Creating ADSL

Introduction

This article describes creating an `ADSL` ADaM. Examples are currently presented and tested using `DM`, `EX` , `AE`, `LB` and `DS` SDTM domains. However, other domains could be used.

Note: *All examples assume CDISC SDTM and/or ADaM format as input unless otherwise specified.*

Programming Flow

- Read in Data
- Derive Period, Subperiod, and Phase Variables (e.g. `APxxSDT`, `APxxEDT`, ...)
- Derive Treatment Variables (`TRT0xP`, `TRT0xA`)
- Derive/Impute Numeric Treatment Date/Time and Duration (`TRTSDT`, `TRTEDT`, `TRTDURD`)
- Derive Disposition Variables
  - Disposition Dates (e.g. `EOSDT`)
  - Disposition Status (e.g. `EOSTT`)
  - Disposition Reason(s) (e.g. `DCSREAS`, `DCSREASP`)
  - Randomization Date (`RANDDT`)
- Derive Death Variables
  - Death Date (`DTHDT`)
  - Cause of Death (`DTHCAUS`)
  - Duration Relative to Death
- Derive Last Known Date Alive (`LSTALVDT`)
- Derive Groupings and Populations
  - Grouping (e.g. `AGEGR1` or `REGION1`)
  - Population Flags (e.g. `SAFFL`)
- Derive Other Variables
- Add Labels and Attributes

Read in Data

To start, all data frames needed for the creation of `ADSL` should be read into the environment. This will be a company specific process. Some of the data frames needed may be `DM`, `EX`, `DS`, `AE`, and `LB`.

For example purpose, the CDISC Pilot SDTM datasets—which are included in `{pharmaversesdtm}`—are used.

```
library(admiral)
library(dplyr, warn.conflicts = FALSE)
library(pharmaversesdtm)
library(lubridate)
library(stringr)

dm <- pharmaversesdtm::dm
ds <- pharmaversesdtm::ds
ex <- pharmaversesdtm::ex
ae <- pharmaversesdtm::ae
lb <- pharmaversesdtm::lb
```

```
dm <- convert_blanks_to_na(dm)
ds <- convert_blanks_to_na(ds)
ex <- convert_blanks_to_na(ex)
ae <- convert_blanks_to_na(ae)
lb <- convert_blanks_to_na(lb)
```

The `DM` domain is used as the basis for `ADSL`:

```
adsl <- dm %>%
  select(-DOMAIN)
```

USUBJID	RFSTDTC	COUNTRY	AGE	SEX	RACE	ETHNIC	ARM	ACTARM
01-701-1015	2014-01-02	USA	63	F	WHITE	HISPANIC OR LATINO	Placebo	Placebo
01-701-1023	2012-08-05	USA	64	M	WHITE	HISPANIC OR LATINO	Placebo	Placebo
01-701-1028	2013-07-19	USA	71	M	WHITE	NOT HISPANIC OR LATINO	Xanomeline High Dose	Xanomeline High Dose
01-701-1033	2014-03-18	USA	74	M	WHITE	NOT HISPANIC OR LATINO	Xanomeline Low Dose	Xanomeline Low Dose
01-701-1034	2014-07-01	USA	77	F	WHITE	NOT HISPANIC OR LATINO	Xanomeline High Dose	Xanomeline High Dose
01-701-1047	2013-02-12	USA	85	F	WHITE	NOT HISPANIC OR LATINO	Placebo	Placebo
01-701-1057	NA	USA	59	F	WHITE	HISPANIC OR LATINO	Screen Failure	Screen Failure
01-701-1097	2014-01-01	USA	68	M	WHITE	NOT HISPANIC OR LATINO	Xanomeline Low Dose	Xanomeline Low Dose
01-701-1111	2012-09-07	USA	81	F	WHITE	NOT HISPANIC OR LATINO	Xanomeline Low Dose	Xanomeline Low Dose
01-701-1115	2012-11-30	USA	84	M	WHITE	NOT HISPANIC OR LATINO	Xanomeline Low Dose	Xanomeline Low Dose

Derive Period, Subperiod, and Phase Variables (e.g. `APxxSDT`, `APxxEDT`, ...)

See the "Visit and Period Variables" vignette for more information.

If the variables are not derived based on a period reference dataset, they may be derived at a later point of the flow. For example, phases like "Treatment Phase" and "Follow up" could be derived based on treatment start and end date.

Derive Treatment Variables (`TRT0xP`, `TRT0xA`)

The mapping of the treatment variables is left to the ADaM programmer. An example mapping for a study without periods may be:

```
adsl <- dm %>%
  mutate(TRT01P = ARM, TRT01A = ACTARM)
```

For studies with periods see the "Visit and Period Variables" vignette.

Derive/Impute Numeric Treatment Date/Time and Duration (`TRTSDTM`, `TRTEDTM`, `TRTDURD`)

The function `derive_vars_merged()` can be used to derive the treatment start and end date/times using the `ex` domain. A pre-processing step for `ex` is required to convert the variable `EXSTDTC` and `EXSTDTC` to datetime variables and impute missing date or time components. Conversion and imputation is done by `derive_vars_dtm()`.

Example calls:

```
# impute start and end time of exposure to first and last respectively,
# do not impute date
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 <- adsl %>%
  derive_vars_merged(
    dataset_add = ex_ext,
    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)
)
```

This call returns the original data frame with the column `TRTSDTM`, `TRTSTMF`, `TRTEDTM`, and `TRTETMF` added. Exposure observations with incomplete date and zero doses of non placebo treatments are ignored. Missing time parts are imputed as first or last for start and end date respectively.

The datetime variables returned can be converted to dates using the `derive_vars_dtm_to_dt()` function.

```
adsl <- adsl %>%
  derive_vars_dtm_to_dt(source_vars = exprs(TRTSDTM, TRTEDTM))
```

Now, that `TRTSDT` and `TRTEDT` are derived, the function `derive_var_trtdurd()` can be used to calculate the Treatment duration (`TRTDURD`).

```
adsl <- adsl %>%
  derive_var_trtdurd()
```

USUBJID	RFSTDTC	TRTSDTM	TRTSDT	TRTEDTM	TRTEDT	TRTDURD
01-701-1015	2014-01-02	2014-01-02	2014-01-02	2014-07-02 23:59:59	2014-07-02	182
01-701-1023	2012-08-05	2012-08-05	2012-08-05	2012-09-01 23:59:59	2012-09-01	28
01-701-1028	2013-07-19	2013-07-19	2013-07-19	2014-01-14 23:59:59	2014-01-14	180
01-701-1033	2014-03-18	2014-03-18	2014-03-18	2014-03-31 23:59:59	2014-03-31	14
01-701-1034	2014-07-01	2014-07-01	2014-07-01	2014-12-30 23:59:59	2014-12-30	183
01-701-1047	2013-02-12	2013-02-12	2013-02-12	2013-03-09 23:59:59	2013-03-09	26
01-701-1057	NA	NA	NA	NA	NA	NA
01-701-1097	2014-01-01	2014-01-01	2014-01-01	2014-07-09 23:59:59	2014-07-09	190
01-701-1111	2012-09-07	2012-09-07	2012-09-07	2012-09-16 23:59:59	2012-09-16	10
01-701-1115	2012-11-30	2012-11-30	2012-11-30	2013-01-23 23:59:59	2013-01-23	55

Derive Disposition Variables

Disposition Dates (e.g. `EOSDT`)

The functions `derive_vars_dt()` and `derive_vars_merged()` can be used to derive a disposition date. First the character disposition date (`DS.DSSTDTC`) is converted to a numeric date (`DSSTDT`) calling `derive_vars_dt()`. The `DS` dataset is extended by the `DSSTDT` variable because the date is required by other derivations, e.g., `RANDDT` as well. Then the relevant disposition date is selected by adjusting the `filter_add` argument.

To add the End of Study date (`EOSDT`) to the input dataset, a call could be:

```
# convert character date to numeric date without imputation
ds_ext <- derive_vars_dt(
  ds,
  dtc = DSSTDTC,
  new_vars_prefix = "DSST"
)
```

```
adsl <- adsl %>%
  derive_vars_merged(
    dataset_add = ds_ext,
    by_vars = exprs(STUDYID, USUBJID),
    new_vars = exprs(EOSDT = DSSTDT),
    filter_add = DSCAT == "DISPOSITION EVENT" & DSDECOD != "SCREEN FAILURE"
  )
```

The `ds_ext` dataset:

USUBJID	DSCAT	DSDECOD	DSTERM	DSSTDT	DSSTDTC
01-701-1015	PROTOCOL MILESTONE	RANDOMIZED	RANDOMIZED	2014-01-02	2014-01-02
01-701-1015	DISPOSITION EVENT	COMPLETED	PROTOCOL COMPLETED	2014-07-02	2014-07-02
01-701-1015	OTHER EVENT	FINAL LAB VISIT	FINAL LAB VISIT	2014-07-02	2014-07-02
01-701-1023	PROTOCOL MILESTONE	RANDOMIZED	RANDOMIZED	2012-08-05	2012-08-05
01-701-1023	DISPOSITION EVENT	ADVERSE EVENT	ADVERSE EVENT	2012-09-02	2012-09-02
01-701-1023	OTHER EVENT	FINAL LAB VISIT	FINAL LAB VISIT	2012-09-02	2012-09-02
01-701-1023	OTHER EVENT	FINAL RETRIEVAL VISIT	FINAL RETRIEVAL VISIT	2013-02-18	2013-02-18
01-701-1028	PROTOCOL MILESTONE	RANDOMIZED	RANDOMIZED	2013-07-19	2013-07-19
01-701-1028	DISPOSITION EVENT	COMPLETED	PROTOCOL COMPLETED	2014-01-14	2014-01-14
01-701-1028	OTHER EVENT	FINAL LAB VISIT	FINAL LAB VISIT	2014-01-14	2014-01-14

The `adsl` dataset:

USUBJID	EOSDT
01-701-1015	2014-07-02
01-701-1023	2012-09-02
01-701-1028	2014-01-14
01-701-1033	2014-04-14
01-701-1034	2014-12-30
01-701-1047	2013-03-29
01-701-1057	NA
01-701-1097	2014-07-09

USUBJID	EOSDT
01-701-1111	2012-09-17
01-701-1115	2013-01-23

The `derive_vars_dt()` function allows to impute partial dates as well. If imputation is needed and missing days are to be imputed to the first of the month and missing months to the first month of the year, set `highest_imputation = "M"`.

Disposition Status (e.g. `EOSSTT`)

The function `derive_vars_merged()` can be used to derive the End of Study status (`EOSSTT`) based on `DSCAT` and `DSDECOD` from `DS`. The relevant observations are selected by adjusting the `filter_add` argument. A function mapping `DSDECOD` values to `EOSSTT` values can be defined and used in the `new_vars` argument. The mapping for the call below is

- `"COMPLETED"` if `DSDECOD == "COMPLETED"`
- `NA_character_` if `DSDECOD` is `"SCREEN FAILURE"`
- `"DISCONTINUED"` otherwise

Example function `format_eosstt()`:

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

The customized mapping function `format_eosstt()` can now be passed to the main function. For subjects without a disposition event the end of study status is set to `"ONGOING"` by specifying the `missing_values` argument.

```
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")
  )
```

USUBJID	EOSDT	EOSSTT
01-701-1015	2014-07-02	COMPLETED
01-701-1023	2012-09-02	DISCONTINUED
01-701-1028	2014-01-14	COMPLETED
01-701-1033	2014-04-14	DISCONTINUED
01-701-1034	2014-12-30	COMPLETED
01-701-1047	2013-03-29	DISCONTINUED

USUBJID	EOSDT	EOSSTT
01-701-1057	NA	NA
01-701-1097	2014-07-09	COMPLETED
01-701-1111	2012-09-17	DISCONTINUED
01-701-1115	2013-01-23	DISCONTINUED

This call would return the input dataset with the column `EOSSTT` added.

If the derivation must be changed, the user can create his/her own function to map `DSDECOD` to a suitable `EOSSTT` value.

Disposition Reason(s) (e.g. `DCSREAS`, `DCSREASP`)

The main reason for discontinuation is usually stored in `DSDECOD` while `DSTERM` provides additional details regarding subject's discontinuation (e.g., description of `"OTHER"`).

The function `derive_vars_merged()` can be used to derive a disposition reason (along with the details, if required) at a specific timepoint. The relevant observations are selected by adjusting the `filter_add` argument.

To derive the End of Study reason(s) (`DCSREAS` and `DCSREASP`), the function will map `DCSREAS` as `DSDECOD`, and `DCSREASP` as `DSTERM` if `DSDECOD` is not `"COMPLETED"`, `"SCREEN FAILURE"`, or `NA`, `NA` otherwise.

```
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))
  )
```

USUBJID	EOSDT	EOSSTT	DCSREAS	DCSREASP
01-701-1015	2014-07-02	COMPLETED	NA	NA
01-701-1023	2012-09-02	DISCONTINUED	ADVERSE EVENT	ADVERSE EVENT
01-701-1028	2014-01-14	COMPLETED	NA	NA
01-701-1033	2014-04-14	DISCONTINUED	STUDY TERMINATED BY SPONSOR	SPONSOR DECISION (STUDY OR PATIENT DISCONTINUED BY THE SPONSOR)
01-701-1034	2014-12-30	COMPLETED	NA	NA
01-701-1047	2013-03-29	DISCONTINUED	ADVERSE EVENT	ADVERSE EVENT
01-701-1057	NA	NA	NA	NA

USUBJID	EOSDT	EOSSTT	DCSREAS	DCSREASP
01-701-1097	2014-07-09	COMPLETED	NA	NA
01-701-1111	2012-09-17	DISCONTINUED	ADVERSE EVENT	ADVERSE EVENT
01-701-1115	2013-01-23	DISCONTINUED	ADVERSE EVENT	ADVERSE EVENT

This call would return the input dataset with the column `DCSREAS` and `DCSREASP` added.

If the derivation must be changed, the user can define that derivation in the `filter_add` argument of the function to map `DSDECOD` and `DSTERM` to a suitable `DCSREAS`/`DCSREASP` value.

The call below maps `DCSREAS` and `DCREASP` as follows:

- `DCSREAS` as `DSDECOD` if `DSDECOD` is not `"COMPLETED"` or `NA`, `NA` otherwise
- `DCSREASP` as `DSTERM` if `DSDECOD` is equal to `OTHER`, `NA` otherwise

```
adsl <- adsl %>%
  derive_vars_merged(
    dataset_add = ds,
    by_vars = exprs(USUBJID),
    new_vars = exprs(DCSREAS = DSDECOD),
    filter_add = DSCAT == "DISPOSITION EVENT" &
      DSDECOD %notin% c("SCREEN FAILURE", "COMPLETED", NA)
  ) %>%
  derive_vars_merged(
    dataset_add = ds,
    by_vars = exprs(USUBJID),
    new_vars = exprs(DCSREASP = DSTERM),
    filter_add = DSCAT == "DISPOSITION EVENT" & DSDECOD %in% "OTHER"
  )
```

USUBJID	EOSDT	EOSSTT	DCSREAS	DCSREASP
01-701-1015	2014-07-02	COMPLETED	NA	NA
01-701-1023	2012-09-02	DISCONTINUED	ADVERSE EVENT	NA
01-701-1028	2014-01-14	COMPLETED	NA	NA
01-701-1033	2014-04-14	DISCONTINUED	STUDY TERMINATED BY SPONSOR	NA
01-701-1034	2014-12-30	COMPLETED	NA	NA
01-701-1047	2013-03-29	DISCONTINUED	ADVERSE EVENT	NA
01-701-1057	NA	NA	NA	NA
01-701-1097	2014-07-09	COMPLETED	NA	NA
01-701-1111	2012-09-17	DISCONTINUED	ADVERSE EVENT	NA
01-701-1115	2013-01-23	DISCONTINUED	ADVERSE EVENT	NA

Randomization Date (`RANDDT`)

The function `derive_vars_merged()` can be used to derive randomization date variable. To map Randomization Date (`RANDDT`), the call would be:

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

This call would return the input dataset with the column `RANDDT` is added.

USUBJID	RANDDT
01-701-1015	2014-01-02
01-701-1023	2012-08-05
01-701-1028	2013-07-19
01-701-1033	2014-03-18
01-701-1034	2014-07-01
01-701-1047	2013-02-12
01-701-1057	NA
01-701-1097	2014-01-01
01-701-1111	2012-09-07
01-701-1115	2012-11-30

Derive Death Variables

Death Date (`DTHDT`)

The function `derive_vars_dt()` can be used to derive `DTHDT`. This function allows the user to impute the date as well.

Example calls:

```
adsl <- adsl %>%
  derive_vars_dt(
    new_vars_prefix = "DTH",
    dtc = DTHDTC
  )
```

USUBJID	TRTEDT	DTHDTC	DTHDT	DTHFL
01-701-1211	2013-01-12	2013-01-14	2013-01-14	Y
01-701-1442	2014-04-26	NA	NA	NA
01-704-1445	2014-11-01	2014-11-01	2014-11-01	Y

USUBJID	TRTEDT	DTHDTC	DTHDT	DTHFL
01-705-1058	NA	NA	NA	NA
01-708-1347	2013-06-18	NA	NA	NA
01-710-1083	2013-08-01	2013-08-02	2013-08-02	Y
01-710-1235	2013-03-27	NA	NA	NA
01-715-1207	2013-05-27	NA	NA	NA
01-718-1172	2013-11-29	NA	NA	NA

This call would return the input dataset with the columns `DTHDT` added and, by default, the associated date imputation flag (`DTHDTF`) populated with the controlled terminology outlined in the ADaM IG for date imputations. If the imputation flag is not required, the user must set the argument `flag_imputation` to `"none"`.

If imputation is needed and the date is to be imputed to the first day of the month/year the call would be:

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

See also Date and Time Imputation.

Cause of Death (DTHCAUS)

The cause of death `DTHCAUS` can be derived using the function `derive_vars_extreme_event()`.

Since the cause of death could be collected/mapped in different domains (e.g. `DS`, `AE`, `DD`), it is important the user specifies the right source(s) to derive the cause of death from.

For example, 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.

An example call to `derive_vars_extreme_event()` would be:

```
adsl <- adsl %>%
  derive_vars_extreme_event(
    by_vars = exprs(STUDYID, USUBJID),
    events = list(
      event(
        dataset_name = "ae",
        condition = AEOUT == "FATAL",
        set_values_to = exprs(DTHCAUS = AEDECOD),
      ),
      event(
        dataset_name = "ds",
        condition = DSDECOD == "DEATH" & grepl("DEATH DUE TO", DSTERM),
        set_values_to = exprs(DTHCAUS = DSTERM),
      )
```

```
    ),
    source_datasets = list(ae = ae, ds = ds),
    tmp_event_nr_var = event_nr,
    order = exprs(event_nr),
    mode = "first",
    new_vars = exprs(DTHCAUS)
  )
```

USUBJID	EOSDT	DTHDTC	DTHDT	DTHCAUS
01-701-1211	2013-01-14	2013-01-14	2013-01-14	SUDDEN DEATH
01-704-1445	2014-11-01	2014-11-01	2014-11-01	COMPLETED SUICIDE
01-710-1083	2013-08-02	2013-08-02	2013-08-02	MYOCARDIAL INFARCTION

The function also offers the option to add some traceability variables (e.g. `DTHDOM` would store the domain where the date of death is collected, and `DTHSEQ` would store the `xxSEQ` value of that domain). The traceability variables should be added to the `event()` calls and included in the `new_vars` parameter of `derive_vars_extreme_event()`.

```
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)
  )
```

USUBJID	TRTEDT	DTHDTC	DTHDT	DTHCAUS	DTHDOM	DTHSEQ
01-701-1211	2013-01-12	2013-01-14	2013-01-14	SUDDEN DEATH	AE	9
01-704-1445	2014-11-01	2014-11-01	2014-11-01	COMPLETED SUICIDE	AE	1
01-710-1083	2013-08-01	2013-08-02	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.

```
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"
  ))
```

Duration Relative to Death

The function `derive_vars_duration()` can be used to derive duration relative to death like the Relative Day of Death (`DTHADY`) or the numbers of days from last dose to death (`LDDTHELD`).

Example calls:

- Relative Day of Death

```
adsl <- adsl %>%
  derive_vars_duration(
    new_var = DTHADY,
    start_date = TRTSDT,
    end_date = DTHDT
  )
```

- Elapsed Days from Last Dose to Death

```
adsl <- adsl %>%
  derive_vars_duration(
    new_var = LDDTHELD,
    start_date = TRTEDT,
    end_date = DTHDT,
    add_one = FALSE
  )
```

USUBJID	TRTEDT	DTHDTC	DTHDT	DTHCAUS	DTHADY	LDDTHELD
01-701-1211	2013-01-12	2013-01-14	2013-01-14	SUDDEN DEATH	61	2
01-704-1445	2014-11-01	2014-11-01	2014-11-01	COMPLETED SUICIDE	175	0
01-710-1083	2013-08-01	2013-08-02	2013-08-02	MYOCARDIAL INFARCTION	12	1

Derive the Last Date Known Alive (`LSTALVDT`)

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()`.

An example could be (DTC dates are converted to numeric dates imputing missing day and month to the first):

```
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)
)
```

USUBJID	TRTEDT	DTHDTC	LSTALVDT
01-701-1015	2014-07-02	NA	2014-07-02
01-701-1023	2012-09-01	NA	2012-09-02
01-701-1028	2014-01-14	NA	2014-01-14
01-701-1033	2014-03-31	NA	2014-04-14
01-701-1034	2014-12-30	NA	2014-12-30
01-701-1047	2013-03-09	NA	2013-04-07
01-701-1097	2014-07-09	NA	2014-07-09

USUBJID	TRTEDT	DTHDTC	LSTALVDT
01-701-1111	2012-09-16	NA	2012-09-17
01-701-1115	2013-01-23	NA	2013-01-23
01-701-1118	2014-09-09	NA	2014-09-09

Traceability variables can be added by specifying the variables in the `set_values_to` parameter of the `event()` function.

```
adsl <- adsl %>%
  select(-LSTALVDT) %>% # created in the previous call
  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"),
          LALVSEQ = AESEQ,
          LALVDOM = "AE",
          LALVVAR = "AESTDTC"
        ),
      ),
      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"),
          LALVSEQ = AESEQ,
          LALVDOM = "AE",
          LALVVAR = "AEENDTC"
        ),
      ),
      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"),
          LALVSEQ = LBSEQ,
          LALVDOM = "LB",
          LALVVAR = "LBDTC"
        ),
      ),
      event(
        dataset_name = "adsl",
        condition = !is.na(TRTEDT),
        set_values_to = exprs(LSTALVDT = TRTEDT, LALVSEQ = NA_integer_, LALVDOM = "ADSL",
LALVVAR = "TRTEDTM"),
      )
    ),
```

```
    source_datasets = list(ae = ae, lb = lb, adsl = adsl),
    tmp_event_nr_var = event_nr,
    order = exprs(LSTALVDT, LALVSEQ, event_nr),
    mode = "last",
    new_vars = exprs(LSTALVDT, LALVSEQ, LALVDOM, LALVVAR)
  )
```

USUBJID	TRTEDT	DTHDTC	LSTALVDT	LALVDOM	LALVSEQ	LALVVAR
01-701-1015	2014-07-02	NA	2014-07-02	ADSL	NA	TRTEDTM
01-701-1023	2012-09-01	NA	2012-09-02	LB	107	LBDTC
01-701-1028	2014-01-14	NA	2014-01-14	ADSL	NA	TRTEDTM
01-701-1033	2014-03-31	NA	2014-04-14	LB	107	LBDTC
01-701-1034	2014-12-30	NA	2014-12-30	ADSL	NA	TRTEDTM
01-701-1047	2013-03-09	NA	2013-04-07	LB	134	LBDTC
01-701-1097	2014-07-09	NA	2014-07-09	ADSL	NA	TRTEDTM
01-701-1111	2012-09-16	NA	2012-09-17	LB	73	LBDTC
01-701-1115	2013-01-23	NA	2013-01-23	ADSL	NA	TRTEDTM
01-701-1118	2014-09-09	NA	2014-09-09	ADSL	NA	TRTEDTM

Derive Groupings and Populations

Grouping (e.g. `AGEGR1` or `REGION1`)

Numeric and categorical variables (`AGE`, `RACE`, `COUNTRY`, etc.) may need to be grouped to perform the required analysis. `{admiral}` does not **currently** have functionality to assist with all required groupings. So, the user will often need to create his/her own function to meet his/her study requirement.

For example, if

- `AGEGR1` is required to categorize `AGE` into `<18`, `18-64` and `>64`, or
- `REGION1` is required to categorize `COUNTRY` in `North America`, `Rest of the World`,

the user defined functions would look like the following:

```
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"
```

```
  )
}
```

These functions are then used in a `mutate()` statement to derive the required grouping variables:

```
adsl <- adsl %>%
  mutate(
    AGEGR1 = format_agegr1(AGE),
    REGION1 = format_region1(COUNTRY)
  )
```

USUBJID	AGE	SEX	COUNTRY	AGEGR1	REGION1
01-701-1015	63	F	USA	18-64	North America
01-701-1023	64	M	USA	18-64	North America
01-701-1028	71	M	USA	>64	North America
01-701-1033	74	M	USA	>64	North America
01-701-1034	77	F	USA	>64	North America
01-701-1047	85	F	USA	>64	North America
01-701-1057	59	F	USA	18-64	North America
01-701-1097	68	M	USA	>64	North America
01-701-1111	81	F	USA	>64	North America
01-701-1115	84	M	USA	>64	North America

Population Flags (e.g. `SAFFL`)

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`.

An example of an implementation could be:

```
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")))
  )
```

USUBJID	TRTSDT	ARM	ACTARM	SAFFL
01-701-1015	2014-01-02	Placebo	Placebo	Y
01-701-1023	2012-08-05	Placebo	Placebo	Y
01-701-1028	2013-07-19	Xanomeline High Dose	Xanomeline High Dose	Y
01-701-1033	2014-03-18	Xanomeline Low Dose	Xanomeline Low Dose	Y
01-701-1034	2014-07-01	Xanomeline High Dose	Xanomeline High Dose	Y

USUBJID	TRTSDT	ARM	ACTARM	SAFFL
01-701-1047	2013-02-12	Placebo	Placebo	Y
01-701-1057	NA	Screen Failure	Screen Failure	NA
01-701-1097	2014-01-01	Xanomeline Low Dose	Xanomeline Low Dose	Y
01-701-1111	2012-09-07	Xanomeline Low Dose	Xanomeline Low Dose	Y
01-701-1115	2012-11-30	Xanomeline Low Dose	Xanomeline Low Dose	Y

Derive Other Variables

The users can add specific code to cover their need for the analysis.

The following functions are helpful for many ADSL derivations:

- `derive_vars_merged()` - Merge Variables from a Dataset to the Input Dataset
- `derive_var_merged_exist_flag()` - Merge an Existence Flag
- `derive_var_merged_summary()` - Merge Summary Variables

See also Generic Functions.

Add Labels and Attributes

Adding labels and attributes for SAS transport files is supported by the following packages:

- metacore: establish a common foundation for the use of metadata within an R session.
- metatools: enable the use of metacore objects. Metatools can be used to build datasets or enhance columns in existing datasets as well as checking datasets against the metadata.
- xportr: functionality to associate all metadata information to a local R data frame, perform data set level validation checks and convert into a transport v5 file(xpt).

NOTE: All these packages are in the experimental phase, but the vision is to have them associated with an End to End pipeline under the umbrella of the pharmaverse. An example of applying metadata and perform associated checks can be found at the pharmaverse E2E example.

Example Script

ADaM	Sourcing Command
ADSL	`use_ad_template("ADSL")`