Creating ADaM Subject-level Analysis (ADSL) using R admiral package

Lun-Hsien Chang

2024-11-29

Programming workflow

Get required R packages

```
Warning: package 'pharmaversesdtm' was built under R version 4.4.2
```

```
Attaching package: 'lubridate'
```

```
The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

Read CDISC pilot SDTM datasets

Derive treatment variables

TRT01P

- **P**lanned **Tr**ea**t**ment for Period **01**
- Derived from DM.ARM

TRT01A

- **A**ctual **Tr**ea**t**ment for Period **01**
- Derived from DM.ACTARM

```
adsl <- dm %>%
  dplyr::select(-DOMAIN) %>%
  dplyr::mutate(TRT01P = ARM, TRT01A = ACTARM) # dim(adsl) 306 26
```

Derive treatment datetime, duration variables

EXSTDTM

- Numeric start datetime of exposure derived from character EXSTDTC
- Date with missing time can be imputated. e.g., EXSTDTC="2014-01-02" –> EXSTDTM= 2014-01-02 00:00:00

EXSTTMF

EXENDTM

- Numeric end datetime of exposure derived from character EXENDTC
- Date with missing time can be imputed. E.g., EXENDTC="2014-01-16" –> EXENDTM=2014-01-16 23:59:59

EXENTMF

```
# Derive a datetime object --DTM from a date character vector --DTC
ex_ext <- ex %>%
  admiral::derive_vars_dtm(
    dtc = EXSTDTC
    ,new_vars_prefix = "EXST"
    ,time_imputation = "first") %>%
  admiral::derive_vars_dtm(
    dtc = EXENDTC
    ,new_vars_prefix = "EXEN"
    ,time_imputation = "last") # dim(ex_ext) 591 21

ex_ext %>% select(EXSTDTC, EXSTDTM, EXSTTMF, EXENDTC, EXENDTM, EXENTMF) %>% head()
```

```
# A tibble: 6 x 6
  EXSTDTC    EXSTDTM             EXSTTMF EXENDTC    EXENDTM             EXENTMF
  <chr>      <dttm>              <chr>   <chr>      <dttm>              <chr>
1 2014-01-02 2014-01-02 00:00:00 H       2014-01-16 2014-01-16 23:59:59 H
2 2014-01-17 2014-01-17 00:00:00 H       2014-06-18 2014-06-18 23:59:59 H
3 2014-06-19 2014-06-19 00:00:00 H       2014-07-02 2014-07-02 23:59:59 H
4 2012-08-05 2012-08-05 00:00:00 H       2012-08-27 2012-08-27 23:59:59 H
5 2012-08-28 2012-08-28 00:00:00 H       2012-09-01 2012-09-01 23:59:59 H
6 2013-07-19 2013-07-19 00:00:00 H       2013-08-01 2013-08-01 23:59:59 H
```

TRTSDTM

- Datetime of First Exposure to Treatment
- Numeric version of datetime derived from DM.RFXSTDTC (but here it is from EXSTDTM)

TRTEDTM

- Datetime of Last Exposure to Treatment
- Numeric version of datetime derived from DM.RFXENDTC (but here it is from EXENDTM)

```
# Left join adsl and ex_ext
# new variables added: "TRTSDTM" "TRTSTMF"  "TRTEDTM"  "TRTETMF"
adsl <- adsl %>%
  derive_vars_merged(
    dataset_add = ex_ext
    # Observations from dataset_add that meet the conditions will be merged to adsl
    ,filter_add = (EXDOSE > 0 |
                     (EXDOSE == 0 &
                      str_detect(EXTRT, "PLACEBO"))) & !is.na(EXSTDTM)
    ,new_vars = exprs(TRTSDTM = EXSTDTM, TRTSTMF = EXSTTMF)
    ,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"))) & !is.na(EXENDTM),
    new_vars = exprs(TRTEDTM = EXENDTM, TRTETMF = EXENTMF),
    order = exprs(EXENDTM, EXSEQ),
    mode = "last",
    by_vars = exprs(STUDYID, USUBJID)
  ) # dim(adsl) after merging: 306 30 # dim(adsl) before merging: 306 26

# Old variables
ex_ext %>% select(EXSTDTM, EXSTTMF,EXENDTM, EXENTMF) %>% head()
```

```
# A tibble: 6 x 4
  EXSTDTM             EXSTTMF EXENDTM             EXENTMF
  <dttm>              <chr>   <dttm>              <chr>
1 2014-01-02 00:00:00 H       2014-01-16 23:59:59 H
2 2014-01-17 00:00:00 H       2014-06-18 23:59:59 H
3 2014-06-19 00:00:00 H       2014-07-02 23:59:59 H
4 2012-08-05 00:00:00 H       2012-08-27 23:59:59 H
5 2012-08-28 00:00:00 H       2012-09-01 23:59:59 H
6 2013-07-19 00:00:00 H       2013-08-01 23:59:59 H
```

```
# New variables
adsl %>% select(TRTSDTM, TRTSTMF, TRTEDTM, TRTETMF) %>% head()
```

```
# A tibble: 6 x 4
  TRTSDTM             TRTSTMF TRTEDTM             TRTETMF
  <dttm>              <chr>   <dttm>              <chr>
1 2014-01-02 00:00:00 H       2014-07-02 23:59:59 H
2 2012-08-05 00:00:00 H       2012-09-01 23:59:59 H
3 2013-07-19 00:00:00 H       2014-01-14 23:59:59 H
4 2014-03-18 00:00:00 H       2014-03-31 23:59:59 H
5 2014-07-01 00:00:00 H       2014-12-30 23:59:59 H
6 2013-02-12 00:00:00 H       2013-03-09 23:59:59 H
```

TRTSDT

- Date of First Exposure to Treatment
- Numeric version of date portion of DM.RFXSTDTC formatted as a SAS date (But here it is from TRTSDTM)

TRTEDT

- Date of Last Exposure to Treatment
- Numeric version of date portion of DM.RFXENDTC formatted as a SAS date (But here it is from TRTEDTM)

TRTDURD

- Treatment duration
- ‘TRTDURD= TRTEDT- TRTSDT+1‘

```
# New variables added: "TRTSDT" "TRTEDT" "TRTDURD"
adsl <- adsl %>%
  # Derive date variables from datetime variables
  admiral::derive_vars_dtm_to_dt(source_vars = exprs(TRTSDTM, TRTEDTM)) %>%
  # Derives total treatment duration (days) (TRTDURD). TRTDURD= TRTEDT- TRTSDT+1
  admiral::derive_var_trtdurd() # dim(adsl) 306 33

adsl %>% select(TRTSDT, TRTEDT, TRTDURD) %>% head()
```

```
# A tibble: 6 x 3
  TRTSDT     TRTEDT     TRTDURD
  <date>     <date>       <dbl>
1 2014-01-02 2014-07-02     182
2 2012-08-05 2012-09-01      28
3 2013-07-19 2014-01-14     180
4 2014-03-18 2014-03-31      14
5 2014-07-01 2014-12-30     183
6 2013-02-12 2013-03-09      26
```

Derive disposition variables

DSSTDT

- Convert character disposition date DS.DSSTDTC to numeric date DSSTDT using `derive_vars_dt()`

EOSDT

- End of Study Date
- Numeric version of DS.DSSTDTC or data cutoff date

```
# New variable added: DSSTDT
ds_ext <- admiral::derive_vars_dt(dataset = ds # dim(ds) 850 13
                                  ,dtc = DSSTDTC
                                  ,new_vars_prefix = "DSST") # dim(ds_ext) 850 14
ds_ext %>% select(DSSTDTC, DSSTDT) %>% tail()
```

```
# A tibble: 6 x 2
  DSSTDTC    DSSTDT
  <chr>      <date>
1 2013-08-01 2013-08-01
2 2013-08-08 2013-08-08
3 2012-12-17 2012-12-17
4 2013-02-18 2013-02-18
5 2013-02-18 2013-02-18
6 2013-06-03 2013-06-03
```

```
# Check protocol milestones
ds_ext %>% filter(DSCAT=="PROTOCOL MILESTONE") %>% distinct(DSDECOD)
```

```
# A tibble: 1 x 1
  DSDECOD
  <chr>
1 RANDOMIZED
```

```r
# Check disposition events
ds_ext %>% filter(DSCAT=="DISPOSITION EVENT") %>% distinct(DSDECOD)
```

```
# A tibble: 10 x 1
   DSDECOD
   <chr>
 1 COMPLETED
 2 ADVERSE EVENT
 3 STUDY TERMINATED BY SPONSOR
 4 SCREEN FAILURE
 5 DEATH
 6 WITHDRAWAL BY SUBJECT
 7 PHYSICIAN DECISION
 8 PROTOCOL VIOLATION
 9 LOST TO FOLLOW-UP
10 LACK OF EFFICACY
```

```r
# Left join adsl and ds_ext
# New variable added: EOSDT
adsl <- admiral::derive_vars_merged(
  dataset=adsl # dim(adsl) 306 33
  ,dataset_add = ds_ext
  ,by_vars = exprs(STUDYID, USUBJID)
  ,new_vars = exprs(EOSDT = DSSTDT)
  ,filter_add = DSCAT == "DISPOSITION EVENT" & DSDECOD != "SCREEN FAILURE") # dim(adsl) 306 34

adsl %>% select(USUBJID, EOSDT) %>% tail()
```

```
# A tibble: 6 x 2
  USUBJID     EOSDT
  <chr>       <date>
1 01-718-1250 2014-02-08
2 01-718-1254 2014-01-09
3 01-718-1328 2013-05-01
4 01-718-1355 2013-08-29
5 01-718-1371 2013-08-08
6 01-718-1427 2013-02-18
```

EOSSTT

- Subject's status as of the end of study or data cutoff. Examples: COMPLETED, DISCONTINUED, ONGOING.
- Derived based on DS.DSCAT and DS.DSDECOD

```r
# Example function format_eosstt():
format_eosstt <- function(x) {
  case_when(
    x %in% c("COMPLETED") ~ "COMPLETED"
    ,x %in% c("SCREEN FAILURE") ~ NA_character_
    ,TRUE ~ "DISCONTINUED")
}

# New variables added: EOSSTT (End of Study Status)
adsl <- adsl %>%
  derive_vars_merged(
     dataset_add = ds
    ,by_vars = exprs(STUDYID, USUBJID)
    ,filter_add = DSCAT == "DISPOSITION EVENT"
    ,new_vars = exprs(EOSSTT = format_eosstt(DSDECOD))
    ,missing_values = exprs(EOSSTT = "ONGOING")
    ) # dim(adsl) 306 34 before merging # dim(adsl) 306 35 after merging

adsl %>% select(USUBJID, EOSDT,EOSSTT) %>% tail()
```

```
# A tibble: 6 x 3
  USUBJID     EOSDT      EOSSTT
  <chr>       <date>     <chr>
1 01-718-1250 2014-02-08 DISCONTINUED
2 01-718-1254 2014-01-09 COMPLETED
3 01-718-1328 2013-05-01 DISCONTINUED
4 01-718-1355 2013-08-29 COMPLETED
5 01-718-1371 2013-08-08 DISCONTINUED
6 01-718-1427 2013-02-18 DISCONTINUED
```

DCSREAS

- Reason for Discontinuation from Study
- If DS.DSDECOD <> "COMPLETED where DSSCAT ="STUDY PARTICIPATION" (i.e. ADSL.EOSSTT is "DISCONTINUED") then ADSL.DCSREAS = DS.DSDECOD; If DS.DSDECOD = "COMPLETED" where DSSCAT = "STUDY PARTICIPATION", then ADSL.DCSREAS is ; If there is no DS record where DSSCAT = "STUDY PARTICIPATION" (i.e. EOSSTT is "ONGOING") then ADSL.DCSREAS is null.

DCSREASP

- Reason Specified for Discontinuation from Study

- If DS.DSDECOD <> “COMPLETED” where DS.DSSCAT = “STUDY PARTICIPATION” (i.e. ADSL.EOSSTT is “DISCONTINUED”) CO.COVAL / CO.COVAL1 where COREF = “PRIMARY REASON FOR STUDY DISCONTINUATION” (if populated); otherwise ADSL.DCSREASP is null.

```
adsl <- adsl %>%
  derive_vars_merged(
    dataset_add = ds
    ,by_vars = exprs(USUBJID)
    ,new_vars = exprs(DCSREAS = DSDECOD, DCSREASP = DSTERM)
    ,filter_add = DSCAT == "DISPOSITION EVENT" &
      !(DSDECOD %in% c("SCREEN FAILURE", "COMPLETED", NA))
    ) # dim(adsl) 306 35 before merging # dim(adsl) 306 37 after merging

adsl %>% select(USUBJID,EOSDT,EOSSTT,DCSREAS,DCSREASP) %>% head()
```

```
# A tibble: 6 x 5
  USUBJID     EOSDT      EOSSTT       DCSREAS                    DCSREASP
  <chr>       <date>     <chr>        <chr>                      <chr>
1 01-701-1015 2014-07-02 COMPLETED    <NA>                       <NA>
2 01-701-1023 2012-09-02 DISCONTINUED ADVERSE EVENT              ADVERSE EVENT
3 01-701-1028 2014-01-14 COMPLETED    <NA>                       <NA>
4 01-701-1033 2014-04-14 DISCONTINUED STUDY TERMINATED BY SPONSOR SPONSOR DECIS~
5 01-701-1034 2014-12-30 COMPLETED    <NA>                       <NA>
6 01-701-1047 2013-03-29 DISCONTINUED ADVERSE EVENT              ADVERSE EVENT
```

RANDDT

- Date of Randomization
- DS.DSSTDTC is a character (text) variable with date in ISO 8601 format: YYYY-MM-DD (e.g. 1997-07-16). ADSL.RANDDT is the DS.DSSTDTC where DSDECOD = “RANDOMIZED”, SAS date format DATE11.; If a subject was not randomized (e.g. Screen Failure) and there is no record in DS for the subject where DSDECOD = “Randomized” then ADSL.RANDDT is null.

```
adsl <- adsl %>%
  derive_vars_merged(
    dataset_add = ds_ext
    ,filter_add = DSDECOD == "RANDOMIZED"
    ,by_vars = exprs(STUDYID, USUBJID)
    ,new_vars = exprs(RANDDT = DSSTDT)
  )
adsl %>% select(USUBJID,RANDDT) %>% head()
```

```
# A tibble: 6 x 2
  USUBJID     RANDDT
  <chr>       <date>
1 01-701-1015 2014-01-02
2 01-701-1023 2012-08-05
3 01-701-1028 2013-07-19
4 01-701-1033 2014-03-18
5 01-701-1034 2014-07-01
6 01-701-1047 2013-02-12
```

Derive death variables

DTHDT

- Death date
- Convert character DM.DTHDTC to numeric DTHDT

```
adsl <- adsl %>%
  derive_vars_dt(
    new_vars_prefix = "DTH"
    ,dtc = DTHDTC
    #,date_imputation = "first"
    ) # dim(adsl) 306 39

adsl %>% select(USUBJID,TRTEDT, DTHDTC, DTHDT) %>% filter(!is.na(DTHDT)) %>% head()
```

```
# A tibble: 3 x 4
  USUBJID     TRTEDT     DTHDTC     DTHDT
  <chr>       <date>     <chr>      <date>
1 01-701-1211 2013-01-12 2013-01-14 2013-01-14
2 01-704-1445 2014-11-01 2014-11-01 2014-11-01
3 01-710-1083 2013-08-01 2013-08-02 2013-08-02
```

DTHCAUS

- Cause of death
- if the date of death is collected in the AE form when the AE is Fatal, the cause of death would be set to the preferred term (`AEDECOD`) of that Fatal AE, while if the date of death is collected in the `DS` form, the cause of death would be set to the disposition term (`DSTERM`). To achieve this, the ‘`event()`‘ objects within ‘`derive_vars_extreme_event()`‘ must be specified and defined such that they fit the study requirement.

DTHDOM

- Death Domain
- Store the domain where the date of death is collected

DTHSEQ

- Death Sequence Number
- Store the xxSEQ value of that domain

```
# New variables: DTHCAUS, DTHDOM, DTHSEQ
adsl <- adsl %>%
  #select(-DTHCAUS) %>% # remove it before deriving it again
  derive_vars_extreme_event(
    by_vars = exprs(STUDYID, USUBJID),
    events = list(
      event(
        dataset_name = "ae",
        condition = AEOUT == "FATAL",
        set_values_to = exprs(DTHCAUS = AEDECOD, DTHDOM = "AE", DTHSEQ = AESEQ),
      ),
      event(
        dataset_name = "ds",
        condition = DSDECOD == "DEATH" & grepl("DEATH DUE TO", DSTERM),
        set_values_to = exprs(DTHCAUS = DSTERM, DTHDOM = "DS", DTHSEQ = DSSEQ),
      )
    ),
    source_datasets = list(ae = ae, ds = ds),
    tmp_event_nr_var = event_nr,
    order = exprs(event_nr),
    mode = "first",
    new_vars = exprs(DTHCAUS, DTHDOM, DTHSEQ)
  ) # dim(adsl) 306 42

adsl %>% select(USUBJID, DTHDT, DTHCAUS, DTHDOM, DTHSEQ) %>% filter(!is.na(DTHDT)) %>% head()
```

```
# A tibble: 3 x 5
  USUBJID     DTHDT      DTHCAUS               DTHDOM DTHSEQ
  <chr>       <date>     <chr>                 <chr>   <dbl>
1 01-701-1211 2013-01-14 SUDDEN DEATH          AE          9
2 01-704-1445 2014-11-01 COMPLETED SUICIDE     AE          1
3 01-710-1083 2013-08-02 MYOCARDIAL INFARCTION AE          1
```

Following the derivation of DTHCAUS and related traceability variables, it is then possible to derive grouping variables such as death categories (DTHCGRx) using standard tidyverse code.

```r
adsl <- adsl %>%
  mutate(DTHCGR1 = case_when(
    is.na(DTHDOM) ~ NA_character_,
    DTHDOM == "AE" ~ "ADVERSE EVENT",
    str_detect(DTHCAUS, "(PROGRESSIVE DISEASE|DISEASE RELAPSE)") ~ "PROGRESSIVE DISEASE",
    TRUE ~ "OTHER"
  )) # dim(adsl) 306 43

adsl %>% filter(!is.na(DTHDT)) %>% select(USUBJID, DTHCAUS, DTHDOM, DTHCGR1) %>% head()
```

```
# A tibble: 3 x 4
  USUBJID     DTHCAUS                DTHDOM DTHCGR1
  <chr>       <chr>                  <chr>  <chr>
1 01-701-1211 SUDDEN DEATH           AE     ADVERSE EVENT
2 01-704-1445 COMPLETED SUICIDE      AE     ADVERSE EVENT
3 01-710-1083 MYOCARDIAL INFARCTION AE     ADVERSE EVENT
```

DTHADY

- Relative Day of Death
- DTHADY=DTHDT-TRTSDT+1

```r
adsl <- adsl %>%
  derive_vars_duration(
    new_var = DTHADY,
    start_date = TRTSDT,
    end_date = DTHDT
  ) # dim(adsl) 306 44

adsl %>% filter(!is.na(DTHDT)) %>% select(USUBJID, TRTSDT, DTHDT, DTHADY) %>% head()
```

```
# A tibble: 3 x 4
  USUBJID     TRTSDT     DTHDT      DTHADY
  <chr>       <date>     <date>      <dbl>
1 01-701-1211 2012-11-15 2013-01-14     61
2 01-704-1445 2014-05-11 2014-11-01    175
3 01-710-1083 2013-07-22 2013-08-02     12
```

LDDTHELD

- Numbers of days from last dose to death
- LDDTHELD=DTHDT-TRTEDT

```r
adsl <- adsl %>%
  derive_vars_duration(
    new_var = LDDTHELD,
    start_date = TRTEDT,
    end_date = DTHDT,
    add_one = FALSE
  ) # dim(adsl) 306 45

adsl %>% filter(!is.na(DTHDT)) %>% select(USUBJID, TRTEDT, DTHDT, LDDTHELD) %>% head()
```

```
# A tibble: 3 x 4
  USUBJID     TRTEDT     DTHDT      LDDTHELD
  <chr>       <date>     <date>        <dbl>
1 01-701-1211 2013-01-12 2013-01-14        2
2 01-704-1445 2014-11-01 2014-11-01        0
3 01-710-1083 2013-08-01 2013-08-02        1
```

LSTALVDT

- Last Date Known Alive
- Similarly as for the cause of death (`DTHCAUS`), the last known alive date (`LSTALVDT`) can be derived from multiples sources using ‘`derive_vars_extreme_event()`.

```r
adsl <- adsl %>%
  derive_vars_extreme_event(
    by_vars = exprs(STUDYID, USUBJID),
    events = list(
      event(
        dataset_name = "ae",
        order = exprs(AESTDTC, AESEQ),
        condition = !is.na(AESTDTC),
        set_values_to = exprs(
          LSTALVDT = convert_dtc_to_dt(AESTDTC, highest_imputation = "M"),
          seq = AESEQ
        ),
      ),
      event(
        dataset_name = "ae",
        order = exprs(AEENDTC, AESEQ),
        condition = !is.na(AEENDTC),
        set_values_to = exprs(
          LSTALVDT = convert_dtc_to_dt(AEENDTC, highest_imputation = "M"),
          seq = AESEQ
        ),
      ),
```

```
      event(
        dataset_name = "lb",
        order = exprs(LBDTC, LBSEQ),
        condition = !is.na(LBDTC),
        set_values_to = exprs(
          LSTALVDT = convert_dtc_to_dt(LBDTC, highest_imputation = "M"),
          seq = LBSEQ
        ),
      ),
      event(
        dataset_name = "adsl",
        condition = !is.na(TRTEDT),
        set_values_to = exprs(LSTALVDT = TRTEDT, seq = 0),
      )
    ),
    source_datasets = list(ae = ae, lb = lb, adsl = adsl),
    tmp_event_nr_var = event_nr,
    order = exprs(LSTALVDT, seq, event_nr),
    mode = "last",
    new_vars = exprs(LSTALVDT)
  ) # dim(adsl) 306 46

adsl %>% select(USUBJID, TRTEDT, DTHDT, LSTALVDT) %>% head()
```

```
# A tibble: 6 x 4
  USUBJID     TRTEDT     DTHDT  LSTALVDT
  <chr>       <date>     <date> <date>
1 01-701-1015 2014-07-02 NA     2014-07-02
2 01-701-1023 2012-09-01 NA     2012-09-02
3 01-701-1028 2014-01-14 NA     2014-01-14
4 01-701-1033 2014-03-31 NA     2014-04-14
5 01-701-1034 2014-12-30 NA     2014-12-30
6 01-701-1047 2013-03-09 NA     2013-04-07
```

Derive grouping, population variables

AGEGR1

- Pooled Age Group 1
- Study-specific threshold. e.g. If ADSL.AGE < 65, AGEGR1 = "<65"; If ADSL.AGE = 65 or if ADSL.AGE > 65, AGEGR1 = ">=65"

REGION1

- Study-specific grouping variable

```
format_agegr1 <- function(var_input) {
  case_when(
    var_input < 18 ~ "<18",
    between(var_input, 18, 64) ~ "18-64",
    var_input > 64 ~ ">64",
    TRUE ~ "Missing"
  )
}

format_region1 <- function(var_input) {
  case_when(
    var_input %in% c("CAN", "USA") ~ "North America",
    !is.na(var_input) ~ "Rest of the World",
    TRUE ~ "Missing"
  )
}

adsl <- adsl %>%
  mutate(
    AGEGR1 = format_agegr1(AGE),
    REGION1 = format_region1(COUNTRY)
  ) # dim(adsl) 306 48

adsl %>% select(USUBJID, AGE, COUNTRY, AGEGR1, REGION1) %>% head()
```

```
# A tibble: 6 x 5
  USUBJID       AGE COUNTRY AGEGR1 REGION1
  <chr>       <dbl> <chr>   <chr>  <chr>
1 01-701-1015    63 USA     18-64  North America
2 01-701-1023    64 USA     18-64  North America
3 01-701-1028    71 USA     >64    North America
4 01-701-1033    74 USA     >64    North America
5 01-701-1034    77 USA     >64    North America
6 01-701-1047    85 USA     >64    North America
```

SAFFL

- Safety Population Flag
- These flags identify whether or not the subject is included in the specified population. A minimum of one subject-level population flag variable is required in ADSL. Not all of the indicators listed here need to be included in ADSL. As stated in Section 3.1.4, Item 2, only those indicators corresponding to populations defined in the statistical analysis plan or populations used as a basis for analysis need be

included in ADSL. This list of flags is not meant to be all-inclusive. Additional population flags may be added. The values of subject-level population flags cannot be blank. If a flag is used, the corresponding numeric version (*FN, where 0=no and 1=yes) of the population flag can also be included. Please also refer to Section 3.1.4.

- Since the populations flags are mainly company/study specific no dedicated functions are provided, but in most cases they can easily be derived using `derive_var_merged_exist_flag`.

```
adsl <- adsl %>%
  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")))
  ) # dim(adsl) 306 49

adsl %>% select(USUBJID, ARM, ACTARM, SAFFL) %>% head()
```

```
# A tibble: 6 x 4
  USUBJID     ARM                  ACTARM               SAFFL
  <chr>       <chr>                <chr>                <chr>
1 01-701-1015 Placebo              Placebo              Y
2 01-701-1023 Placebo              Placebo              Y
3 01-701-1028 Xanomeline High Dose Xanomeline High Dose Y
4 01-701-1033 Xanomeline Low Dose  Xanomeline Low Dose  Y
5 01-701-1034 Xanomeline High Dose Xanomeline High Dose Y
6 01-701-1047 Placebo              Placebo              Y
```

References

Creating ADSL

ADaM Subject-level Analysis - ADSL Dataset