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

Jeffrey Dickinson, Navitas Data Sciences

Table of contents

0.1 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.

0.2 Introduction

Pharmacokinetics considers the effect of the body on a drug. Typically samples are drawn at set time intervals after dose administration as illustrated in the schematic above. 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. 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.

The Population PK Analysis Data (ADPPK) follows the CDISC Implementation Guide (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. This is equivalent to the ADaM `AVAL` variable and will be included in addition to `AVAL` for ADPPK.

0.3 Time Variables (`ADPPK`)

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

0.4 `ADPPK` Covariates

- `<COV>BL` for baseline covariate, (e.g., `WTBL`, `BMIBL`)
- `<COV>N` for numerical version of categorical covariate (e.g., `SEXN`, `RACEN`)
- `<COV>I` for any covariates with imputed values (e.g., `WTI`, `BMII`)
- `<COV>GRy` for grouping covariates (e.g. `AGEGR1`)

0.5 `{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 table to the right shows the functions used in the creation of the ADPPK dataset. Follow the link in the QR code below to see an ADPPK template example and run the code in Posit Cloud.

0.6 `{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()`

0.7 Programming Workflow.

- 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

0.8 Load Specifications for `{metacore}`

We have saved our specifications in an Excel file and will load them into `{metacore}` with the `metacore::spec_to_metacore()` function. The spec file can be found here

```
# ---- Load Specs for Metacore ----
metacore <- spec_to_metacore("pk_spec.xlsx") %>%
  select_dataset("ADPPK")
```

0.8.1 Derive PC Dates

At this step, it may be useful to join `ADSL` to your `PC` and `EX` domains as well. 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 `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
  )
```

0.8.2 Expand Dosing Records

The `{admiral}` function `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(
```

```r
        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))
```

0.8.3 Find First Dose

We find the first dose for the concentration records using the `{admiral}` function `derive_vars_merged()`

```r
# ---- 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),
  )
```

0.8.4 Find Previous Dose

For ADPPK we will find the previous dose with respect to actual time and nominal time. We will use ‘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"
  )
```

0.8.5 Find 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"
  )
```

0.9 Derive Covariates Using Metacore

In this step we will create our numeric covariates using the `create_var_from_codelist()` function from `{metatools}`.

```
#---- 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) %>%
```

```r
  create_var_from_codelist(metacore, input_var = ROUTE, out_var = ROUTEN) %>%
  create_var_from_codelist(metacore, input_var = SUBJTYPC, out_var = SUBJTYP)
```

0.10 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.

```r
# Final Steps, Select final variables and Add labels
# This process will be based on your metadata, no example given for this reason
# ...
dir <- "./output"

# 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 and no more
  check_ct_data(metacore) %>% # Checks all variables with CT only contain values within th
  order_cols(metacore) %>% # Orders the columns according to the spec
  sort_by_key(metacore) # Sorts the rows by the sort keys
```

0.11 Apply Labels and Formats with xportr

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

```r
adppk_xpt <- adppk %>%
  xportr_type(metacore) %>% # Coerce variable type to match spec
  xportr_length(metacore) %>% # Assigns SAS length from a variable level metadata
  xportr_label(metacore) %>% # Assigns variable label from metacore specifications
  xportr_format(metacore) %>% # Assigns variable format from metacore specifications
  xportr_df_label(metacore) %>% # Assigns dataset label from metacore specifications
  xportr_write(file.path(dir, "adppk.xpt")) # Write xpt v5 transport file
```

1 Example Scripts

ADaM	Sample Code
ADPPK	ad_adppk_spec.R

2 Spec File

pk_spec.xlsx