Population Pharmacokinetic Analysis Data (ADPPK) Programming in {admiral} and the Pharmaverse

Jeffrey Dickinson, Navitas Data Sciences, Pottstown, PA, USA

# 1. Abstract

Population Pharmacokinetic modeling is an important tool for drug development. The CDISC ADaM Population PK Implementation Guide was released on October 6, 2023. Population PK models generally make use of nonlinear mixed effects models that require numeric variables. The data used in the models will include both dosing and concentration records, relative time variables, and numeric covariate variables. {admiral} is an open-source R package for creating CDISC ADaM data. It can be used effectively to create Population PK analysis data (ADPPK). Additional tools from other Pharmaverse packages such as {metacore}, {metatools} and {xportr} can be used to simplify the workflow. I will discuss some of the challenges of Population Pharmacokinetic analysis data programming and show some of the solutions developed in {admiral} and the Pharmaverse.

# 2. Introduction

Pharmacokinetics considers the effect of the body on a drug. Typically samples are drawn at set time intervals after dose administration. The resulting concentration profiles can be analyzed. With population pharmacokinetic models, variations within and between populations can be assessed. The CDISC ADaM Population PK Analysis Data Implementation Guide was released on October 6, 2023. (<https://www.cdisc.org/standards/foundational/adam/basic-data-structure-adam-poppk-implementation-guide-v1-0>)

Population PK models generally make use of nonlinear mixed effects models that require numeric variables. The data used in the models will include both dosing and concentration records, relative time variables, and numeric covariate variables. A DV or dependent variable is often expected, typically the concentration. This is equivalent to the ADaM AVAL variable and will be included in addition to AVAL for ADPPK. The relative time variables are listed in the table below. Also below are the expected variables unique to ADPPK and the numeric covariates.

## 2.1 Time Variables (ADPPK)

These are the names for the relative time variables:

| Variable | Variable Label |
| --- | --- |
| NFRLT | Nominal Rel Time from First Dose |
| AFRLT | Actual Rel Time from First Dose |
| NPRLT | Nominal Rel Time from Previous Dose |
| APRLT | Actual Rel Time from Previous Dose |

## 2.2 Expected Variables (ADPPK)

These variables are expected in ADPPK:

| Variable | Variable Label |
| --- | --- |
| EVID | Event ID |
| DV | Dependent Variable |
| MDV | Missing Dependent Variable |
| BLQ | Below Limit of Quantitation |

## 2.3 ADPPK Covariates

These are the conventions for the numeric covariates:

| Covariate | Covariate Example |
| --- | --- |
| <COV>BL | Baseline covariate, (e.g., WTBL, BMIBL) |
| <COV>N | Numeric version of categorical covariate (e.g., SEXN, RACEN) |
| <COV>I | Covariates with imputed values (e.g., WTI, BMII) |
| <COV>GRy | Grouping covariates (e.g. AGEGR1 |

# 3. {admiral}

{admiral} is an open-source R package for creating CDISC analysis datasets. It is modular and consists of a set of functions for many of the operations required for dataset construction. There are template programs available for most CDISC ADaM datasets including for PK (i.e. ADPPK, ADNCA and ADPP). The {admiral} functions used are listed below:

## 3.1 {admiral} Functions Used

* derive\_vars\_dtm()
* derive\_vars\_dtm\_to\_dt()
* derive\_vars\_dtm\_to\_tm()
* derive\_vars\_dy()
* derive\_vars\_duration()
* create\_single\_dose\_dataset()
* derive\_vars\_merged()
* derive\_vars\_joined()
* derive\_vars\_transposed()
* compute\_bmi()
* compute\_bsa()
* compute\_egfr()

## 3.2 Programming Workflow.

The complete programming workflow for building the ADPPK analysis dataset can be found on the Pharmaverse examples website <https://pharmaverse.github.io/examples/adam/adppk.html>. I will highlight the following steps here:

* Load Specs with {metacore}
* Derive PC Dates
* Expand Dosing Records
* Find First Dose
* Find Previous Dose
* Find Previous Nominal Dose
* Derive Covariates Using {metacore}
* {metacore} Checks
* {xportr} Steps

## 3.3 Load Specifications for {metacore}

There is a sample specification file for ADPPK which can be loaded into {metacore}. The file is loaded into {metacore} with the metacore::spec\_to\_metacore() function. The spec file can be found [here](https://github.com/pharmaverse/examples/blob/main/adam/pk_spec.xlsx).

# ---- Load Specs for Metacore ----  
metacore <- spec\_to\_metacore("pk\_spec.xlsx") %>%  
 select\_dataset("ADPPK")

### 3.3.1 Derive PC Dates

One of the first things done after loading the SDTM domains is to convert character dates to numeric. We will also combine some variables from ADSL at this step. Only the ADSL variables used for derivations are selected at this step. The rest of the relevant ADSL variables will be added later. In this case we will keep TRTSDT/TRTSDTM for day derivation and TRT01P/TRT01A for planned and actual treatments.

In this segment we will use admiral::derive\_vars\_merged() to join the ADSL variables and the following {admiral} functions to derive analysis dates, times and days:

* derive\_vars\_dtm()
* derive\_vars\_dtm\_to\_dt()
* derive\_vars\_dtm\_to\_tm()
* derive\_vars\_dy()

We will also create NFRLT for PC data based on PCTPTNUM. We will create an event ID (EVID) of 0 for concentration records and 1 for dosing records.

# ---- Derivations ----  
  
# Get list of ADSL vars required for derivations  
adsl\_vars <- exprs(TRTSDT, TRTSDTM, TRT01P, TRT01A)  
  
pc\_dates <- pc %>%  
 # Join ADSL with PC (need TRTSDT for ADY derivation)  
 derive\_vars\_merged(  
 dataset\_add = adsl,  
 new\_vars = adsl\_vars,  
 by\_vars = exprs(STUDYID, USUBJID)  
 ) %>%  
 # Derive analysis date/time  
 # Impute missing time to 00:00:00  
 derive\_vars\_dtm(  
 new\_vars\_prefix = "A",  
 dtc = PCDTC,  
 time\_imputation = "00:00:00"  
 ) %>%  
 # Derive dates and times from date/times  
 derive\_vars\_dtm\_to\_dt(exprs(ADTM)) %>%  
 derive\_vars\_dtm\_to\_tm(exprs(ADTM)) %>%  
 # Derive event ID and nominal relative time from first dose (NFRLT)  
 mutate(  
 EVID = 0,  
 DRUG = PCTEST,  
 NFRLT = if\_else(PCTPTNUM < 0, 0, PCTPTNUM), .after = USUBJID  
 )

### 3.3.2 Expand Dosing Records

The function admiral::create\_single\_dose\_dataset() will be used to expand dosing records between the start date and end date. The nominal time will also be expanded based on the values of EXDOSFRQ, for example “QD” will result in nominal time being incremented by 24 hours and “BID” will result in nominal time being incremented by 12 hours.

# ---- Expand dosing records between start and end dates ----  
# Updated function includes nominal\_time parameter  
  
ex\_exp <- ex\_dates %>%  
 create\_single\_dose\_dataset(  
 dose\_freq = EXDOSFRQ,  
 start\_date = ASTDT,  
 start\_datetime = ASTDTM,  
 end\_date = AENDT,  
 end\_datetime = AENDTM,  
 nominal\_time = NFRLT,  
 lookup\_table = dose\_freq\_lookup,  
 lookup\_column = CDISC\_VALUE,  
 keep\_source\_vars = exprs(  
 STUDYID, USUBJID, EVID, EXDOSFRQ, EXDOSFRM,  
 NFRLT, EXDOSE, EXDOSU, EXTRT, ASTDT, ASTDTM, AENDT, AENDTM,  
 VISIT, VISITNUM, VISITDY,  
 TRT01A, TRT01P, DOMAIN, EXSEQ, !!!adsl\_vars  
 )  
 ) %>%  
 # Derive AVISIT based on nominal relative time  
 # Derive AVISITN to nominal time in whole days using integer division  
 # Define AVISIT based on nominal day  
 mutate(  
 AVISITN = NFRLT %/% 24 + 1,  
 AVISIT = paste("Day", AVISITN),  
 ADTM = ASTDTM,  
 DRUG = EXTRT  
 ) %>%  
 # Derive dates and times from datetimes  
 derive\_vars\_dtm\_to\_dt(exprs(ADTM)) %>%  
 derive\_vars\_dtm\_to\_tm(exprs(ADTM)) %>%  
 derive\_vars\_dtm\_to\_tm(exprs(ASTDTM)) %>%  
 derive\_vars\_dtm\_to\_tm(exprs(AENDTM))

### 3.3.3 Find First Dose

We find the first dose for the concentration records using the function admiral::derive\_vars\_merged()

# ---- Find first dose per treatment per subject ----  
# ---- Join with ADPPK data and keep only subjects with dosing ----  
  
adppk\_first\_dose <- pc\_dates %>%  
 derive\_vars\_merged(  
 dataset\_add = ex\_exp,  
 filter\_add = (!is.na(ADTM)),  
 new\_vars = exprs(FANLDTM = ADTM, EXDOSE\_first = EXDOSE),  
 order = exprs(ADTM, EXSEQ),  
 mode = "first",  
 by\_vars = exprs(STUDYID, USUBJID, DRUG)  
 ) %>%  
 filter(!is.na(FANLDTM)) %>%  
 # Derive AVISIT based on nominal relative time  
 # Derive AVISITN to nominal time in whole days using integer division  
 # Define AVISIT based on nominal day  
 mutate(  
 AVISITN = NFRLT %/% 24 + 1,  
 AVISIT = paste("Day", AVISITN),  
 )

### 3.3.4 Find Previous Dose

For ADPPK we will find the previous dose with respect to actual time and nominal time. We will use admiral::derive\_vars\_joined().

# ---- Find previous dose ----  
  
adppk\_prev <- adppk\_first\_dose %>%  
 derive\_vars\_joined(  
 dataset\_add = ex\_exp,  
 by\_vars = exprs(USUBJID),  
 order = exprs(ADTM),  
 new\_vars = exprs(  
 ADTM\_prev = ADTM, EXDOSE\_prev = EXDOSE, AVISIT\_prev = AVISIT,  
 AENDTM\_prev = AENDTM  
 ),  
 join\_vars = exprs(ADTM),  
 join\_type = "all",  
 filter\_add = NULL,  
 filter\_join = ADTM > ADTM.join,  
 mode = "last",  
 check\_type = "none"  
 )

### 3.3.5 Find Previous Nominal Dose

Here we use admiral::derive\_vars\_joined() to also find the previous nominal dose.

# ---- Find previous nominal dose ----  
  
adppk\_nom\_prev <- adppk\_prev %>%  
 derive\_vars\_joined(  
 dataset\_add = ex\_exp,  
 by\_vars = exprs(USUBJID),  
 order = exprs(NFRLT),  
 new\_vars = exprs(NFRLT\_prev = NFRLT),  
 join\_type = "all",  
 join\_vars = exprs(NFRLT),  
 filter\_add = NULL,  
 filter\_join = NFRLT > NFRLT.join,  
 mode = "last",  
 check\_type = "none"  
 )

## 3.4 Derive Covariates Using Metacore

In this step we will create our numeric covariates using the metatools::create\_var\_from\_codelist() function.

#---- Derive Covariates ----  
# Include numeric values for STUDYIDN, USUBJIDN, SEXN, RACEN etc.  
  
covar <- adsl %>%  
 create\_var\_from\_codelist(metacore, input\_var = STUDYID, out\_var = STUDYIDN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = SEX, out\_var = SEXN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = RACE, out\_var = RACEN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = ETHNIC, out\_var = AETHNIC) %>%  
 create\_var\_from\_codelist(metacore, input\_var = AETHNIC, out\_var = AETHNICN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = ARMCD, out\_var = COHORT) %>%  
 create\_var\_from\_codelist(metacore, input\_var = ARMCD, out\_var = COHORTC) %>%  
 create\_var\_from\_codelist(metacore, input\_var = COUNTRY, out\_var = COUNTRYN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = COUNTRY, out\_var = COUNTRYL) %>%  
 mutate(  
 STUDYIDN = as.numeric(word(USUBJID, 1, sep = fixed("-"))),  
 SITEIDN = as.numeric(word(USUBJID, 2, sep = fixed("-"))),  
 USUBJIDN = as.numeric(word(USUBJID, 3, sep = fixed("-"))),  
 SUBJIDN = as.numeric(SUBJID),  
 ROUTE = unique(ex$EXROUTE),  
 FORM = unique(ex$EXDOSFRM),  
 REGION1 = COUNTRY,  
 REGION1N = COUNTRYN,  
 SUBJTYPC = "Volunteer",  
 ) %>%  
 create\_var\_from\_codelist(metacore, input\_var = FORM, out\_var = FORMN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = ROUTE, out\_var = ROUTEN) %>%  
 create\_var\_from\_codelist(metacore, input\_var = SUBJTYPC, out\_var = SUBJTYP)

## 3.5 Check Data With {metacore}

We use {metacore} to perform a number of checks on the data. We will drop variables not in the specs and make sure all the variables from the specs are included.

# Final Steps, Select final variables and Add labels  
# This process will be based on your metadata, no example given for this reason  
# ...  
  
  
# Apply metadata and perform associated checks ----  
# uses {metatools}  
  
adppk <- adppk\_prefinal %>%  
 drop\_unspec\_vars(metacore) %>% # Drop unspecified variables from specs  
 check\_variables(metacore) %>% # Check all variables specified are present  
 check\_ct\_data(metacore) %>% # Checks Control Terminology  
 order\_cols(metacore) %>% # Orders the columns according to the spec  
 sort\_by\_key(metacore) # Sorts the rows by the sort keys

## 3.6 Apply Labels and Formats with {xportr}

Using {xportr} we check variable type, assign variable length, add variable labels, add variable formats, and save a transport file.

adppk\_xpt <- adppk %>%  
 xportr\_type(metacore) %>% # Coerce variable type to match spec  
 xportr\_length(metacore) %>% # Assigns SAS length from a variable metadata  
 xportr\_label(metacore) %>% # Assigns variable label from specifications  
 xportr\_format(metacore) %>% # Assigns variable format from specifications  
 xportr\_df\_label(metacore) %>% # Assigns dataset label from specifications  
 xportr\_write(file.path(dir, "adppk.xpt")) # Write xpt v5 transport file

# 4. Conclusions

Analsis data standards for Population PK (ADPPK) have been recently released. Within a few years submissions to regulatory agencies such as the FDA may require submissions of ADPPK analysis data. As we begin to see submissions programmed in R the use of packages such as {admiral} will become more widespread. The code presented here shows that {admiral} and other Pharmaverse packages provide an excellent way to program ADPPK analysis data.

# 5. Contact Information

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

Author Name: Jeffrey Dickinson

Company: Navitas Data Sciences

Address: 1610 Medical Drive, Suite 300, Pottstown, PA 19464 USA

Work Phone: +1 402 319 9380

Email: jeff.dickinson@navitaslifesciences.com

Website: www.navitaslifesciences.com