPARE DOSING DATA SET

Before delving into the programming details, it is essential to comprehend the structure of the source data. A sample of the ADEX domain with the variables of interest is presented in Figure 1. Each record in the data set represents a dosing record for a single day.

ID Unique Subject Identifier	EXDT Dosing Date	EXSTTM Dosing Start Time	EXENTM Dosing End Time	EXDOSE Dose Amount (mg)	EXDY Dosing Day
6	2017-06-08	*NA*	15:45:00	58	1
6	2017-06-09	*NA*	15:20:00	58	2
6	2017-06-10	*NA*	15:20:00	58	3
6	2017-06-11	*NA*	15:20:00	58	4
6	2017-06-12	*NA*	15:35:00	58	5
6	2017-06-13	*NA*	15:32:00	58	6
6	2017-06-14	*NA*	14:53:00	58	7
6	2017-07-06	15:15:00	15:25:00	58	29
6	2017-07-07	15:10:00	15:20:00	58	30
6	2017-07-08	*NA*	15:40:00	58	31

Figure 1. Sample ADEX domain

To obtain the desired output, the following R scripts are employed to derive the required variables:

```
dosing <- adex %>%
  select(ID, EXDY, EXDT, EXSTTM, EXENTM, EXDOSE) %>%
  rename(AMT = EXDOSE, ADY = EXDY, DATE = EXDT, TIME = EXSTTM) %>%
  filter(!is.na(DATE)) %>%
  mutate(
    DTTM = ymd_hms(paste(DATE, TIME)),
    EDTTM = ymd_hms(paste(DATE, EXENTM)),
    DOSEDUR = difftime(EDTTM, DTTM, units = "hours"),
    DOSEDUR = as.numeric(DOSEDUR),
    RATE = AMT / DOSEDUR
  )
```

Following variable selection, doses with missing dosing dates are excluded using the "filter" function as actual time after the dose cannot be calculated without the dosing date. In the "mutate" section, DTTM is created as the dosing date and time, while EDTTM is derived as the dosing end date and time. For intravenous drug administration, infusion duration (DOSEDUR) is calculated using the dosing start time (DTTM) and dosing end time (EDTTM), and infusion rate (RATE) is computed using the equation AMT/DOSEDUR.

The "lubridate" package is utilized in this step to handle date and time format variables. The "ymd_hms" function recognizes and parses date and time components such as year, month, day, hour, and minute. It facilitates the transformation of character or numeric vectors of dates and times into POSIXct objects, which are the most common type of date-time variables in R. The "difftime" function is used to calculate the time difference between two date/time objects. Since it returns an object of class "difftime" with an attribute indicating the units, it is preferable to convert DOSEDUR to a numeric vector using the "as.numeric" function.

Figure 2 demonstrates a desired output for dosing data set:

ID Unique Subject Identifier	DTTM Dose Date and Time	AMT Dose Amount (mg)	DOSEDUR Duration of Treatment Dose (hr)	RATE Infusion Rate (mg/hr)	EVID Event ID
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	NA	58	NA	NA	1
6	2017-07-06 15:15:00	58	0.1666667	348	1
6	2017-07-07 15:10:00	58	0.1666667	348	1
6	NA	58	NA	NA	1

Figure 2. Output dosing data set

STEP3: PREPARE PK DATA SET

PK concentration and sampling date and time are sourced from the ADPC dataset. Figure 3 illustrates the structure of ADPC and the variables required for preparing the PK data set. Each row corresponds to a single PK sample, with PCSTRESC and PCSTRESN indicating the concentration in character and numeric values, respectively, and PCDTC representing the actual sampling date and time. The nominal sampling time point after the dose is provided by PCTPT and PCTPTNUM, as defined in the protocol. For this study, pre-dose PK samples are collected on Cycle 1 days 5 and 6, while post-dose PK samples are collected on Cycle 1 day 7.

ID	PCSEQ Sequence Number	PCSTRESC Character Result/Finding in Std Format	PCSTRESN Numeric Result/Finding in Standard Units	PCDTC Date/Time of Specimen Collection	PCTPT Planned Time Point Name	PCTPTNUM Planned Time Point Number	VISIT Visit Name
6	1	< 1.00	0.0	2017-06-12T14:44:00	0 h pre-dose	0.000	Cycle 1 Day 5
6	2	< 1.00	0.0	2017-06-13T14:15:00	0 h pre-dose	0.000	Cycle 1 Day 6
6	3	607	607.0	2017-06-14T15:11:00	0.083 h (5 m) post-dose	0.083	Cycle 1 Day 7
6	4	87.1	87.1	2017-06-14T15:31:00	0.5 h post-dose	0.500	Cycle 1 Day 7
6	5	32.2	32.2	2017-06-14T16:06:00	1 h post-dose	1.000	Cycle 1 Day 7
6	6	< 1.00	NA	2017-06-14T19:06:00	4 h post-dose	4.000	Cycle 1 Day 7
6	7	< 1.00	NA	2017-06-14T21:06:00	6 h post-dose	6.000	Cycle 1 Day 7

Figure 3. Sample ADPC domain

The goal of PK data preparation is to format the concentration and sampling date and time in a way that is suitable for modeling. To this end, the following code snippets can be used to generate concentration-related variables, such as the PK concentration results (DV), the logarithm of concentration values (LOGDV), missing concentration values (MDV), and concentration flag (BLQFN), which indicates the samples that fall below the limit of quantification (ISoP 2020).

```
pk = adpc %>%
  select(
    ID, PCSEQ, PCSTRESC, PCSTRESN, PCDTC, PCTPT, PCTPTNUM, VISIT,
    VISITDY
    )%>%
  rename(NPRELTM = PCTPTNUM, DV = PCSTRESN) %>%
  mutate(
    EVID = 0,
    LOGDV = log(DV),
    MDV = ifelse(DV > 0 & !is.na(DV), 0, 1),
    BLQFN = ifelse(PCSTRESC == "< 1.00", 1, 0)
  )
```

As shown in Figure 3, PK sampling date and time is stored in the PCDTC variable. To use it for calculations in R, it is necessary to convert it to a POSIXct object using “ymd_hms” function. Moreover, we need to parse the date and time separately to utilize the time of PK samples for imputing dose time. The functions “date” from “lubridate” and “as_hms” from “hms” are applied to create sampling date (DATE) and sampling time (TIME) variables as follows:

```
pk <- pk %>%
```

```
  mutate(
    DTTM = ymd_hms(PCDTC),
    DATE = date(DTTM),
    TIME = as_hms(DTTM)
  )
```

The example of the output PK data set in this step is shown in Figure 4.

ID Unique Subject Identifier	DV Analysis Value (ng/mL)	MDV Missing DV	BLQFN BLQ Flag	LOGDV Logarithm of Analysis Value	DTTM PK Sampling Date and Time	DATE PK Sampling Date	TIME PK Sampling Time	EVID Event ID
6	NA	1	1	NA	2017-06-12 14:44:00	2017-06-12	14:44:00	0
6	NA	1	1	NA	2017-06-13 14:15:00	2017-06-13	14:15:00	0
6	607.0	0	0	6.408529	2017-06-14 15:11:00	2017-06-14	15:11:00	0
6	87.1	0	0	4.467057	2017-06-14 15:31:00	2017-06-14	15:31:00	0
6	32.2	0	0	3.471966	2017-06-14 16:06:00	2017-06-14	16:06:00	0
6	NA	1	1	NA	2017-06-14 19:06:00	2017-06-14	19:06:00	0
6	NA	1	1	NA	2017-06-14 21:06:00	2017-06-14	21:06:00	0

Figure 4. Output PK data set

STEP4: COMBINE DOSING AND PK DATA SET AND CONDUCT DOSE IMPUTATION

Once the daily dosing and PK data sets are ready, we use the function “bind_rows” from “tidyverse” to simply append the dosing records and PK samples. Figure 5 demonstrates the data set after row binding.

```
dosepk <- bind_rows(dosing, pk) %>%
  arrange (USUBJID, DTTM, EVID)
```

ID Unique Subject Identifier	DV Analysis Value (ng/mL)	MDV Missing DV	BLQFN BLQ Flag	LOGDV Logarithm of Analysis Value	DTTM Event Date and Time	DATE Event Date	TIME Event Time	EVID Event ID
6	NA	1	NA	NA	NA	2017-06-08	NA	1
6	NA	1	NA	NA	NA	2017-06-09	NA	1
6	NA	1	NA	NA	NA	2017-06-10	NA	1
6	NA	1	NA	NA	NA	2017-06-11	NA	1
6	NA	1	1	NA	2017-06-12 14:44:00	2017-06-12	14:44:00	0
6	NA	1	NA	NA	NA	2017-06-12	NA	1
6	NA	1	1	NA	2017-06-13 14:15:00	2017-06-13	14:15:00	0
6	NA	1	NA	NA	NA	2017-06-13	NA	1
6	607.0	0	0	6.408529	2017-06-14 15:11:00	2017-06-14	15:11:00	0
6	87.1	0	0	4.467057	2017-06-14 15:31:00	2017-06-14	15:31:00	0
6	32.2	0	0	3.471966	2017-06-14 16:06:00	2017-06-14	16:06:00	0
6	NA	1	1	NA	2017-06-14 19:06:00	2017-06-14	19:06:00	0
6	NA	1	1	NA	2017-06-14 21:06:00	2017-06-14	21:06:00	0
6	NA	1	NA	NA	NA	2017-06-14	NA	1
6	NA	1	NA	NA	2017-07-06 15:15:00	2017-07-06	15:15:00	1
6	NA	1	NA	NA	2017-07-07 15:10:00	2017-07-07	15:10:00	1
6	NA	1	NA	NA	NA	2017-07-08	NA	1

Figure 5. Combined dose and PK data set

After combining the dose and PK data sets, we can impute missing dosing times for dose records using the following general rules (Thanneer N, et al 2014):

1. If a trough PK sample is available on the same day, impute the dose time using the time of the trough sample.
2. If a post-dose sample is available on the same day, impute the dose time using the time of the sample minus the nominal sampling time (in this example, a 5-minute post-dose sample is used).
3. If there are no available trough or post-dose samples on the same day, impute the dose time using the time of the previous administered dose.
4. If there are no available trough, post-dose, or previous dose times on the same day, impute the dose time using the time of the next administered dose.

In this section, we demonstrate how to perform dose time imputation using R functions for rules 1 and 3. Trough samples are typically collected immediately before the next dose is administered. To impute with the time of trough PK samples, we begin with extracting the trough samples into a temporary tibble using the following codes:

```
imp1_trough <- dosepk %>%
  filter(EVID == 0 & NPRELTM == 0 & !(CYCLE == 1 & VISITDY == 1)) %>%
  rename(TDTTM = DTTM) %>%
  select(ID, DATE, TDTTM)
```

Next, we add the trough time back to the dosepk data set by performing a "left_join" on the ID and DATE variables, and perform dose imputation for step 1:

```
dosepk <- dosepk %>%
  left_join(imp1_trough) %>%
  mutate(
    IMP1_TF = ifelse(
      EVID == 1 & !is.na(AMT) & is.na(DTTM) & !is.na(TDTTM), TRUE, FALSE
      ),
    DTTM = case_when(
      IMP1_TF ~ TDTTM,
      TRUE ~ DTTM
    ),
    TIME = case_when(
      IMP1_TF~ as_hms(DTTM),
      TRUE ~ TIME
    )
  )
```

The above code ensures that the dose datetime is imputed to be the trough datetime (TDTTM) if they share the same date. To flag the records that require imputation, we create a variable called "IMP1_TF" which has a value of 1 for records with a missing dose time and a non-missing trough time. If IMP1_TF is 1, the dose time is imputed with the trough time; otherwise, the original datetime is retained.

After the imputation is complete, it is recommended to remove the temporary tibble:

```
rm(imp1_trough)
```

Table 1 provides an example before and after dose imputation with trough sampling datetime.

ID	EVID	DTTM_before	DTTM_after
6	0	2017-06-12 14:44:00	2017-06-12 14:44:00
6	1	NA	2017-06-12 14:44:00
6	1	2017-06-13 14:15:00	2017-06-13 14:15:00

Table 1. Comparison before and after dose time imputation with trough datetime

To perform step 3 of the imputation process, we run the R codes below:

```
dosepk <- dosepk %>%
  arrange(ID, EVID, DATE, DTTM) %>%
  group_by(ID, EVID) %>%
  mutate(IMP_TIME_PRE = TIME) %>%
  fill(IMP_TIME_PRE) %>%
  ungroup() %>%
  mutate(
    IMP3_TF = if_else(
      EVID == 1 & !is.na(AMT) & !is.na(DATE) & is.na(DTTM), TRUE, FALSE
      ),
    TIME = ifelse(IMP3_TF, IMP_TIME_PRE, TIME),
    TIME = as_hms(TIME),
```

```
    DTTM = case_when(
      IMP3_TF ~ ymd_hms(paste(DATE, TIME)),
      TRUE ~ DTTM
    )
  )
```

We use the "group_by" function to group the dosing events (EVID=1) by subject and order them by date and time. We then create a variable called “IMP_TIME_PRE” and use the "fill" function to fill in missing values in the "IMP_TIME_PRE" column using the previous entry. Once we obtain the previous dose time, we use a similar process in the "mutate" section as in the trough time imputation to complete this step. Table 2 provides an example before and after dose imputation with the previous dose datetime.

ID	EVID	DTTM_before	DTTM_after
6	1	2017-07-06 15:15:00	2017-07-06 15:15:00
6	1	2017-07-07 15:10:00	2017-07-07 15:10:00
6	1	NA	2017-07-08 15:10:00

Table 2. Comparison before and after dose time imputation with previous dose datetime

Once the dose time imputation process is completed, the subsequent crucial step is to derive time-related variables such as the actual time after previous dose (APRELTM) and the actual time after first dose (AFRELTM). To calculate the time-related variables, it is necessary to obtain the first dose time for each subject as well as the previous dose time for each PK record.

We start with creating a data set containing the first dose, and a variable DOSE1_DTTM indicating the first dose datetime for each subject:

```
dosing_1st <- dosepk %>%
  filter(EVID == 1) %>%
  arrange(ID, DTTM) %>%
  group_by(ID) %>%
  mutate(DOSE1_DTTM = DTTM[1]) %>%
  select(ID, DOSE1_DTTM) %>%
  ungroup() %>%
  distinct()

dosepk <- left_join(
  dosepk,
  dosing_1st,
  by = "ID"
)
```

As shown in the code snippet provided above, once the data set is sorted by subject (ID) and datetime (DTTM), we can easily obtain the first dose datetime by using “DOSE1_DTTM = DTTM[1]” within each ID group. This step enables us to extract the value on the first row for each subject. Subsequently, we add back the first dose datetime into the dosepk data set.

Once we obtain the first dose time, we can run the following R codes to obtain the previous dose time for each PK record and derive AFRELTM and APRELTM:

```
dosepk <- dosepk %>%
  arrange(ID, DTTM, EVID) %>%
  group_by(ID) %>%
  # Get the previous dose datetime for each pk record.
  mutate(
    PDOSE_DTTM = case_when(
      EVID == 1 & AMT > 0 & !is.na(DTTM) ~ DTTM
      )
    ) %>%
  fill(PDOSE_DTTM) %>%
```

```
    ungroup() %>%
    mutate(
      AFRELTM = difftime(DTTM, DOSE1_DTTM, units = "hours"),
      AFRELTM = as.numeric(AFRELTM),
      APRELTM = difftime(DTTM, PDOSE_DTTM, units = "hours"),
      APRELTM = as.numeric(APRELTM)
  )
```

In this section, the variable “PDOSE_DTTM” is created to store the previous dose time, and the "fill" function is employed to fill in missing values using the previous entry within each group ID. We once again use “difftime” to calculate the time difference in hours between the current PK sampling time and the first/previous dose time. Subsequently, the calculated values are converted into numeric format. By the end of step 4, the dose and PK data set should be in good shape, as demonstrated in Figure 6.

ID	EVID	DV	BLQFN	MDV	LOGDV	AFRELTM	APRELTM	DTTM	AMT	DOSEDUR	RATE
6	1	NA	NA	1	NA	0.0000	0.00000000	2017-06-08 14:44:00	58	NA	NA
6	1	NA	NA	1	NA	24.0000	0.00000000	2017-06-09 14:44:00	58	NA	NA
6	1	NA	NA	1	NA	48.0000	0.00000000	2017-06-10 14:44:00	58	NA	NA
6	1	NA	NA	1	NA	72.0000	0.00000000	2017-06-11 14:44:00	58	NA	NA
6	0	NA	1	1	NA	96.0000	24.00000000	2017-06-12 14:44:00	NA	NA	NA
6	1	NA	NA	1	NA	96.0000	0.00000000	2017-06-12 14:44:00	58	NA	NA
6	0	NA	1	1	NA	119.5167	23.51666667	2017-06-13 14:15:00	NA	NA	NA
6	1	NA	NA	1	NA	119.5167	0.00000000	2017-06-13 14:15:00	58	NA	NA
6	1	NA	NA	1	NA	144.3667	0.00000000	2017-06-14 15:06:00	58	NA	NA
6	0	607.0	0	0	6.408529	144.4500	0.08333333	2017-06-14 15:11:00	NA	NA	NA
6	0	87.1	0	0	4.467057	144.7833	0.41666667	2017-06-14 15:31:00	NA	NA	NA
6	0	32.2	0	0	3.471966	145.3667	1.00000000	2017-06-14 16:06:00	NA	NA	NA
6	0	NA	1	1	NA	148.3667	4.00000000	2017-06-14 19:06:00	NA	NA	NA
6	0	NA	1	1	NA	150.3667	6.00000000	2017-06-14 21:06:00	NA	NA	NA
6	1	NA	NA	1	NA	672.5167	0.00000000	2017-07-06 15:15:00	58	0.1666667	348
6	1	NA	NA	1	NA	696.4333	0.00000000	2017-07-07 15:10:00	58	0.1666667	348
6	1	NA	NA	1	NA	720.4333	0.00000000	2017-07-08 15:10:00	58	NA	NA

Figure 6. Combined dose and PK output

STEP5: PREPARE STANDARD POPPK DATA SET

The final step involves combining the dosepk data set with baseline covariates, and ensure that the resulting data set meets the requirements of PopPK modeling based on the standard data specification. Several tasks can be completed within this step, such as deriving the remaining variables, flagging the records based on exclusion criteria, rounding the numeric variables, and adding variable labels. Below, we provide some typical sample codes for this step.

Sample codes for combining the dosepk and baseline covariates:

```
ppk <- left_join(dosepk, baseline, by = "ID")
```

Sample codes for rounding the numeric variables to 0.01 as necessary:

```
rou001_vars <- c("AFRELTM", "APRELTM", "BMIB", "BSAB")
rnd001 <- function(x) {
  floor(x * 100 + 0.5) / 100
}
ppk <- ppk %>% mutate_at(rou001_vars, rnd001)
```

It is important to note that the "round" function in R always rounds half-way decimals to the nearest even number, while SAS® always rounds up. Therefore, the sample code provided above helps to align with the rounding rule in SAS®.

Sample codes for adding variable labels using the function “set_variable_labels” from “labelled” package:

```
ppk <- ppk %>%
  set_variable_labels(
    STUDYID = "Study Identifier",
```

```
    USUBJID = "Unique Subject Identifier",
    AFRELTM = "Actual Rel Time From First Dose[hr]",
    DV = "Analysis Value[ng/mL]"
  )
```

Figure 7 illustrates the structure of final standard PopPK data set with some variables commonly used in the modeling.

ID NONMEM ID	EVID Event ID	DV Analysis Value[ng/mL]	BLQFN Blq Flag	MDV Missing DV	AFRELTM Actual Rel Time From First Dose[hr]	APRELTM Actual Rel Time From Previous Dose[hr]	DOSEA Actual Treatment Dose[mg]	AGE Age[year]	WTB Baseline Body Weight[kg]
6	1	NA	NA	1	0.00	0.00	58	6.9	18.5
6	1	NA	NA	1	24.00	0.00	58	6.9	18.5
6	1	NA	NA	1	48.00	0.00	58	6.9	18.5
6	1	NA	NA	1	72.00	0.00	58	6.9	18.5
6	0	NA	1	1	96.00	24.00	58	6.9	18.5
6	1	NA	NA	1	96.00	0.00	58	6.9	18.5
6	0	NA	1	1	119.52	23.52	58	6.9	18.5
6	1	NA	NA	1	119.52	0.00	58	6.9	18.5
6	1	NA	NA	1	144.37	0.00	58	6.9	18.5
6	0	607.0	0	0	144.45	0.08	58	6.9	18.5
6	0	87.1	0	0	144.78	0.42	58	6.9	18.5
6	0	32.2	0	0	145.37	1.00	58	6.9	18.5
6	0	NA	1	1	148.37	4.00	58	6.9	18.5
6	0	NA	1	1	150.37	6.00	58	6.9	18.5
6	1	NA	NA	1	672.52	0.00	58	6.9	18.5
6	1	NA	NA	1	696.43	0.00	58	6.9	18.5
6	1	NA	NA	1	720.43	0.00	58	6.9	18.5

Figure 7. Sample ADDPK data set for a PopPK analysis

ADVANTAGES OF USING R FOR POPPK DATA SET PREAPRATION

As the popularity of R as a statistical programming tool rises, there have been many discussions on the comparisons of R and SAS® in the pharmaceutical and biotech industries. In this paper, we briefly summarize some advantages of using R for preparing PopPK data sets.

Due to the widespread use of R in pharmacometric PopPK analyses, it has become necessary to use R for preparing PopPK data sets, especially for those who work on exploratory analyses without the support of programmers. The codes provided in this paper can be seamlessly integrated into the workflow of pharmacometricians, greatly increasing their efficiency by utilizing R and Rmarkdown for preparing PopPK data sets. Rmarkdown allows users to create dynamic and user-friendly documents that include text, R codes, output, as well as summaries and graphics used to analyze data sets. It also allows users to generate a wide range of document formats (pdf, html web pages, MS Word, etc.), which can be easily shared and collaborated on using version control systems like Git.

In addition, R is designed to work with a wide range of data formats, including CSV, Excel, SQL, and others, making it more flexible in terms of data integration. This flexibility is particularly useful for users working in the early clinical trial stage, who may not have access to standard SDTM/ADaM data sets as a source.

Another key advantage of using R is its large and active community of users, who create and share packages containing pre-built functions and tools. Such examples are “tidyverse” and “lubridate”, which provide a wide range of data manipulation techniques, including reshaping, merging, transforming, and retaining values, which are essential for preparing PopPK data set.

Furthermore, the powerful integrated development environment (IDE) RStudio® offers the ability to call up potential syntax options and variables names with the tab key, which makes it easier to write new scripts. Additionally, it offers a convenient interface to view and interact with objects stored in the environment, as well as debugging tools, syntax highlighting, and version control through integration with Git and GitHub.

Overall, R provides flexibility to non-SAS users in PopPK data set preparation. However, SAS® is still widely used in many industries and has its own advantages, such as a long-standing reputation for reliability, support, and compatibility with legacy systems.

CONCLUSION

In this paper we explore the use of R for creating a PopPK data set, outlining a five-step process for preparing and combining the baseline, dosing, and PK data sets. We provide a detailed, step-by-step programming guide, using an example study, and ensure that the resulting data set is compliant with the upcoming CDISC standard. We have leveraged the power of interactive IDE RStudio, along with packages like “tidyverse”, “lubridate”, and “hms”, to facilitate efficient data manipulation. We successfully tackled some of the challenges associated with handling datetime format variables and perform complex dose imputations with relatively concise code snippets with R. We believe that our comprehensive guide will provide significant benefits to pharmacometricians and programmers working on PopPK data set programming.

REFERENCES

Food and Drug Administration, 1999. “Guidance for industry: population pharmacokinetics.” http://www.fda. gov/cder/guidance/1852fnl.pdf.

International Society of Pharmacometrics, 2020. “PopPK Data Standard Implementation Guide.” http://go-isop.org/wp-content/uploads/2020/11/PopPK-Data-Standard-Implementation-Guide.pdf.

Thanneer N, Roy A, Sukumar P, Bandaru J, Carleen E. Oct 12-15, 2014. “Best Practices for Preparation of Pharmacometric Analysis Data Sets.” *Poster session presented at: 5th American Conference on Pharmacometrics,* Las Vegas, NV.

Hadley W. and Garrett G. “R for Data Science.” Accessed March 1, 2023. https://r4ds.had.co.nz/index.html

ACKNOWLEDGMENTS

The author would like to acknowledge all Bristol-Myers Squibb, Princeton, NJ, USA colleagues who have provided their valuable inputs on this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yangwei Yan
Senior Data Scientist
Bristol-Myers Squibb, Princeton, NJ, USA
E-mail: yangwei.yan@bms.com