create_adae.r

Admin

2024-10-23

```
# Name: ADAE
#
# Label: Adverse Event Analysis Dataset
#
# Input: ae, adsl, ex_single

library(admiral)
library(pharmaversesdtm) # Contains example SDTM from the CDISC pilot project

library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

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

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(haven)
library(xportr)
library(metacore)
library(metatools)

# Load source datasets ----

# Use e.g. haven::read_sas to read in .sas7bdat, or other suitable functions
# as needed and assign to the variables below.
# For illustration purposes read in admiral test data

data("ae")
data("ex_single")
data("suppae")
data("admiral_adsl")
adsl <- admiral_adsl


# read in ADSL
# When SAS datasets are imported into R using haven::read_sas(), missing
# character values from SAS appear as "" characters in R, instead of appearing
# as NA values. Further details can be obtained via the following link:
# https://pharmaverse.github.io/admiral/articles/admiral.html#handling-of-missing-values # no
lint

ae <- convert_blanks_to_na(ae)
ex <- convert_blanks_to_na(ex_single)
adsl <- convert_blanks_to_na(adsl)


# Derivations ----

# Get list of ADSL vars required for derivations
adsl_vars <- exprs(TRTSDT, TRTEDT, DTHDT, EOSDT)

adae <- ae %>%
  # join adsl to ae
  derive_vars_merged(
    dataset_add = adsl,
    new_vars = adsl_vars,
    by = exprs(STUDYID, USUBJID)
  ) %>%
  ## Derive analysis start time ----
derive_vars_dtm(
  dtc = AESTDTC,
  new_vars_prefix = "AST",
  highest_imputation = "M",
  min_dates = exprs(TRTSDT)
) %>%
  ## Derive analysis end time ----
derive_vars_dtm(
  dtc = AEENDTC,
  new_vars_prefix = "AEN",
  highest_imputation = "M",
  date_imputation = "last",
```

```
  time_imputation = "last",
  max_dates = exprs(DTHDT, EOSDT)
) %>%
  ## Derive analysis end/start date ----
derive_vars_dtm_to_dt(exprs(ASTDTM, AENDTM)) %>%
  ## Derive analysis start relative day and  analysis end relative day ----
derive_vars_dy(
  reference_date = TRTSDT,
  source_vars = exprs(ASTDT, AENDT)
) %>%
  ## Derive analysis duration (value and unit) ----
derive_vars_duration(
  new_var = ADURN,
  new_var_unit = ADURU,
  start_date = ASTDT,
  end_date = AENDT,
  in_unit = "days",
  out_unit = "days",
  add_one = TRUE,
  trunc_out = FALSE
)

ex_ext <- derive_vars_dtm(
  ex,
  dtc = EXSTDTC,
  new_vars_prefix = "EXST",
  flag_imputation = "none"
)

adae <- adae %>%
  ## Derive last dose date/time ----
derive_vars_joined(
  dataset_add = ex_ext,
  by_vars = exprs(STUDYID, USUBJID),
  new_vars = exprs(LDOSEDTM = EXSTDTM),
  join_vars = exprs(EXSTDTM),
   join_type = "all",
  order = exprs(EXSTDTM),
  filter_add = (EXDOSE > 0 | (EXDOSE == 0 & grepl("PLACEBO", EXTRT))) & !is.na(EXSTDTM),
  filter_join = EXSTDTM <= ASTDTM,
  mode = "last"
) %>%
  ## Derive severity / causality / ... ----
# mutate(
#   ASEV = AESEV,
#   AREL = AEREL
# ) %>%
## Derive treatment emergent flag ----
derive_var_trtemfl(
  trt_start_date = TRTSDT,
  trt_end_date = NULL,
  end_window = NULL
) %>%
  ## Derive occurrence flags: first occurrence of most severe AE ----
# create numeric value ASEVN for severity
mutate(
```

```
  ASEVN = as.integer(factor(AESEV, levels = c("MILD", "MODERATE", "SEVERE", "DEATH THREATENIN
G")))
) %>%
  restrict_derivation(
    derivation = derive_var_extreme_flag,
    args = params(
      by_vars = exprs(USUBJID),
      order = exprs(desc(ASEVN), ASTDTM, AESEQ),
      new_var = AOCCIFL,
      mode = "first"
    ),
    filter = TRTEMFL == "Y"
  )

#Derive query variables

# creating a query dataset for a customized query
cqterms <- tribble(
  ~TERMCHAR, ~TERMNUM,
  "APPLICATION SITE ERYTHEMA", 10003041L,
  "APPLICATION SITE PRURITUS", 10003053L,
) %>%
  mutate(SRCVAR = "AEDECOD")

cqterms_can <- tribble(
  ~TERMCHAR, ~TERMNUM,
  "COLON CANCER", 10000000L,
  "PROSTATE CANCER", 10000001L
) %>%
  mutate(SRCVAR = "AEDECOD")

cq7 <- query(
  prefix = "CQ07",
  name = "Application Site Issues",
  definition = cqterms
)

cq8 <- query(
  prefix = "CQ08",
  name = "Cancer",
  definition = cqterms_can
)

custom <- create_query_data(queries = list(cq7,cq8 ))

data("queries") #get current query data from admiral package

adag <- bind_rows(queries, custom) #combine query data with custom queries created above


#Derive query variables
# data("queries")
adae <- derive_vars_query(dataset = adae, dataset_queries = adag)


# Join all ADSL with AE
```

```
adae <- adae %>%
  derive_vars_merged(
    dataset_add = select(adsl, !!!negate_vars(adsl_vars)),
    new_vars = exprs(STUDYID,
                     USUBJID,
                     SUBJID,
                     SITEID,
                     REGION1,
                     COUNTRY,
                     ETHNIC,
                     AGE,
                     AGEU,
                     AGEGR1,
                     SEX,
                     RACE,
                     DTHDTC,
                     SAFFL,
                     TRT01P,
                     TRT01A,
                     TRTSDTM,
                     TRTEDTM),
    by_vars = exprs(STUDYID, USUBJID)
  )


# ASEQ
adae <- adae %>%
  derive_var_obs_number(
    by_vars    = exprs(USUBJID),
    order      = exprs(ASTDTM, AETERM, AESEQ),
    new_var    = ASEQ,
    check_type = "error"
  )


# Read in specifications .xlsx
metacore <- spec_to_metacore(file.path("~/coursera_admiral/specifications",
                                       "coursera_adam_spec.xlsx"), where_sep_sheet = FALSE,
                             quiet = TRUE)
```

```
##
##  Metadata successfully imported
```

```
## Loading in metacore object with suppressed warnings
```

```
adae_spec <- metacore %>% select_dataset("ADAE") # Get the specifications for the dataset we
are currently building
adae_spec_tbl <- metacore %>% select_dataset("ADAE", simplify=T) # Get the specifications for
the dataset we are currently building

# Final ADAE ----------------------------------------------------
adae_final <- adae %>%
  drop_unspec_vars(adae_spec) %>%  # Check all variables specified are present and no more
  check_variables(adae_spec) %>%
  order_cols(adae_spec) %>% # Orders the columns according to the spec
  sort_by_key(adae_spec) %>% # Sorts the rows by the sort keys
  xportr_length(adae_spec) %>% # Assigns SAS length from a variable level metadata
  xportr_label(adae_spec) %>% # Assigns variable label from metacore specifications
  xportr_format(adae_spec) %>%
  check_ct_data(adae_spec, na_acceptable = TRUE) # Checks all variables with CT only contain
values within the CT
```

```
## The following variable(s) were dropped:
##   AEDTC
##   DTHDT
##   EOSDT
##   SMQ02NAM
##   SMQ02CD
##   SMQ02SC
##   SMQ02SCN
##   SMQ03NAM
##   SMQ03CD
##   SMQ03SC
##   SMQ03SCN
##   SMQ05NAM
##   SMQ05CD
##   SMQ05SC
```

```
## No missing or extra variables
```

```
# Export to xpt
#-------------------------------------------------------------------------------------------
adae_final %>%
  xportr_write(file.path("~/coursera_admiral/adam", "adae.xpt"))
```