Package: admiralvaccine (via r-universe)

September 26, 2024

Type Package

Title Vaccine Extension Package for ADaM in 'R' Asset Library

Version 0.3.0

Description Programming vaccine specific Clinical Data Interchange Standards Consortium (CDISC) compliant Analysis Data Model (ADaM) datasets in 'R'. Flat model is followed as per Center for Biologics Evaluation and Research (CBER) guidelines for creating vaccine specific domains. ADaM datasets are a mandatory part of any New Drug or Biologics License Application submitted to the United States Food and Drug Administration (FDA). Analysis derivations are implemented in accordance with the ``Analysis Data Model Implementation Guide" (CDISC Analysis Data Model Team (2021), <https://www.cdisc.org/standards/foundational/adam/adamig-v1-3-release-package>). The package is an extension package of the 'admiral' package.

License Apache License (>= 2)

URL https://pharmaverse.github.io/admiralvaccine/, https://github.com/pharmaverse/admiralvaccine/

BugReports https://github.com/pharmaverse/admiralvaccine/issues/

Depends R (>= 3.5)

Imports admiral (>= 1.0.0), admiraldev (>= 1.0.0), assertthat (>= 0.2.1), dplyr (>= 0.8.4), hms (>= 0.5.3), lifecycle (>= 0.1.0), lubridate (>= 1.7.4), magrittr (>= 1.5), purrr (>= 0.3.3), rlang (>= 0.4.4), stringr (>= 1.4.0), tidyr (>= 1.0.2), tidyselect (>= 1.0.0)

Suggests covr, devtools, diffdf, DT, knitr, lintr, methods, miniUI, pharmaversesdtm, pkgdown, rmarkdown, roxygen2, spelling, testthat, tibble, usethis, metatools

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Repository https://pharmaverse.r-universe.dev

RemoteUrl https://github.com/pharmaverse/admiralvaccine

RemoteRef HEAD

RemoteSha 5bf3c69d701ea2ebac9e5c0e34cea7eec2d55a5f

Contents

`admiralvaccine_adce` *Clinical Events Analysis Dataset - Vaccine Specific*

Description

An example Clinical Events analysis dataset

Usage

```
admiralvaccine_adce
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 44 rows and 56 columns.

Source

(https://github.com/pharmaverse/admiralvaccine/blob/main/inst/templates/ad_adce.R)

See Also

Other dataset: admiralvaccine_adface, admiralvaccine_adis, admiralvaccine_adsl

---

admiralvaccine_adface *Findings About Clinical Events Analysis Dataset - Vaccine Specific*

Description

An example Findings About Clinical Events analysis dataset

Usage

```
admiralvaccine_adface
```

Format

An object of class tbl_df (inherits from tbl, data.frame) with 371 rows and 60 columns.

Source

(https://github.com/pharmaverse/admiralvaccine/blob/main/inst/templates/ad_adface.R)

See Also

Other dataset: admiralvaccine_adce, admiralvaccine_adis, admiralvaccine_adsl

---

admiralvaccine_adis *Immunogenicity Specimen Assessments Analysis Dataset - Vaccine Specific*

Description

An example Immunogenicity Specimen Assessments analysis dataset

Usage

```
admiralvaccine_adis
```

Format

An object of class tbl_df (inherits from tbl, data.frame) with 64 rows and 102 columns.

Source

(https://github.com/pharmaverse/admiralvaccine/blob/main/inst/templates/ad_adis.R)

See Also

Other dataset: admiralvaccine_adce, admiralvaccine_adface, admiralvaccine_adsl

---

`admiralvaccine_adsl` *Subject Level Analysis Dataset - Vaccine Specific*

Description

An example Subject Level analysis dataset

Usage

```
admiralvaccine_adsl
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 2 rows and 46 columns.

Source

(https://github.com/pharmaverse/admiralvaccine/blob/main/inst/templates/ad_adsl.R)

See Also

Other dataset: admiralvaccine_adce, admiralvaccine_adface, admiralvaccine_adis

---

`derive_diam_to_sev_records`
Creating Severity Records From Diameter

Description

To derive the severity records from the diameter records.

Usage

```
derive_diam_to_sev_records(
  dataset,
  filter_add = NULL,
  diam_code = "DIAMETER",
  faobj_values = c("REDNESS", "SWELLING"),
  testcd_sev = "SEV",
  test_sev = "Severity/Intensity",
  none = 0,
  mild = 2,
  mod = 5,
  sev = 10
)
```

Arguments

dataset	Input data set The variables USUBJID,FAOBJ,AVAL, AVALC, FATESTCD and FATEST are expected for Input data set.
filter_add	filter for the dataset.
diam_code	Diameter record filter *Permitted Value*: A character vector or scalar. Helps to filter the diameter records to derive the severity records by passing the FATESTCD value for diameter which is corresponding to the specified events in faobj_values.
faobj_values	Event filter *Permitted Value*: A character vector or Scalar. Helps to filter the events (Redness and Swelling) which has diameter records to derive severity records by passing the events from FAOBJ.
testcd_sev	To assign FATESTCD value for severity *Permitted Value*: A character scalar Assign the value for FATESTCD variable to indicate the severity records. Ignore the argument if you want to set the default value (SEV).
test_sev	FATEST Value for severity *Permitted Value*: A Character scalar Assign the value for FATEST variable to indicate the severity records. Ignore the argument if you want to set the default value.
none	Pass the lower limit for grade "NONE" *Permitted Value:* A numeric vector The none and the following arguments (mild, mode and sev) will be used for assigning the diameter limit to derive the AVALC (severity grade). *Assign the lower limit to derive the Severity Grade (*AVALC*).* *For Example: User passing 0 to* none *and 2 to* mild*, 0 will act as lower limit and 2 will act as upper limit.*

Note: Use the limit reference to pass the values to these arguments
Since the condition was coded like this,
NONE : `none` < *AVAL* <= `mild`
MILD : `mild` < *AVAL* <= `mod`
MODERATE : `mod` < *AVAL* <= `sev`
SEVERE : `sev` < *AVAL*
User should pass the values as numeric scalar. Refer the default values.

`mild` Pass the lower limit for grade `"MILD"`

Permitted Value: A numeric vector

`mod` Pass the lower limit for grade `"MODERATE"`

Permitted Value: A numeric vector

`sev` Pass the lower limit for grade `"SEVERE"`

Permitted Value: A numeric vector

Value

The Input data with the new severity records for Redness and swelling which is specified in `faobj_values` and AVAL, AVALC will be derived and `FATESTCD`, `FATEST` will be changed as per the values.

Note

Basically, This function will derive and create the severity records from the diameter record for the particular events specified in the `faobj_values` that user wants. If you want to derive the Severity from diameter, even though you have the severity in SDTM data. This function will re-derive the severity and remove the derived SDTM severity records.

Author(s)

Arjun Rubalingam

See Also

Other der_rec: `derive_fever_records()`

Examples

```
library(dplyr)
library(admiral)
library(tibble)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~AVAL, ~AVALC, ~ATPTREF, ~FATEST, ~FATESTCD,
  "XYZ1001", "REDNESS", 7.5, "7.5", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "REDNESS", 3.5, "3.5", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "REDNESS", 2, "2", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "REDNESS", 1.8, "1.8", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "REDNESS", 1.4, "1.4", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1002", "REDNESS", 11.1, "11.1", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "REDNESS", 7.4, "7.4", "VACCINATION 2", "Diameter", "DIAMETER",
```

```
  "XYZ1002", "REDNESS", 6, "6", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "REDNESS", 2.1, "2.1", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "REDNESS", 1.1, "1.1", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1001", "SWELLING", 5.5, "5.5", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "SWELLING", 2.5, "2.5", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "SWELLING", 2, "2", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "SWELLING", 1.8, "1.8", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1001", "SWELLING", 1.4, "1.4", "VACCINATION 1", "Diameter", "DIAMETER",
  "XYZ1002", "SWELLING", 10.1, "10.1", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "SWELLING", 7.1, "7.1", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "SWELLING", 5, "5", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "SWELLING", 1.8, "1.8", "VACCINATION 2", "Diameter", "DIAMETER",
  "XYZ1002", "SWELLING", 1.4, "1.4", "VACCINATION 2", "Diameter", "DIAMETER"
)

derive_diam_to_sev_records(
  dataset = input,
  faobj_values = c("REDNESS", "SWELLING"),
  diam_code = "DIAMETER",
  testcd_sev = "SEV",
  test_sev = "Severity"
)
```

`derive_fever_records` *Creating Fever Records*

Description

Creating Fever records from the VS SDTM dataset.

Usage

```
derive_fever_records(dataset, dataset_source, filter_source, faobj)
```

Arguments

`dataset`	Input Dataset Input dataset is expected to have variables USUBJID and FAOBJ.
`dataset_source`	Source Dataset - SDTM Vital Sign (VS) Source Dataset (VS) is expected to have temperature records.
`filter_source`	Filter condition for Source dataset.
`faobj`	FAOBJ Value for fever records in output dataset.

Details

Check if FAOBJ = FEVER record is present in input dataset, if not then use SDTM.VS to get FEVER records. With temperature values from VSSTRESN we decide if FEVER has occurred or not (FAORRES = "Y"/"N"). Since records are derived, these FEVER records are considered DTYPE = "DERIVED" if FAOBJ = FEVER record is present, then input dataset will be made as output with no further analysis.

The temperature value greater or equal 38° C will be considered as FEVER records.

Value

The output dataset contains records with FATESTCD = "OCCUR" for FAOBJ = FEVER records.

Author(s)

Dhivya Kanagaraj

See Also

Other der_rec: derive_diam_to_sev_records()

Examples

```
library(tibble)
library(dplyr)
library(admiraldev)
library(admiral)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~FATESTCD, ~FACAT, ~FASCAT, ~FATPT,
  "ABC101", "REDNESS", "SEV", "REACTOGENICITY", "ADMINISTRATIVE SITE", "DAY 1",
  "ABC101", "REDNESS", "DIAM", "REACTOGENICITY", "ADMINISTRATIVE SITE", "DAY 2",
  "ABC101", "VOMITTING", "SEV", "REACTOGENICITY", "SYSTEMIC", "DAY 1",
  "ABC101", "FATIQUE", "OCCUR", "REACTOGENICITY", "SYSTEMIC", "DAY 3"
)

vs <- tribble(
  ~USUBJID, ~VSTESTCD, ~VSCAT, ~VSSTRESN, ~VSSTRESU, ~VSTPT,
  "ABC101", "TEMP", "REACTOGENICITY", 38.3, "C", "DAY 1",
  "ABC101", "TEMP", "REACTOGENICITY", 38, "C", "DAY 2",
  "ABC101", "TEMP", "REACTOGENICITY", 36, "C", "DAY 3",
  "ABC101", "TEMP", "REACTOGENICITY", 37, "C", "DAY 4",
  "ABC101", "TEMP", "REACTOGENICITY", 39, "C", "DAY 5",
  "ABC101", "TEMP", "REACTOGENICITY", 39, "C", "DAY 6",
  "ABC101", "TEMP", "REACTOGENICITY", 38, "C", "DAY 7"
)

derive_fever_records(
  dataset = input,
  dataset_source = vs,
  filter_source = VSCAT == "REACTOGENICITY" & VSTESTCD == "TEMP",
  faobj = "FEVER"
)
```

---

derive_vars_crit *Derive Analysis Criterion Evaluation Variables*

Description

Derive analysis criterion evaluation result variable, paired with character and numeric flags. This function allows also the derivation of a CRIT like variable with a different name (ex: ANL01FL), without generating additional numeric (ex: ANL01FN) and character label (ex: ANL01) variables.

Usage

```
derive_vars_crit(dataset, prefix, crit_label, condition, criterion)
```

Arguments

dataset	Input dataset
prefix	Variables to add The analysis criterion evaluation variable's name (i.e., CRIT1) This name is also used in order to create both character and numeric flags variables (i.e., CRIT1FL and CRIT1FN). If the name does not contain CRIT wording, it generates a flag variable (ex: ANL01FL) whose logic is equals to CRIT1 variable, without generating additional numeric (ex: ANL01FN) and character (ANL01) variables.
crit_label	Criterion value A text description defining the condition necessary to satisfy the presence of the criterion
condition	Condition for selecting a subset The condition specified in order to select a subset from the input dataset in which the rule is applied.
criterion	Criterion rule The criterion that each selected row satisfies or not. Returns Y or N for character variable and 1 or 0 for numeric variable if the criterion is met or not, respectively. Returns NA for not selected rows (not taken into account from condition)

Value

Dataset with criterion variables

Author(s)

Federico Baratin

See Also

Other der_var: derive_var_aval_adis(), derive_vars_event_flag(), derive_vars_max_flag(), derive_vars_merged_vaccine(), derive_vars_params(), derive_vars_vaxdt()

Examples

```
library(tibble)
library(admiral)
library(admiraldev)
library(dplyr)
```

```
input <- tribble(
  ~USUBJID, ~AVISITN, ~ISCAT, ~PARAMCD, ~AVAL, ~ISLLOQ,
  "999999-000001", 10, "IMMUNOLOGY", "J0033VN", 2, 4,
  "999999-000001", 10, "IMMUNOLOGY", "I0019NT", 3, 6,
  "999999-000001", 10, "IMMUNOLOGY", "M0019LN", 4, 4,
  "999999-000001", 10, "IMMUNOLOGY", "R0003MA", 3, 6,
  "999999-000001", 30, "IMMUNOLOGY", "J0033VN", 60, 4,
  "999999-000001", 30, "IMMUNOLOGY", "I0019NT", 567, 6,
  "999999-000001", 30, "IMMUNOLOGY", "M0019LN", 659, 4,
  "999999-000001", 30, "IMMUNOLOGY", "R0003MA", 250, 6,
  "999999-000002", 10, "IMMUNOLOGY", "J0033VN", 2, 4,
  "999999-000002", 10, "IMMUNOLOGY", "I0019NT", 7, 6,
  "999999-000002", 10, "IMMUNOLOGY", "M0019LN", 5, 4,
  "999999-000002", 10, "IMMUNOLOGY", "R0003MA", 3, 6,
  "999999-000002", 30, "IMMUNOLOGY", "J0033VN", 55, 4,
  "999999-000002", 30, "IMMUNOLOGY", "I0019NT", 89, 6,
  "999999-000002", 30, "IMMUNOLOGY", "M0019LN", 990, 4,
  "999999-000002", 30, "IMMUNOLOGY", "R0003MA", 340, 6,
  "999999-000003", 10, "IMMUNOLOGY", "J0033VN", 3, 4,
  "999999-000003", 10, "IMMUNOLOGY", "I0019NT", 6, 6,
  "999999-000003", 10, "IMMUNOLOGY", "M0019LN", 2, 4,
  "999999-000003", 10, "IMMUNOLOGY", "R0003MA", 2, 6,
  "999999-000003", 30, "IMMUNOLOGY", "J0033VN", 45, 4,
  "999999-000003", 30, "IMMUNOLOGY", "I0019NT", 381, 6,
  "999999-000003", 30, "IMMUNOLOGY", "M0019LN", 542, 4,
  "999999-000003", 30, "IMMUNOLOGY", "R0003MA", NA, 6
)


derive_vars_crit(
  dataset = input,
  prefix = "CRIT1",
  crit_label = "Titer >= ISLLOQ",
  condition = !is.na(AVAL) & !is.na(ISLLOQ),
  criterion = AVAL >= ISLLOQ
)
```

`derive_vars_event_flag` *Adds Flag Variables for an Occurred Event .*

Description

Creates two flag variables for the event occurred, one for the event occurred within each by group and one to flag if the event occurred or not for each day.

Usage

```
derive_vars_event_flag(
  dataset,
  by_vars,
  aval_cutoff,
  new_var1 = NULL,
  new_var2 = NULL
)
```

Arguments

dataset	Input dataset The variables specified by the by_vars argument are expected.
by_vars	Grouping variables The variables to be considered for grouping for creating a new variable new_var1
aval_cutoff	Cutoff value for AVAL For TESTCD code list values based on diameter, if AVAL is greater than aval_cutoff then the event is considered to have occurred. For example, if aval_cutoff = 2.5 then the subjects with AVAL value greater than 2.5 are considered.
new_var1	Name of the new flag variable 1 A new flag variable will be created with values `Y` or `N`. If the event is occurred at least once during a observation period for a subject then the new variable will be flagged as `Y` otherwise `N`.
new_var2	Name of the new flag variable 2. A new flag variable will be created with values `Y` or `N`. If the event is occurred on the particular day then the new variable will be flagged as `Y` otherwise `N`.

Details

The event is considered to have occurred if AVAL is greater than the aval_cutoff or AVALC has values Y, MILD, MODERATE, SEVERE. In all other cases, the event is not considered to have occurred.

The names for the new flag variables created will be sponsor specific.

For the new_var1 it will flag all observations as "Y" within the by group if the event occurred at least once during observation period. If the event is not at all occurred during the observation period then all the observations within by group will be flagged as "N".

For derived maximum records in FATESTCD , the new_var2 will be set to NA.

If both new_var1 and new_var2 are NULL, this function will return the input dataset as output dataset.

Value

The dataset with the flag variables added to it.

See Also

Other der_var: derive_var_aval_adis(), derive_vars_crit(), derive_vars_max_flag(), derive_vars_merged_vac
derive_vars_params(), derive_vars_vaxdt()

Examples

```
library(tibble)
library(admiral)
library(dplyr)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~ATPTREF, ~AVAL, ~AVALC, ~FATEST, ~FATESTCD, ~FASCAT,
  "1", "REDNESS", "VAC1", 3.5, "3.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC1", 4.5, "4.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC1", 1.5, "1.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC1", 4.5, "4.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "FATIGUE", "VAC1", 1, "MILD", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC1", 2, "MODERATE", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC1", 0, "NONE", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC1", 2, "MODERATE", "Severity", "SEV", "SYSTEMIC",
  "1", "REDNESS", "VAC2", 6.5, "6.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC2", 7.5, "7.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC2", 2.5, "2.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "REDNESS", "VAC2", 7.5, "7.5", "Diameter", "DIAMETER", "ADMIN-SITE",
  "1", "FATIGUE", "VAC2", 1, "MILD", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC2", 2, "MODERATE", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC2", 0, "NONE", "Severity", "SEV", "SYSTEMIC",
  "1", "FATIGUE", "VAC2", 2, "MODERATE", "Severity", "SEV", "SYSTEMIC",
)

derive_vars_event_flag(
  dataset = input,
  by_vars = exprs(USUBJID, FAOBJ, ATPTREF),
  aval_cutoff = 2.5,
  new_var1 = EVENTL,
  new_var2 = EVENTDL
)
```

---

derive_vars_max_flag *Creating ANLxxFL Variables To Flag The Maximum Records*

Description

Adds Flags variables for maximum record per subject per event for overall and per vaccination

Usage

```
derive_vars_max_flag(dataset, flag1 = "ANL01FL", flag2 = "ANL02FL")
```

Arguments

dataset	Input dataset
flag1	Flags the maximum record per subject per event per vaccination. *Permitted value: Any variable name as a string or NULL.* NULL denotes not to create the flag
flag2	Flags the maximum record per subject per event for Overall *Permitted value: Any variable name as a string or NULL.* NULL denotes not to create the flag

Details

This utility flags the maximum record per subject per event per vaccination/Overall If both parameters flag1 & flag2 are passed as NULL then utility will throw error and flags will not be created.

Value

The output dataframe with ANLxxFL flags

Author(s)

Dhivya Kanagaraj

See Also

Other der_var: derive_var_aval_adis(), derive_vars_crit(), derive_vars_event_flag(), derive_vars_merged_vaccine(), derive_vars_params(), derive_vars_vaxdt()

Examples

```
library(dplyr)
library(admiraldev)
library(admiral)
library(tibble)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~FATESTCD, ~FATPTREF, ~AVAL, ~FATPT, ~PARAMCD,
  "ABC101", "REDNESS", "DIAMETER", "VACC 1", 10, "DAY 1", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 1", 7, "DAY 2", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 2", 3, "DAY 1", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 2", 8, "DAY 2", "DIARE",
  "ABC101", "FATIQUE", "SEV", "VACC 1", 1, "DAY 1", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 1", 1, "DAY 2", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 2", 2, "DAY 1", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 2", 3, "DAY 2", "SEVFAT"
)

derive_vars_max_flag(
  dataset = input,
  flag1 = "ANL01FL",
  flag2 = "ANL02FL"
```

```
)
derive_vars_max_flag(
  dataset = input,
  flag1 = NULL,
  flag2 = "ANL02FL"
)
derive_vars_max_flag(
  dataset = input,
  flag1 = "ANL01FL",
  flag2 = NULL
)
```

`derive_vars_merged_vaccine` *Add New Variable(s) to the Input dataset Based on Variables from Another dataset*

Description

Add new variables to the input dataset based on variables from another dataset. The variables to be added to the output dataset will be based on input variables passed on `ex_vars` argument.

Usage

```
derive_vars_merged_vaccine(
  dataset,
  dataset_ex,
  by_vars_sys,
  by_vars_adms,
  ex_vars
)
```

Arguments

`dataset`	Input dataset which should have been combined with the supplementary(if exists). The variables specified by the `by_vars` argument inside the `derive_vars_mergedare` expected.
`dataset_ex`	`ex` dataset(combined with `suppex`) to merge with the input dataset. The variables specified by the `ex_vars` argument are expected.
`by_vars_sys`	Grouping variables for systemic events.
`by_vars_adms`	Grouping variables for administration site events.
`ex_vars`	Variables to be added to the output dataset from EX dataset

Details

The input dataset will be merged with EX dataset for "ADMINISTRATION SITE" and "SYSTEMIC" categories separately and these datasets will be bound together as the final output dataset.

This function is intended to add only EX variables to the input dataset and user is expected to handle if any pre-processing is required.

Only the variables passed to the ex_vars will be added in the output dataset

If the input dataset has multiple vaccination for a subject at same visit then this function will not merge ex dataset and will return the dataset.

Value

The dataset with variables added from the EX dataset.

Author(s)

Vikram S

See Also

Other der_var: derive_var_aval_adis(), derive_vars_crit(), derive_vars_event_flag(), derive_vars_max_flag(), derive_vars_params(), derive_vars_vaxdt()

Examples

```
library(tibble)
library(admiral)
library(dplyr)
library(pharmaversesdtm)

derive_vars_merged_vaccine(
  dataset = face_vaccine,
  dataset_ex = ex_vaccine,
  by_vars_sys = exprs(USUBJID, FATPTREF = EXLNKGRP),
  by_vars_adms = exprs(USUBJID, FATPTREF = EXLNKGRP, FALOC = EXLOC, FALAT = EXLAT),
  ex_vars = exprs(EXTRT, EXDOSE, EXDOSU, EXSTDTC, EXENDTC)
) %>%
  select(USUBJID, FATPTREF, FALOC, FALAT, EXTRT, EXDOSE, EXDOSU, EXSTDTC, EXENDTC) %>%
  head(10)

derive_vars_merged_vaccine(
  dataset = face_vaccine,
  dataset_ex = ex_vaccine,
  by_vars_sys = exprs(USUBJID, FATPTREF = EXLNKGRP),
  by_vars_adms = exprs(USUBJID, FATPTREF = EXLNKGRP, FALOC = EXLOC, FALAT = EXLAT),
  ex_vars = exprs(EXTRT, EXDOSE, EXDOSU, EXSTDTC, EXENDTC)
)
```

derive_vars_params *Assigning Parameter Variables*

Description

Creating PARAMCD from lookup dataset and assigning PARAM,PARAMN,PARCAT1, PARCAT2 variables

Usage

```
derive_vars_params(dataset, lookup_dataset, merge_vars)
```

Arguments

dataset	Input dataset Input dataset is expected to have variables USUBJID,FAOBJ, FACAT, FATESTCD and FATEST
lookup_dataset	lookup dataset containing PARAMCD values for every unique FATESTCD and FAOBJ lookup dataset is expected to have variables FATEST, PARAMCD, FATESTCD, FAOBJ and one entry for every unique FATESTCD and FAOBJ
merge_vars	List of Variables need to be merged from lookup dataset

Details

A lookup dataset is required with PARAMCD values for every combination of FATEST & FAOBJ. PARAMCD PARAMN PARAMN PARCAT1 PARCAT2 values can be assigned from lookup dataset.

```
if `PARAMN` not assigned in lookup dataset then
`PARAMN` is assigned with a unique number for every unique PARAM value.
if `PARAM` value not assigned in lookup dataset then
`PARAM` value is a combination of `FAOBJ` `FATEST` `FASTRESU` `FALOC`
`FADIR` `FALAT`
if `PARCAT1` value not assigned in lookup dataset then
`PARCAT1` is assigned as `FACAT`
if `PARCAT2` value not assigned in lookup dataset then
`PARCAT2` is assigned as `FASCAT`
```

Value

The output dataset contains all observations and variables of the input dataset along with PARAM,PARAMCD,PARCAT1,PARCAT2,PA

Author(s)

Dhivya Kanagaraj

See Also

Other der_var: derive_var_aval_adis(), derive_vars_crit(), derive_vars_event_flag(), derive_vars_max_flag(), derive_vars_merged_vaccine(), derive_vars_vaxdt()

Examples

```
library(admiral)
library(tibble)
library(dplyr)

lookup_dataset <- tibble::tribble(
  ~FATESTCD, ~PARAMCD, ~PARAMN, ~FATEST, ~FAOBJ,
  "SEV", "SEVREDN", 1, "Severity", "Redness",
  "DIAMETER", "DIARE", 2, "Diameter", "Redness",
  "MAXDIAM", "MDIRE", 3, "Maximum Diameter cm", "Redness",
  "MAXTEMP", "MAXTEMP", 4, "Maximum Temperature", "Fever",
  "OCCUR", "OCFEVER", 5, "Occurrence Indicator", "Fever",
  "OCCUR", "OCERYTH", 6, "Occurrence Indicator", "Erythema",
  "SEV", "SEVPAIN", 7, "Severity", "Pain at Injection site",
  "OCCUR", "OCPAIN", 8, "Occurrence Indicator", "Pain at Injection site",
  "OCCUR", "OCSWEL", 9, "Occurrence Indicator", "Swelling"
)

input <- tibble::tribble(
  ~USUBJID, ~FACAT, ~FASCAT, ~FATESTCD, ~FAOBJ, ~FATEST, ~FALOC, ~FALAT,
  "ABC101", "REACTO", "ADMIN", "SEV", "Redness", "Severity", "ARM", "LEFT",
  "ABC101", "REACTO", "ADMIN", "DIAMETER", "Redness", "Diameter", "ARM", "RIGHT",
  "ABC101", "REACTO", "ADMIN", "MAXDIAM", "Redness", "Maximum Diameter", NA, NA,
  "ABC101", "REACTO", "SYSTEMIC", "MAXTEMP", "Fever", "Maximum Temp", NA, NA,
  "ABC101", "REACTO", "SYSTEMIC", "OCCUR", "Fever", "Occurrence", NA, NA,
  "ABC101", "REACTO", "ADMIN", "OCCUR", "Erythema", "Occurrence", NA, NA,
  "ABC101", "REACTO", "ADMIN", "SEV", "Swelling", "Severity", NA, NA,
  "ABC101", "REACTO", "ADMIN", "OCCUR", "Swelling", "Occurrence", NA, NA,
  "ABC101", "REACTO", "ADMIN", "OCCUR", "Swelling", "Occurrence", NA, NA
)

derive_vars_params(
  dataset = input,
  lookup_dataset = lookup_dataset,
  merge_vars = exprs(PARAMCD, PARAMN)
)
```

---

derive_vars_vaxdt *Add Vaccination Date Variables to the Output Dataset*

Description

Creates vaccination date variables from EX domain. A date variable will be created for each vaccination taking values from the variable EXSTDTC.

Usage

```
derive_vars_vaxdt(dataset, dataset_adsl, by_vars, order)
```

Arguments

dataset	Input dataset The variables specified by the by_vars argument are expected.
dataset_adsl	Input adsl dataset The vaccination date variables created will be merged with this adsl dataset.
by_vars	Grouping variables. The variables to be grouped to filter the first observation within each by group.
order	Sorting variables. The variables order to be specified either in ascending or descending order. By default ascending order will be applicable.

Details

If there are multiple vaccinations for a visit per subject,warning will be provided and only first observation will be filtered based on the variable order specified on the order argument. In this case, user need to select the by_vars appropriately.

The number of variables created will be based on the number of vaccinations per subject per visit.

Value

The adsl dataset with vaccination date variables added to it.

Author(s)

Vikram S

See Also

Other der_var: derive_var_aval_adis(), derive_vars_crit(), derive_vars_event_flag(), derive_vars_max_flag(), derive_vars_merged_vaccine(), derive_vars_params()

Examples

```
library(tibble)
library(admiral)
library(dplyr)

input <- tribble(
  ~USUBJID, ~EXSTDTC, ~VISITNUM, ~EXTRT, ~EXLNKGRP, ~VISIT,
  "A001", "2015-01-10", 1, "DRUG A", "VAC 1", "VISIT 1",
  "A001", "2015-01-11", 2, "DRUG A", "VAC 2", "VISIT 2",
  "A001", "2015-01-12", 3, "DRUG B", "VAC 3", "VISIT 3",
  "A002", "2015-01-13", 1, "DRUG B", "VAC 1", "VISIT 1",
  "A002", "2015-01-14", 2, "DRUG C", "VAC 2", "VISIT 2"
)

adsl <- tribble(
  ~USUBJID, ~SEX, ~AGE,
```

```
  "A001", "MALE", 23,
  "A002", "FEMALE", 26,
)

derive_vars_vaxdt(
  dataset = input,
  dataset_adsl = adsl,
  by_vars = exprs(USUBJID, VISITNUM),
  order = exprs(USUBJID, VISITNUM, VISIT, EXSTDTC)
)
```

`derive_var_aval_adis` *Derive AVAL variable for ADIS ADaM domain*

Description

Derive AVAL variable for Laboratory Immunology Data ADaM domain. A common rule has been decided for its derivation, based on ISLLOQ, ISULOQ and ISORRES when both ISLLOQ and ISULOQ are present. If ISULOQ is not present, the variables used are ISLLOQ and ISORRES. Please, refers to arguments description for additional details.

Usage

```
derive_var_aval_adis(
  dataset,
  lower_rule,
  middle_rule,
  upper_rule = NULL,
  round = NULL
)
```

Arguments

dataset	Input dataset.
lower_rule	Derivation rule when ISSTRESN value is below ISLLOQ. When ISSTRESN is missing, the inequality in ISORRES is checked for the derivation.
middle_rule	Derivation rule when ISSTRESN value is greater than ISLLOQ and lower than ISULOQ. If ISULOQ is not present, derivation rule when ISSTRESN is greater than ISLLOQ. When ISSTRESN is missing, the inequality in ISORRES is checked for the derivation.
upper_rule	Derivation rule when ISSTRESN value is greater than ISULOQ. This is an optional argument since ISULOQ may not be present. When ISSTRESN is missing, the inequality in ISORRES. is checked for the derivation. Default value is NULL.
round	Rounding for AVAL variable. An integer argument which specifies the number of decimals displayed. Default value is NULL.

Value

Dataset with AVAL variable derived.

Author(s)

Federico Baratin

See Also

Other der_var: derive_vars_crit(), derive_vars_event_flag(), derive_vars_max_flag(), derive_vars_merged_vaccine(), derive_vars_params(), derive_vars_vaxdt()

Examples

```
library(tibble)
library(admiral)
library(admiraldev)
library(dplyr)
library(rlang)

input <- tribble(
  ~USUBJID, ~AVISITN, ~PARAMCD, ~PARAM, ~ISORRES, ~ISSTRESN, ~ISLLOQ, ~ISULOQ,
  "ABC-1001", 10, "J0033VN", "J0033VN Antibody", NA, NA, 2, 100,
  "ABC-1001", 10, "I0019NT", "I0019NT Antibody", "3", 3.0, 4, 200,
  "ABC-1001", 10, "M0019LN", "M0019LN Antibody", ">150", NA, 8, 150,
  "ABC-1001", 10, "R0003MA", "R0003MA Antibody", "140.5", 140.5, 4, 120,
  "ABC-1001", 30, "J0033VN", "J0033VN Antibody", "2", 2.0, 2, 100,
  "ABC-1001", 30, "I0019NT", "I0019NT Antibody", NA, NA, 4, 200,
  "ABC-1001", 30, "M0019LN", "M0019LN Antibody", NA, NA, 8, 150,
  "ABC-1001", 30, "R0003MA", "R0003MA Antibody", "98.2", 98.2, 4, 120,
  "ABC-1001", 10, "J0033VNL", "LOG10 (J0033VN Antibody)", NA, NA, 2, 100,
  "ABC-1001", 10, "I0019NTL", "LOG10 (I0019NT Antibody)", "3", 3.0, 4, 200,
  "ABC-1001", 10, "M0019LNL", "LOG10 (M0019LN Antibody)", ">150", NA, 8, 150,
  "ABC-1001", 10, "R0003MAL", "LOG10 (R0003MA Antibody)", "140.5", 140.5, 4, 120,
  "ABC-1001", 30, "J0033VNL", "LOG10 (J0033VN Antibody)", "2", 2.0, 2, 100,
  "ABC-1001", 30, "I0019NTL", "LOG10 (I0019NT Antibody)", NA, NA, 4, 200,
  "ABC-1001", 30, "M0019LNL", "LOG10 (M0019LN Antibody)", NA, NA, 8, 150,
  "ABC-1001", 30, "R0003MAL", "LOG10 (R0003MA Antibody)", "98.2", 98.2, 4, 120,
  "ABC-1002", 10, "J0033VN", "J0033VN Antibody", "3", 3.0, 2, 100,
  "ABC-1002", 10, "I0019NT", "I0019NT Antibody", NA, NA, 4, 200,
  "ABC-1002", 10, "M0019LN", "M0019LN Antibody", NA, NA, 8, 150,
  "ABC-1002", 10, "R0003MA", "R0003MA Antibody", "48.9", 48.9, 4, 120,
  "ABC-1002", 30, "J0033VN", "J0033VN Antibody", NA, NA, 2, 100,
  "ABC-1002", 30, "I0019NT", "I0019NT Antibody", NA, NA, 4, 200,
  "ABC-1002", 30, "M0019LN", "M0019LN Antibody", "5", 5.0, 8, 150,
  "ABC-1002", 30, "R0003MA", "R0003MA Antibody", "228.1", 228.1, 4, 120
)

derive_var_aval_adis(
  dataset = input,
  lower_rule = ISLLOQ / 2,
  middle_rule = ISSTRESN,
```

```
  upper_rule = ISULOQ,
  round = 2
)
```

`max_flag` *Creating Maximum Flag*

Description

To Flag the maximum records depends on the grouping variables in a flag variable.

Usage

```
max_flag(dataset, by_vars, fl)
```

Arguments

`dataset`	Input dataset
`by_vars`	By variables which goes to group by, to create the flag. Pass the variables inside the exprs().
`fl`	Flag variable name, Pass it as string.

Value

Data frame with flag variable which is flagged for the maximum value records depending on the variables passed in `by_vars` by user.

Author(s)

Dhivya Kanagaraj

Examples

```
library(tibble)
library(admiral)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~FATESTCD, ~FATPTREF, ~AVAL, ~FATPT, ~PARAMCD,
  "ABC101", "REDNESS", "DIAMETER", "VACC 1", 10, "DAY 1", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 1", 7, "DAY 2", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 2", 3, "DAY 1", "DIARE",
  "ABC101", "REDNESS", "DIAMETER", "VACC 2", 8, "DAY 2", "DIARE",
  "ABC101", "FATIQUE", "SEV", "VACC 1", 1, "DAY 1", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 1", 1, "DAY 2", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 2", 2, "DAY 1", "SEVFAT",
  "ABC101", "FATIQUE", "SEV", "VACC 2", 3, "DAY 2", "SEVFAT"
)
```

```
max_flag(
  dataset = input,
  by_vars = exprs(USUBJID, FAOBJ, FATPTREF, PARAMCD),
  fl = "ANL01FL"
)
```

---

post_process_reacto *Post processing function for ADFACE dataset*

Description

This is used to do post processing for ADaM reactogenicity dataset, for the derived SDTM level records, the corresponding values in FA variables will be NA.

Usage

```
post_process_reacto(
  dataset,
  filter_dataset = FATESTCD %in% c("MAXDIAM", "MAXSEV", "MAXTEMP") | (FATESTCD ==
    "OCCUR" & FAOBJ == "FEVER")
)
```

Arguments

dataset	Input dataset
filter_dataset	Filter condition Conversion of records in FA variables to NA depends on this condition.

Value

The input dataframe with NA values in FA variables where the SDTM records modified for ADaM derivation purpose.

Author(s)

Arjun Rubalingam

Examples

```
library(dplyr)
library(admiral)
library(tibble)

input <- tribble(
  ~USUBJID, ~FAOBJ, ~FALAT, ~FACAT, ~FASCAT, ~FATPT, ~FATESTCD, ~PARAMCD, ~AVAL,
  "ABC-1001", "FEVER", NA, "REACTO", "SYS", "DAY 1", "MAXTEMP", "MAXTEMP", 39.4,
  "ABC-1001", "VOMITING", NA, "REACTO", "SYS", "DAY 4", "MAXSEV", "MAXVOMIT", 3,
  "ABC-1001", "SWELLING", "LEFT", "REACTO", "ADMIN", "DAY 1", "MAXSEV", "MAXSWEL", 3,
```

```
  "ABC-1001", "REDNESS", "LEFT", "REACTO", "ADMIN", "DAY 2", "DIAMATER", "DIARE", 10.3,
  "ABC-1001", "FEVER", "LEFT", "REACTO", "SYS", "DAY 2", "OCCUR", "OCCFEV", NA
)

post_process_reacto(
  dataset = input,
  filter_dataset = FATESTCD %in% c("MAXSEV", "MAXTEMP") |
    (FATESTCD == "OCCUR" & FAOBJ == "FEVER")
)
```

Index