ADSL

Introduction

This guide will show you how four pharmaverse packages, along with some from tidyverse, can be used to create an ADaM such as `ADSL` end-to-end, using pharmaversesdtm SDTM data as input.

The four packages used with a brief description of their purpose are as follows:

- `{metacore}` : provides harmonized metadata/specifications object.
- `{metatools}` : uses the provided metadata to build/enhance and check the dataset.
- `{admiral}` : provides the ADaM derivations.
- `{xportr}` : delivers the SAS transport file (XPT) and eSub checks.

It is important to understand metacore objects by reading through the above linked package site, as these are fundamental to being able to use metatools and xportr. Each company may need to build a specification reader to create these objects from their source standard specification templates.

Load Data and Required pharmaverse Packages

The first step is to load our pharmaverse packages and i

```
library(metacore)
library(metatools)
library(pharmaversesdtm)
library(admiral)
library(xportr)
library(dplyr)
library(tidyr)
library(lubridate)
library(stringr)

# Read in input SDTM data
data("dm")
data("ex")
```

Next we need to load the specification file in the form o

```
# Read in metacore object
load(metacore_example("pilot_ADaM.rda"))
metacore <- metacore %>%
   select_dataset("ADSL")
```

Here is an example of how a metacore object looks showing variable level metadata:

```
metacore$ds_vars
```

```
# A tibble: 49 × 7
   dataset variable key_seq order keep  core  supp_flag
   <chr>   <chr>      <int> <int> <lgl> <chr> <lgl>
 1 ADSL    STUDYID       NA     1 FALSE <NA>  NA
 2 ADSL    USUBJID        1     2 FALSE <NA>  NA
 3 ADSL    SUBJID        NA     3 FALSE <NA>  NA
 4 ADSL    SITEID        NA     4 FALSE <NA>  NA
 5 ADSL    SITEGR1       NA     5 FALSE <NA>  NA
 6 ADSL    ARM           NA     6 FALSE <NA>  NA
 7 ADSL    TRT01P        NA     7 FALSE <NA>  NA
 8 ADSL    TRT01PN       NA     8 FALSE <NA>  NA
 9 ADSL    TRT01A        NA     9 FALSE <NA>  NA
10 ADSL    TRT01AN       NA    10 FALSE <NA>  NA
# ℹ 39 more rows
```

Start Building Derivations

The first derivation step we are going to do is to pull through all the columns that come directly from the SDTM datasets. You might know which datasets you are going to pull from directly already, but if you don't you can call `metatools::build_from_derived()` with just an empty list and the error will tell you which datasets you need to supply.

```
build_from_derived(metacore, list(), pr
```

```
Error in build_from_derived(metacore, list(),
provided. Please pass the following dataset(s
DM
```

In this case all the columns come from `DM` so that is the `metatools::build_from_derived()`. The resulting da that needed renaming between SDTM and ADaM are re

```
adsl_preds <- build_from_derived(metacc
                                 ds_lis
                                 predec
head(adsl_preds, n=10)
```

```
# A tibble: 10 × 14
   STUDYID      USUBJID SUBJID SITEID ARM     AGE AGEU  RACE  SEX   ETHNIC DTHFL
   <chr>        <chr>   <chr>  <chr>  <chr> <dbl> <chr> <chr> <chr> <chr>  <chr>
 1 CDISCPILOT01 01-701… 1015   701    Plac…    63 YEARS WHITE F     HISPA… <NA>
 2 CDISCPILOT01 01-701… 1023   701    Plac…    64 YEARS WHITE M     HISPA… <NA>
 3 CDISCPILOT01 01-701… 1028   701    Xano…    71 YEARS WHITE M     NOT H… <NA>
 4 CDISCPILOT01 01-701… 1033   701    Xano…    74 YEARS WHITE M     NOT H… <NA>
 5 CDISCPILOT01 01-701… 1034   701    Xano…    77 YEARS WHITE F     NOT H… <NA>
 6 CDISCPILOT01 01-701… 1047   701    Plac…    85 YEARS WHITE F     NOT H… <NA>
 7 CDISCPILOT01 01-701… 1057   701    Scre…    59 YEARS WHITE F     HISPA… <NA>
 8 CDISCPILOT01 01-701… 1097   701    Xano…    68 YEARS WHITE M     NOT H… <NA>
 9 CDISCPILOT01 01-701… 1111   701    Xano…    81 YEARS WHITE F     NOT H… <NA>
10 CDISCPILOT01 01-701… 1115   701    Xano…    84 YEARS WHITE M     NOT H… <NA>
# ℹ 3 more variables: RFSTDTC <chr>, RFENDTC <chr>, TRT01P <chr>
```

Now we have the base dataset, we can start to create some variables. We can start with creating the subgroups using the controlled terminology, in this case `AGEGR1`. The metacore object holds all the metadata needed to make `ADSL`. Part of that metadata is the controlled terminology, which can help automate the creation of subgroups. We can look into the metacore object and see the controlled terminology for `AGEGR1`.

```
1 get_control_term(metacore, variable = AGEGR1)
```

```
# A tibble: 3 × 2
  code  decode
  <chr> <chr>
1 <65   <65
2 65-80 65-80
3 >80   >80
```

Because this controlled terminology is written in a fairly standard format we can automate the creation of `AGEGR1`. The function `metatools::create_cat_var()` takes in a metacore object, a reference variable - in this case `AGE` because that is the continuous variable `AGEGR1` is created from, and the name of the sub-grouped variable. It will take the controlled terminology reference variables accordingly.

Using a similar philosophy we can create the numeric ve in the metacore object with the `metatools::create_v`

```
1  adsl_ct <- adsl_preds %>%
2     create_cat_var(metacore, ref_var = A
3                    grp_var = AGEGR1, num
4     create_var_from_codelist(metacore =
5                              input_var =
6                              out_var = F
7     #Removing screen failures from ARM a
8     mutate(ARM = if_else(ARM == "Screen
9            TRT01P = if_else(TRT01P == "S
10    )
```

```
11
12 head(adsl_ct, n=10)
```

```
# A tibble: 10 × 17
   STUDYID      USUBJID SUBJID SITEID ARM     AGE AGEU  RACE  SEX   ETHNIC DTHFL
   <chr>        <chr>   <chr>  <chr>  <chr> <dbl> <chr> <chr> <chr> <chr>  <chr>
 1 CDISCPILOT01 01-701… 1015   701    Plac…    63 YEARS WHITE F     HISPA… <NA>
 2 CDISCPILOT01 01-701… 1023   701    Plac…    64 YEARS WHITE M     HISPA… <NA>
 3 CDISCPILOT01 01-701… 1028   701    Xano…    71 YEARS WHITE M     NOT H… <NA>
 4 CDISCPILOT01 01-701… 1033   701    Xano…    74 YEARS WHITE M     NOT H… <NA>
 5 CDISCPILOT01 01-701… 1034   701    Xano…    77 YEARS WHITE F     NOT H… <NA>
 6 CDISCPILOT01 01-701… 1047   701    Plac…    85 YEARS WHITE F     NOT H… <NA>
 7 CDISCPILOT01 01-701… 1057   701    <NA>     59 YEARS WHITE F     HISPA… <NA>
 8 CDISCPILOT01 01-701… 1097   701    Xano…    68 YEARS WHITE M     NOT H… <NA>
 9 CDISCPILOT01 01-701… 1111   701    Xano…    81 YEARS WHITE F     NOT H… <NA>
10 CDISCPILOT01 01-701… 1115   701    Xano…    84 YEARS WHITE M     NOT H… <NA>
# ℹ 6 more variables: RFSTDTC <chr>, RFENDTC <chr>, TRT01P <chr>, AGEGR1 <chr>,
#   AGEGR1N <dbl>, RACEN <dbl>
```

Now we have sorted out what we can easily do with controlled terminology it is time to start deriving some variables. Here you could refer directly to using the admiral template and vignette in practice, but for the purpose of this end-to-end ADaM vignette we will share a few exposure derivations from there. We derive the start and end of treatment (which requires dates to first be converted from DTC to DTM), the treatment duration, and the safety population flag.

The provided R script snippet is used for data manipulation to create an ADSL dataset from an ADSL raw dataset. Here is a detailed breakdown of its purpose and functionality:

Purpose

1. The script processes the ex dataset to derive new datetime variables (EXSTDTM and EXENDTM) from the date character variables (EXSTDTC and EXENDTC)

2. The script primarily focuses on merging additional
to derive treatment start and end dates, converting
treatment duration, and flagging subjects for safety

Breakdown of the Script

2. **First Merge (TRTSDTM)**:
    - `derive_vars_merged`:
        - **`dataset_add = ex_ext`**: Merges with ar
        - **`filter_add`**: Filters records where `EXDO`
          "PLACEBO" and ensures the date is prope
        - **`new_vars = exprs(TRTSDTM = EXSTDTM`**
        - **`order = exprs(EXSTDTM, EXSEQ)`**: Ord
        - **`mode = "first"`**: Keeps the first occurre
        - **`by_vars = exprs(STUDYID, USUBJID)`**
3. **Second Merge (TRTEDTM)**:

- Another `derive_vars_merged`:
  - Similar to the first merge but creates `TRTEDTM` from `EXENDTM`.
  - Uses `order = exprs(EXENDTM, EXSEQ)` and `mode = "last"` to get the last occurrence.

4. **Datetime to Date Conversion**:
   - `derive_vars_dtm_to_dt(source_vars = exprs(TRTSDTM, TRTEDTM))`:
     - Converts `TRTSDTM` and `TRTEDTM` datetime variables to date variables.
5. **Treatment Duration**:
   - `derive_var_trtdurd()`:
     - Calculates treatment duration (`TRTDURD`) using the start and end dates.
6. **Safety Analysis Flag (SAFFL)**:
   - `derive_var_merged_exist_flag`:
     - **`dataset_add = ex`**: Checks existence in the EX dataset.
     - **`by_vars = exprs(STUDYID, USUBJID)`**: Merges by `STUDYID` and `USUBJID`.
     - **`new_var = SAFFL`**: Creates a new flag variable `SAFFL`.
     - **`condition`**: Flags records where `EXDOSE > 0` or `EXDOSE == 0` and `EXTRT` contains "PLACEBO".
7. **Drop Unspecified Variables**:
   - `drop_unspec_vars(metacore)`:
     - Drops columns that are not specified in the `metacore` object.

Summary

The script enhances the ADSL dataset with key variables like treatment start and end dates (`TRTSDTM`, `TRTEDTM`), calculates treatment duration (`TRTDURD`), flags subjects for safety analysis (`SAFFL`), and ensures only specified variables are kept. This manipulation is essential for generating accurate and comprehensive datasets for clinical analysis and reporting.

```
ex_ext <- ex %>%
  derive_vars_dtm(
    dtc = EXSTDTC,
    new_vars_prefix = "EXST"
  ) %>%
  derive_vars_dtm(
    dtc = EXENDTC,
    new_vars_prefix = "EXEN",
    time_imputation = "last"
  )

adsl_raw <- adsl_ct %>%
  derive_vars_merged(
    dataset_add = ex_ext,
    filter_add = (EXDOSE > 0 |
      (EXDOSE == 0 &
        str_detect(EXTRT, "PLACEBO")))
    new_vars = exprs(TRTSDTM = EXSTDTM)
    order = exprs(EXSTDTM, EXSEQ),
    mode = "first",
```

```
    by_vars = exprs(STUDYID, USUBJID)
  ) %>%
  derive_vars_merged(
    dataset_add = ex_ext,
    filter_add = (EXDOSE > 0 |
      (EXDOSE == 0 &
        str_detect(EXTRT, "PLACEBO"))) & nchar(EXENDTC) >= 10,
    new_vars = exprs(TRTEDTM = EXENDTM),
    order = exprs(EXENDTM, EXSEQ),
    mode = "last",
    by_vars = exprs(STUDYID, USUBJID)
  ) %>%
   derive_vars_dtm_to_dt(source_vars = exprs(TRTSDTM, TRTEDTM)) %>%  #Convert Dateti
   derive_var_trtdurd() %>%
   derive_var_merged_exist_flag(
     dataset_add = ex,
     by_vars = exprs(STUDYID, USUBJID),
     new_var = SAFFL,
     condition = (EXDOSE > 0 | (EXDOSE == 0 & str_detect(EXTRT, "PLACEBO")))
   ) %>%
   drop_unspec_vars(metacore) #This will drop any columns that aren't specified in t
```

```
The following variable(s) were dropped:
  TRTSDTM
  TRTEDTM
```

```
head(adsl_raw, n=10)
```

```
# A tibble: 10 × 21
   STUDYID      USUBJID SUBJID SITEID ARM     AGE AGEU  RACE  SEX   ETHNIC DTHFL
   <chr>        <chr>   <chr>  <chr>  <chr> <dbl> <chr> <chr> <chr> <chr>  <chr>
 1 CDISCPILOT01 01-701… 1015   701    Plac…
 2 CDISCPILOT01 01-701… 1023   701    Plac…
 3 CDISCPILOT01 01-701… 1028   701    Xano…
 4 CDISCPILOT01 01-701… 1033   701    Xano…
 5 CDISCPILOT01 01-701… 1034   701    Xano…
 6 CDISCPILOT01 01-701… 1047   701    Plac…
 7 CDISCPILOT01 01-701… 1057   701    <NA>
 8 CDISCPILOT01 01-701… 1097   701    Xano…
 9 CDISCPILOT01 01-701… 1111   701    Xano…
10 CDISCPILOT01 01-701… 1115   701    Xano…
# ℹ 10 more variables: RFSTDTC <chr>, RFENDTC
#   AGEGR1 <chr>, AGEGR1N <dbl>, RACEN <dbl>,
#   TRTDURD <dbl>, SAFFL <chr>
```

Apply Metadata to Create an eSub Checks

Now we have all the variables defined we can run some checks before applying the necessary formatting. The top four functions performing checks and sorting/ordering come from metatools, whereas the others focused around applying attributes to prepare for XPT come from xportr. At the end you could add a call to `xportr::xportr_write()` to produce the XPT file.

```
adsl_raw %>%
   check_variables(metacore) %>% # Check all variables specified are present and no
   check_ct_data(metacore, na_acceptable = TRUE) %>% # Checks all variables with CT
   order_cols(metacore) %>% # Orders the columns according to the spec
   sort_by_key(metacore) %>% # Sorts the rows by the sort keys
   xportr_type(metacore, domain = "ADSL") %>% # 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_df_label(metacore) # Assigns dataset label from metacore specifications
```

```
# A tibble: 306 × 49
   STUDYID     USUBJID SUBJID SITEID SITEGR1 ARM   TRT01P TRT01PN TRT01A TRT01AN
   <chr>       <chr>   <chr>  <chr>  <chr>   <chr> <chr>    <dbl> <chr>    <dbl>
 1 CDISCPILOT… 01-701… 1015   701    <NA>    Plac… Place…      NA <NA>        NA
 2 CDISCPILOT… 01-701… 1023   701    <NA>    Plac… Place…      NA <NA>        NA
 3 CDISCPILOT… 01-701… 1028   701    <NA>    Xano… Xanom…      NA <NA>        NA
 4 CDISCPILOT… 01-701… 1033   701    <NA>    Xano… Xanom…      NA <NA>        NA
 5 CDISCPILOT… 01-701… 1034   701    <NA>    Xano… Xanom…      NA <NA>        NA
 6 CDISCPILOT… 01-701… 1047   701    <NA>    Plac… Place…      NA <NA>        NA
 7 CDISCPILOT… 01-701… 1057   701    <NA>    <NA>  <NA>        NA <NA>        NA
 8 CDISCPILOT… 01-701… 1097   701    <NA>    Xano… Xanom…      NA <NA>        NA
 9 CDISCPILOT… 01-701… 1111   701    <NA>    Xano… Xanom…      NA <NA>        NA
10 CDISCPILOT… 01-701… 1115   701    <NA>    Xano… Xanom…      NA <NA>        NA
# ℹ 296 more rows
# ℹ 39 more variables: TRTSDT <date>, TRTEDT <date>, TRTDURD <dbl>,
#   AVGDD <dbl>, CUMDOSE <dbl>, AGE <dbl>, AGEGR1 <chr>, AGEGR1N <dbl>,
#   AGEU <chr>, RACE <chr>, RACEN <dbl>, SEX
#   ITTFL <chr>, EFFFL <chr>, COMP8FL <chr>,
#   DISCONFL <chr>, DSRAEFL <chr>, DTHFL <chr
#   HEIGHTBL <dbl>, WEIGHTBL <dbl>, EDUCLVL <
```